Nature-inspired Intelligent Algorithms for Optimization and Prediction Problems

by

Yanting Liu

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

2018

(Submitted June 21, 2018)

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 husband for their consistent support and encouragement.

Abstract

Nature-inspired algorithms are new optimization ones which are on the basis of synergetic evolution in recent years. They provide new ways to find solutions of complex problems. Because of intelligence, versatility, essence parallelism and global search ability of them, they have shown potential and charm in the field of computer science, knowledge discovery, communication network, robot and so on. They have become research hotspots in the field of intelligent computing. In this paper, several nature-inspired intelligent algorithms are proposed to solve different prediction and optimization problems.

The thesis is organized as in the following.

In chapter 1, we systematically reviewed Natural computing, Artificial intelligence, Machine learning, Heuristic and Metaheuristic.

In chapter 2, we introduce two kinds of nature-inspired algorithms: Decision tree and Gray wolf optimization.

In chapter 3, we built a decision tree as a quick and reliable method adapted in all road segments, and propose a new method to make the vehicle state prediction based on the decision tree.

In chapter 4, we investigate twelve different kinds of chaotic maps to give some insights into the influence of Chaotic Local Search (CLS) on Grey Wolf Optimization

(GWO). Experimental results based on 29 widely used benchmark functions suggest that CLS indeed enables GWO to reach better performance in terms of accuracy, distribution and convergence property of solutions.

In chapter 5 , there are conclusions and future research. We introduced Nature-inspired intelligent algorithms for optimization and prediction problems. And in the future, I will go a step further on various kinds of algorithm mechanisms of Nature-inspired intelligent algorithms. Then, improve the performance of the current existent Nature-inspired intelligent algorithms and apply them to solve problems in new fields. Last but not least, Nature-inspired intelligent algorithms can combine with other computational intelligence algorithms for solving much complex problems.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Natural computing [2], also known as natural computation, is a term that encompasses three types of methods: 1) inspiration from nature for developing new problem-solving techniques; 2) computer-based synthesis of natural phenomena; and 3) the use of Natural materials (e.g., molecules) are those calculated [3]. The main research areas that make up these three branches are artificial neural network, evolutionary algorithm, group intelligence, artificial immune system and so on.

The computational paradigm of natural computational research is abstracted from many natural phenomena such as self-replication, brain function, Darwinian evolution, group behavior, the immune system, the defining properties of life forms, cell membranes and morphogenesis. People can think of what happens in nature as information processing.

1.1 Artificial Intelligence

Artificial Intelligence (AI), is the intelligence displayed by a machine, as opposed to the Natural Intelligence (NI) shown by humans and other animals, to the intelligence shown by machines made by humans. In computer science, artificial intelligence

research is defined as ”intelligent agent” research: any device that perceives its environment and takes action to maximize the chance of success with a given goal [4]. Usually artificial intelligence refers to the human-like intelligence technology that is realized by a common computer program. The term also refers to the field of science that investigates whether such intelligent systems are achievable and how they are implemented.

The traditional problems (or goals) of artificial intelligence research include reasoning, knowledge, planning, learning, natural language processing, the ability to perceive and move and manipulate objects. General information is one of the long-term goals of the field [4]. Methods include statistical methods to compute intelligence and traditional symbolic AI [5]. Artificial intelligence is now playing a more extensive role in the field of computers. And in the robot, economic and political decision-making, control systems, simulation systems have been applied.

Earlier researchers developed algorithms that mimic the step-by-step reasoning people use to solve puzzles or logical inferences [6]. By the late 1980s and 1990s, artificial intelligence research had developed methods of dealing with uncertain or incomplete information, using concepts of probability and economics [7].

For difficult problems, the algorithm may require huge computational resources. Searching for more efficient problem solving algorithms is a high priority [8]. During more than 60 years of research, AI has developed numerous tools to solve the thorniest issue in computer science. The following is a discussion of the most common of these methods.

1.1.1 Search and Optimization

Many of the problems in artificial intelligence can theoretically be solved by intelligently searching for many possible solutions: Inference can be simplified to performing a search. Many learning algorithms use optimization-based search algorithms. The solution to many problems is to use ”heuristics” or ”rules of thumb” to prioritize choices that are more likely to achieve their goals, and take shorter steps [9]. In some search methods, heuristics can also completely eliminate some choices that are less likely to result in a goal (called ”pruning search trees”). Heuristic methods provide ”best guess” for the path where the solution is located. Heuristic methods limit the search solution to a smaller sample size.

1.1.2 Logic

Logic is used for knowledge representation and problem solving, but it can also be applied to other problems. There are several different forms of logic used in AI research [6]. Proposition or sentence logic is the logic of a statement that can be true or false. First-order logic also allows the use of quantifiers and predicates, and can express facts about objects, their properties, and the relationships between them [9]. Fuzzy logic is a version of first-order logic that allows the truth value of a statement to be represented as a value between 0 and 1, not just true (1) or false (0). Subjective logic simulates uncertainty in a more explicit way than fuzzy logic: a given binomial perspective satisfies belief + untrusted + uncertainty = 1 within the Beta distribution [10]. In this way, ignorance can be distinguished from probabilistic statements made highly by agents. Default Logic, Non-monotonic Logic and Limitations are logical

forms designed to help with default reasoning and qualification issues.

1.1.3 Classifier and Statistical Learning Methods

The simplest AI applications can be divided into two types: classifiers and controllers. However, the controller also sorts the conditions before inferring the action, so classification is a central part of many AI systems. A classifier is a function that uses pattern matching to determine the closest match. They can be adjusted according to the instance, making it very attractive in the AI. In supervised learning, each pattern belongs to a predefined category. A class can be seen as a decision that must be made [11]. All the observations combined with their class labels are called datasets. When new observations are received, the observations are categorized based on past experience. Classifiers can be trained in a variety of ways; there are many methods of statistics and machine learning. The most widely used classifiers are neural networks, kernel support vector machines, k-nearest neighbor algorithms, Gaussian mixture models, naive Bayesian classifiers, and decision trees. The performance of these classifiers has been compared over a wide range of tasks. The performance of a classifier depends very much on the nature of the data to be classified. No classifier works best for all given problems; it's also known as the "no free lunch" theorem. Determining a suitable classifier for a given problem is still an art rather than a science [12].

AI is related to any mental task. Modern artificial intelligence technology is everywhere. Often, when a technology reaches mainstream use, it is no longer considered artificial intelligence; this phenomenon is described as an AI effect [13].

1.2 Machine Learning

Machine learning is a branch of artificial intelligence. As a scientific work, machine learning stems from the pursuit of artificial intelligence. Artificial intelligence is from a ”reasoning” as the focus, to ”knowledge” as the focus, to ”learning” as a focus, a naturally clear context. Obviously, machine learning is a way to achieve artificial intelligence, which is to solve the problem of artificial intelligence by means of machine learning.

In the early days of AI as a discipline, some researchers were interested in learning the machine from data. They try to solve this problem in a variety of symbolic ways, as well as what are called ”neural networks.” These are mainly perceptron and other models that were later considered as a reengineering of the generalized linear statistical model. Probabilistic reasoning has also been applied, especially in automated medical diagnostics [4].

Computer analysis of machine learning algorithms and their performance is a branch of theoretical computer science, known as computational learning theory [14]. Machine learning theory is the main design and analysis of some of the computer can automatically ”learn” algorithm. Machine learning algorithm is a kind of automatic analysis from the data to obtain the law, and use the law of unknown data to predict the algorithm.

Tom M. Mitchell provided a widely quoted, more formal definition of the algorithms studied in the machine learning field: ”A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.” [15]

Machine learning has the following definitions:

1. Machine learning is a science of artificial intelligence whose main object of study is artificial intelligence, and in particular how to improve the performance of concrete algorithms in empirical learning.

2. Machine learning is a study of computer algorithms that can be automatically improved by experience [16].

3. Machine learning uses data or past experience to optimize the performance standards of computer programs.

Machine learning is widely used in biochemical informatics, computer networks, information retrieval, machine perception, economics, financial market analysis, natural language processing, search engines, sequence mining, software engineering and other fields.

1.2.1 Classification

1.2.1.1 Supervised / Unsupervised Learning

1.Supervised: classification, regression

1).Calibration training data

2).training process: According to the target output and the actual error signal output to adjust the parameters

3).typical method

- Global: BN, NN, SVM, Decision Tree
- Local: KNN, CBR (Case-base reasoning)

2.Unsupervised: probability density estimation, clustering, dimension reduction There is no calibration training data

1).Learning machine adjusts the system parameters according to the statistical rules of external data, so that the output can reflect some characteristic of the data.

2).typical method

- K-means, SOM, ...

3.Semi-supervised: EM, Co-training

1).Learning (a small amount of) calibration training data and (a large number of) uncalibrated data

2).typical method

- Co-training, EM, Latent variables, ...

1.2.1.2 Other Learning Methods

1.Reinforcement Learning

1).The external environment only gives the evaluation information rather than the correct answer to the output. The learning machine improves its performance by strengthening the rewarded actions.

- The training data contains some learning target information

2.Multi-task learning (Multi-task learning)

1).Learns a problem together with other related problems at the same time, using a shared representation.

Machine learning began to flourish in the 1990s and became an independent area. This area has shifted from the realization of artificial intelligence to the goal of solving

practical problems. It shifted the focus from the symbolic approach it inherited from AI to the methods and models borrowed from statistics and probability theory [4,17]. Increasingly digital information, and the possibility of distribution via the Internet, also benefit.

Machine learning and data mining generally take the same approach and overlap, but machine learning focuses on predictions and data mining focuses on the discovery of (previously) unknown data attributes based on known attributes learned from the training data (this is knowledge discovery in the database Analysis step) [17, 18].

1.2.2 Specific Machine Learning Algorithm

1.2.2.1 Decision Tree Learning

Decision tree learning uses a decision tree as a predictive model that maps the observations about the project to the conclusions about the project's target value [19].

1.2.2.2 Artificial Neural Networks

An Artificial Neural Network (ANN) learning algorithm, commonly referred to as a "Neural Network" (NN), is a learning algorithm that is inspired by the structure and function of biological neural networks [20]. Computation is based on interrelated sets of artificial neurons that process information using connection methods. Modern neural networks are nonlinear statistical data modeling tools. They are often used to model the complex relationships between inputs and outputs, find patterns in the data, or capture statistical structures of unknown joint probability distributions between observed variables [20].

1.2.2.3 Deep Learning

The recent decline in hardware prices and the development of personal-use GPUs have led to the development of deep learning concepts consisting of multiple hidden layers in an artificial neural network. This method attempts to simulate the way humans handle the light and sound of the brain, both visually and aurally. Some successful applications of deep learning are computer vision and speech recognition [21].

1.2.2.4 Bayesian Networks

Bayesian networks, belief networks or directed acyclic graph models are probabilistic graph models that represent a set of random variables and their conditional independence via a directed acyclic graph (DAG) [22]. For example, a Bayesian network can represent the probability relationship between a disease and a symptom. Given the symptoms, the network can be used to calculate the probability of various diseases occurring. There are effective algorithms for performing reasoning and learning.

1.2.2.5 Genetic Algorithms

Genetic Algorithm (GA) is a search heuristic that mimics the natural selection process [23]. It uses mutation and crossover methods to generate new genotypes and hopes to find a good solution to a given problem. In machine learning, genetic algorithms found some uses in the 1980s and 1990s [24]. In contrast, machine learning techniques have been used to improve the performance of genetic and evolutionary algorithms [24].

1.3 Heuristic

Heuristic is a technique, in computer science, artificial intelligence, and mathematical optimization that is a solution to problems, learning, or discovering. Used to solve problems more quickly when the classical method is too slow, or when the classic method fails to find any exact solution, they find an approximate solution. Although not guaranteed to be the best or perfect practical method, but for the immediate goal is sufficient. This is done by the best of deals, completeness, accuracy or speed accuracy. Heuristics are a strategy derived from the experience of previous similar problems. To some extent, it can be considered as a shortcut. The most basic heuristic is trial and error.

Rudolf Groner analyzes the heuristic history from the origins of ancient Greece to the contemporary work of cognitive psychology and artificial intelligence and proposes a heuristic approach to thinking, Can be verified by a confirmatory questionnaire [25].

Heuristic algorithms can be used in artificial intelligence systems while searching for solution spaces. Heuristics are derived by using some of the functions that the designer enters into the system or by adjusting the weights of the branches based on the likelihood that each branch will reach the target node.

1.4 Metaheuristic

In computer science and mathematical optimization, metaheuristic is a more advanced process or heuristic design for finding, generating, or selecting heuristics (partial search algorithms) that may provide a sufficiently good solution to an optimization

problem, in particular In the case of incomplete or imperfect information or limited computing power [26]. Metaheuristics samples a solution that is too large to be completely sampled. Metaheuristics may make very few assumptions about the optimization problems being solved, so they may be used for a variety of problems [27].

Compared with the optimization algorithm and the iterative method, metaheuristics does not guarantee that the global optimal solution can be found on some kinds of problems [27]. In combinatorial optimization, metaheuristics usually find fewer optimization algorithms, iterative methods, or simple heuristics than those with less computational complexity by searching for a large number of feasible solutions. Therefore, they are a useful way to optimize the problem [26].

These are the characteristics of most metaheuristics [27]:

1.Metaheuristics is to guide the search process strategy.

2.The goal is to effectively explore the search space to find near-optimal solutions.

3.Constitute meta-heuristic algorithm from simple local search process to complex learning process.

4.Meta-heuristic algorithm is approximate, usually nondeterministic.

5.Metaheuristics is not problem-specific.

1.4.1 Classification

These are the classification of metaheuristics [27]:

1.Local search vs. Global search

2.Single-solution vs. Population-based

3.Hybridization and memetic algorithms

4.Parallel metaheuristics

5.Nature-inspired metaheuristics

Many recent metaheuristics includes simulated annealing, evolutionary algorithms, ant colony optimization and particle swarm optimization,are inspired by natural systems [27].

In our previous research, we studied (1) Artificial neural networks which include multiple-valued logic networks [28–31], multilayered neural networks [32–41], and dendritic neuron models [42–48], (2) Evolutionary computation which involves artificial immune systems [49–61], multi-objective optimization applications [62–64], gravitational search algorithms [65–68], ant colony optimization [69], imperialist competition algorithm [70, 71], differential evolution [72], and brain storm optimization [73, 74], (3) Complex networks [75, 76], (4) Machine learning technologies [77–82], and (5) Internet of things [83, 84]. All these results give foundations and motivations of the current research of this work.

Chapter 2

Nature-inspired Algorithms

2.1 Decision Tree

Machine learning techniques that generate decision trees from data are called decision tree learning, which is commonly referred to as decision tree. Decision tree to establish and use to assist decision-making, is a special kind of tree structure. Decision tree is a decision support tool that uses a tree-like graph or decision model.

Decision tree learning is one of the predictive modeling methods used in statistics, data mining and machine learning [19]. Decision tree is a similar structure of the flow chart, he represents the object attributes and object values between mappings. Decision tree only a single output, we can establish different independent decision trees to handle complex outputs.

2.1.1 Decision Tree Types

There are two main types of decision trees used in data mining:

1.Classification tree analysis means that the forecast result is the category to which the data belong.

2.Regression tree analysis is when the predicted result can be considered a real

number.

Classification and Regression Tree (CART) analysis terms are used to refer to the general term for the above two procedures, first proposed by Breiman et al [85]. A tree model whose target variable can take a set of discrete values is called a classification tree; a decision tree whose target variable can take continuous values (usually real numbers) is called a regression tree. The algorithm for building a decision tree is usually top-down by choosing a variable that best divides the project set in each step.

2.1.2 A Decision Tree Contains Three Types of Nodes

1. Decision nodes : typically represented by squares

2. Chance nodes : typically represented by circles

3. End nodes : typically represented by triangles [86]

2.1.3 The Advantages of Decision Trees

Compared with other data mining algorithms, decision trees have advantages in the following aspects:

1. Decision tree is easy to understand and explain. People can understand the decision tree model with a brief explanation.

2. For decision trees, data preparation often requires data normalization with little data preparation.

3. Can handle both data type and regular type properties. Other technologies tend to deal with only one type of data.

4. Use the white box model. Given a model for observation, it is easy to derive the corresponding logical expression based on the resulting decision tree.

5. Easy to test the model by static test. The stability of the model can be measured.

6. Can handle large-scale data well.

2.1.4 Shortcomings of Decision Trees

1. The overly complex decision tree creation can lead to inability to predict well beyond the training set, which is called over-fitting.

2. Some problems Decision tree cannot be a good solution, such as: XOR problems. When solving this problem, the decision tree can become overly large. To solve this problem, you can only change the area of the problem or use other more time-consuming learning algorithms.

3. Training an optimal decision tree is a complete NP problem. Therefore, the practical application of decision tree training using heuristic search algorithm to achieve local optimal. Such an algorithm cannot get the optimal decision tree.

4. For data with inconsistent numbers for each category, the result of the information gain in the decision tree is biased towards those with more values.

The actual decision tree learning algorithm is based on the heuristic algorithm.

2.2 Gray Wolf Optimization

Canis lupus belongs to the Canidae family. Gray wolves are considered top predators, meaning they are at the top of the food chain. Gray wolves like to live in a bag. The

average number of groups is 5-12 people. Of particular interest are their very strict social leadership hierarchy, as shown in Fig. 1 [1].

The wolf at the top of the pyramid, called α, is responsible for making decisions about hunting practices, habitat and food distribution. The wolf is not necessarily the strongest wolf, but the best leader. Located on the second level of the pyramid, called β, is the successor to the alpha that helps α in decision-making or other group activities when the entire wolves are missing. Located on the third floor is the δ, obey and instructions, old (poor fitness) and will be reduced to level. The lowest ω is responsible for balancing the internal relationships of the population.

GWO algorithm simulates the graying system and hunting behavior of wolves in nature. As shown in Fig. 2 [1], the entire wolves are divided into four groups: α, β, δ, ω. The first three groups are, in turn, the three most well adapted groups, and the three groups direct the other wolves (ω) toward the target search. During the optimization, the wolves update the positions of α, β, δ, ω.

As mentioned above, the gray wolf surrounds prey during the hunt. In order to simulate the surrounding behavior, the following formula is proposed:

$$\vec{D} = |\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)|, \tag{2.1}$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D}, \tag{2.2}$$

$$\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a}, \tag{2.3}$$

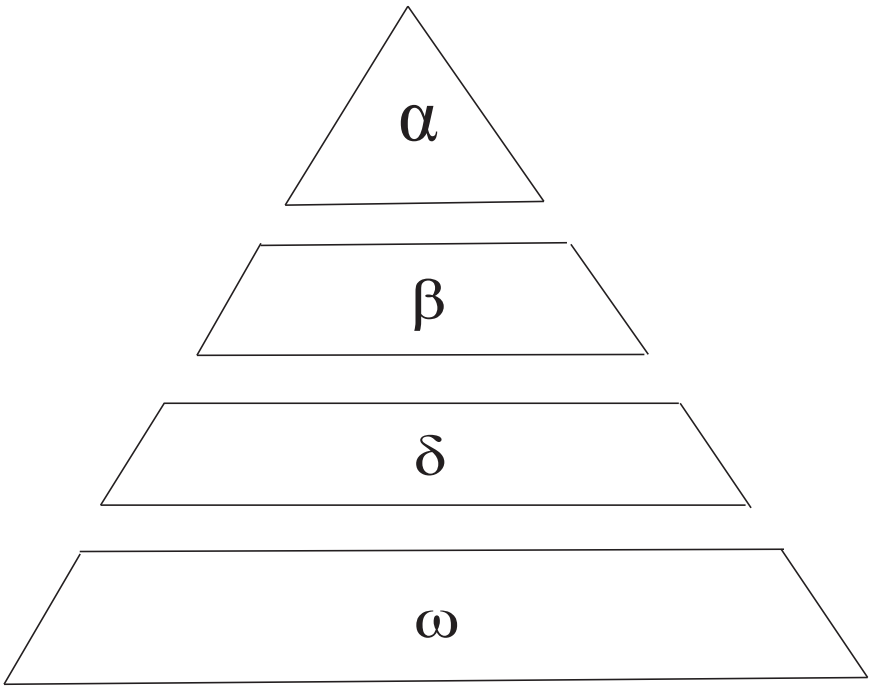

Figure 2.1: Hierarchy of grey wolf (dominance decreases from top down) [1].

$$\vec{C} = 2 \cdot \vec{r}_2. \tag{2.4}$$

$\vec{D}$ represents the distance between the individual and the food, t represents the number of current iteration, when $|\vec{A}| < 1$, the gray population shrinks the encirclement, which corresponds to the local search, $\vec{X}_p$ is the prey position, $\vec{X}$ is the wolf's position. The convergence factor $\vec{a}$ is linearly decreasing from 2 to 0 with the number of iterations. $\vec{r}_1, \vec{r}_2$ are random vectors in [0, 1].

When the gray wolf to determine the location of the prey, α lead β and δ guiding prey surrounded, because α, β, and δ are closest to the prey, so the use of the location of the three wolves to determine the approximate location of prey, Gradually approaching the prey, mathematical description [1] is as follows:

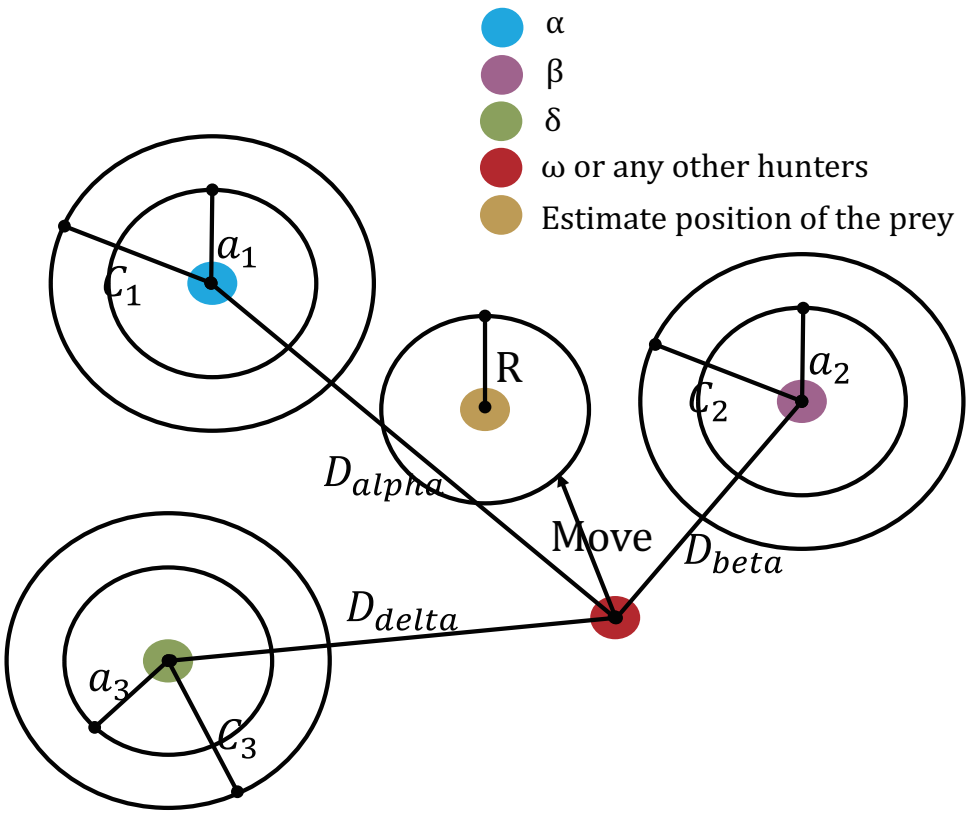

Figure 2.2: Position updating in GWO [1]

$$\vec{D}_\alpha = |\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}|, \tag{2.5}$$

$$\vec{D}_\beta = |\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}|, \tag{2.6}$$

$$\vec{D}_\delta = |\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}|, \tag{2.7}$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot (\vec{D}_\alpha), \tag{2.8}$$

$$\vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot (\vec{D}_\beta), \tag{2.9}$$

$$\vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \cdot (\vec{D}_\delta), \tag{2.10}$$

$$\vec{X}(t+1) = (\vec{X}_1 + \vec{X}_2 + \vec{X}_3)/3. \tag{2.11}$$

$\vec{X}_\alpha$ represents the current position of α, $\vec{X}_\beta$ represents the current position of β, and $\vec{X}_\delta$ represents the current position of δ. $\vec{C}_1$, $\vec{C}_2$, $\vec{C}_3$ represent a random vector, $\vec{X}$ represents the current wolf position vector. Equations (8), (9), (10) define the step and direction of wolf (ω) advancing toward α, β, δ, respectively. Equation (11) defines the final position of the wolf (ω). $\vec{A}$ and $\vec{C}$ these two vectors are random vector and adaptive vector.

Chapter 3

A Novel Intelligent Method for Predicting Vehicle State in Internet of Vehicle

In the fields of advanced driver assistance systems (ADAS) and Internet of Vehicles (IoV), predicting the vehicle state is essential, including the ego vehicle's position, velocity and acceleration. In ADAS, an early position prediction helps to avoid traffic accidents. In IoV, the vehicle state prediction is essential for the required calculation of the expected reliable communication time between two vehicles. Many approaches have emerged to perform this vehicle state prediction. However, such approaches consider limited information of the ego vehicle and its surroundings, and they may not be very effective in practice because the real situation is highly complex and complicated. Moreover, some of the approaches often lead to a delayed prediction time due to collecting and calculating the substantial history information. By assuming that the driver is a robot driver, which eliminates distinct driving behaviors of different persons when facing the same situation, this paper creates a decision tree as a new quick and reliable method adapted to all road segments, and it proposes a new method to perform the vehicle state prediction based on this decision tree.

3.1 Introduction

Advanced driver assistance systems (ADAS) installed in vehicles use sensing and computing technologies to assist drivers in avoiding traffic accidents. Predicting the positions of surrounding vehicles is a crucial problem, and it facilitates the early detection of potential collisions. In Internet of Vehicles (IoV), one of the most important foundations for the network connectivity is the vehicle state, including its position, velocity and acceleration. This importance is because the vehicle state is a deterministic characteristic for communicating among vehicles and infrastructures. Therefore, a common requirement is to calculate the expected reliable communication time quickly in various road segments when two vehicles are to communicate with each other. The vehicle state can influence the network topology of IoV where the location between two vehicles determines the communication range and their velocities and accelerations affect the stability of network topology [87], [88], [89]. The routing protocol based on location is particularly important due to its adaptability for frequently changing IoV [90]. Alsaqour *et al.* found that the inaccurate location obviously decreased the efficiency of routing protocol [91]. Thus, a neighbor wireless link break prediction was proposed to predict the neighbor node's location so as to detect the ineffective node by using their velocities and accelerations [92]. However, this method is just suitable for short-time and short-distance prediction because the accelerations of vehicles may significantly change according to changing environments. Consequently, the vehicle state prediction is necessary and essential for IoV.

A decision tree is an effective method for evaluating the behaviors of vehicle drivers, some studies include a decision tree to predict or monitor vehicle drivers [93], [94].

Ahmed presented a method to predict the vehicle state, in which a decision tree was used to determine whether the vehicle changed its lane and to obtain the lane after the vehicle changed lanes [95]. Kedowide *et al.* used a decision tree to monitor the vehicle driver and log the driving activities, such as to evaluate whether the driver was performing the blind spot check, integrated with the behaviors of the driver [96].

The aforementioned methods are primarily used in advanced driver assistance systems (ADAS) to avoid collisions on the planned trajectories of vehicles and considering limited information about the ego vehicle and its surroundings. Moreover, a delayed prediction time will arise when history data need to be collected. Although some studies have employed a decision tree to quickly make judgments, the decision tree does not use much road and environment information and does not perform predictions across all road segments.

This paper proposes a new approach based on a decision tree that considers more information about the ego vehicle, its surroundings and driver behaviors in varieties of road segments and without a delayed prediction time because no history data are collected as in some previous methods.

This approach saves time and reduces the precious prediction time. According to information of the ego vehicle, roads, traffic lights, other surrounding vehicles and so forth, our approach pre-judges the driving behaviors of the ego vehicle, and then a decision tree is adapted to all road segments. Thus, the state of the ego vehicle, including its position, velocity and acceleration, can be predicted based on a previously created decision tree. Such a decision tree with considerable useful information including more road surrounding cases helps to predict the vehicle state more accurately in

some complex and complicated environments and without a delayed prediction time. The decision tree has advantages such as quick situation judgment and easy extension to more complicated problems with more determination conditions to be adapted to all road segments.

The contributions of this paper can be summarized as following: (1) This paper defines three varieties of road segments: section, intersection and transition. Based on the definitions in the System Representation section, this work extracts different behaviors in distinct road segments. These defined behaviors are introduced in the Driving Behavior Modeling section. (2) To predict the vehicle state from the behavior, we use a decision tree in all road segments, which is illustrated in the State Prediction section. The decision tree includes the pre-defined road segment situations and has advantages such as fast situation judgment and easy extension to more complicated problems with more determination conditions to be adapted to more road segments. We discuss the state prediction by taking advantage of the decision tree, which allows our work to predict the vehicle state through the decision tree.

The remainder of this paper is organized as follows. Section II gives the past works in vehicle state. Section III presents an overview of the system. Section IV delineates several models of driving behaviors. Our prediction approach is described in Section V. The numerical results are presented in Section VI. Finally, the conclusions of this paper are drawn in Section VII.

3.2 Related Works

Researches about vehicle state can be classified as three parts, i.e., environments, maneuvers and trajectories [97]. Environments are components of conducting vehicle behaviors. Various approaches have been done to discuss vehicle behaviors in different driving environments. In [98], a detection-by-tracking method was used to detect vehicles in a spatiotemporal environment. In [99], intersection and nonintersection driving were distinguished by histograms of scene flow vectors. In [100], a dynamic driving environment was established for detecting the vehicle motion. Maneuvers such as overtaking [101], turning [102] and changing lanes [103] are investigated to analyze vehicle motion on the path. Overtaking behavior is implemented generally by using some devices to detect vehicles in front of the ego vehicle [101], [104], [105]. When overtaking conditions are satisfied in search space, vehicles will realize an overtaking maneuver. Turning behavior is another usual maneuver for vehicles. Detecting the yaw rate can judge the vehicle turning behavior [106]. Adopting a clustering of 3-D points to analyze vehicle's shape can also handle a turning behavior [107]. In [103], changing lanes was achieved by establishing a dynamic Bayesian network based on practical data. Trajectories composed of a set of sequences of positions and velocities with a time window are used to extract vehicle behaviors in the past few years. In [108], A Gaussian mixture model was utilized to predict long-term trajectories of vehicles. On the highway, trajectories were constructed using a stereo vision and clustering method [109].

Besides the study of practical vehicle state, many approaches have emerged to obtain a credible vehicle state prediction. Hermes *et al.* predicted the position of a

vehicle after several seconds using the history information of the vehicles [110]. Hermes *et al.* extracted a large number of vehicle trajectories to perform data training based on trajectory classification technology, in which trajectories were classified into several behaviors, such as left-handed rotation, right-handed rotation and so on, and then they classified the existing trajectories [111]. In addition to objectivities, some researches added drivers' subjective purposes such as left turn, right turn and changing lanes into the prediction models [112], [113]. A prediction technology for a motorcade formed by several vehicles was proposed by Pandita *et al.*, and in their approach, how a car follows was simulated using the smart driver model [114]. Additionally, a technology that combined the motion model and maneuver recognition was validated, in which probabilistic finite-state machines, fuzzy logics and driving context recognitions were involved to predict a vehicle trajectory [115], [116], [117]. Petrich *et al.* used additional information from a digital map to enable a stochastic filter to select a representative set of reasonable trajectories [118]. Kumar *et al.* predicted the lane change intention online using a support vector machine and Bayesian filtering [119]. Yao *et al.* learned a simplified trajectory set using a collection of lane change trajectories from real driving data [120]. By introducing essential maneuver recognition, Houenou *et al.* predicted the vehicle trajectory using the constant yaw rate and acceleration motion model [117], which was widely and importantly used in [121], [122] and [123]. These prediction technologies need numerous history information of vehicles, meanwhile, the impact of lane and traffic light on trajectories of vehicles is ignored. The trajectories of vehicles are restricted by lanes, however, a digital map based on routing protocol can offer information of lanes to improve

routing efficiency of IoV. Vehicle state prediction can also use the digital map and traffic light to enhance the accuracy of location prediction of vehicles [124], [118].

3.3 System Overview

3.3.1 Motivation

Vehicle state prediction is suitable for IoV in city scenario, integrating the digital map, traffic light and surrounding vehicles. It models the driver's behavior in different traffic environments. In this paper, according to the driver's behavior, a decision tree is established to describe vehicle state in diverse conditions. Vehicle state prediction is an important part of connectivity model in IoV, which is proper for predicting the positions of vehicles and dynamic changes of links. Taking advantage of information IoV provides, vehicle state can be predicted to further guide the driver to adopt several operations in order to implement a better trajectory of vehicle and save time. Nevertheless, the primary purpose of vehicle state prediction is not to find an optimal route but to predict the vehicle state in next seconds. The vehicle state contains the position, velocity and acceleration of vehicle. The change of vehicle state can influence the topology of IoV. For example, position can determine whether two vehicles are accessible to communicate with each other, whereas velocity and acceleration influence the stability of network topology. These factors could finally affect the survival time of links among vehicles. Hence, vehicle state prediction is mainly to calculate the survival time of links so as to guarantee to achieve a better communication among several vehicles and maintain a steady structure of IoV. Meanwhile, it also offers an effective method to construct a reasonable route.

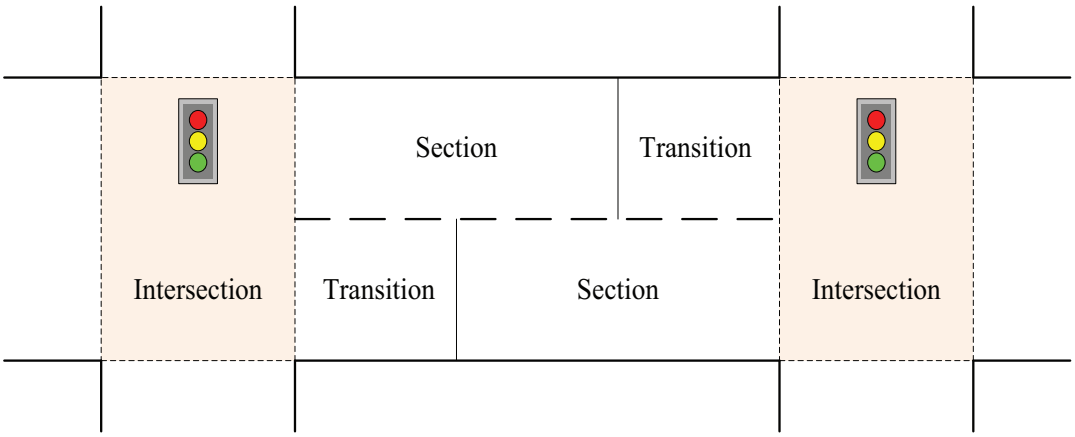

Figure 3.1: Road components

3.3.2 System Architecture

The vehicle state prediction proposed in this paper is designed specifically to be used in Internet of Vehicles (IoV). It is assumed that the state prediction is hosted by the server that maintains the states of vehicles on the Internet. This assumption is often considered to be a reasonable assumption. Each vehicle manages its state prediction via a virtual object. Nowadays, the virtual object plays an important role in Internet of Things to implement its virtualness and service [125]. For IoV, virtual objects implement the communication among vehicles and provide a practical application for managing vehicles. Position-based and map-based routing protocols in the previous literature are widely accepted routing protocols in IoV based on position and path. Cheng *et al.* [83] classify notable routing protocols into routing categories for performing routing. Both position-based and map-based routing protocols require vehicles to send their state information to the server, which is generally distributed, when a source vehicle needs to communicate with other vehicles periodically. The server destination node first queries the state information of the destination from

the server and then sends data toward the vehicle at the position. The position of the destination will often change during data forwarding; thus, if the position of the destination could be predicted, it would improve the routing performance. Moreover, by predicting vehicle states in a forward routing path, the server has the ability to calculate the expected reliable communication time between two vehicles and then calculate the connectivity of the path, which helps to select a stable path from multiple paths. When a source queries for the position of a destination, the server could send the predicted state and the optimal forward routing to the source, which will also improve the routing performance.

3.3.3 System Representation

The following describes the vehicle information that will be used in this work.

- The position of a certain vehicle at a certain time can be represented using a two-dimensional column vector:

$$\mathbf{p}(t) = (x(t), y(t))^T; \tag{3.1}$$

- The velocity of a certain vehicle at a certain time can be represented as follows:

$$\mathbf{v}(t) = (v_x(t), v_y(t))^T; \tag{3.2}$$

- The acceleration of a certain vehicle at a certain time can be expressed as follows:

$$\mathbf{a}(t) = (a_x(t), a_y(t))^T; \tag{3.3}$$

Table 3.1: Variable Summary

Variable	Explanation
$\mathbf{p}(t)$	the position of a vehicle at time t
$\mathbf{v}(t)$	the velocity of a vehicle at time t
$\mathbf{a}(t)$	the acceleration of a vehicle at time t
state(t)	the vehicle state of a vehicle, including its position, velocity and acceleration
intersec	an intersection, which is defined by a point
sec	a section, which is defined by two **intersec**s
lane	a lane, which is defined by **intersec** and its lane number
trans	a transition, whose definition is similar to *sec*

- The length of a certain vehicle, l. Specifically, the length of the ego vehicle is l_α;
- The number of vehicles in front of a certain vehicle at a certain time, $n\left(t\right)$;
- α refers to the ego vehicle, and $\alpha - k$ refers to the kth vehicle that is in front of the ego vehicle. For example, the directly previous vehicle is $\alpha - 1$;
- $d_{\alpha-k}$ refers to the distance between the ego vehicle and the kth vehicle in front of the ego vehicle. For example, the distance between the ego vehicle and the directly previous vehicle $\alpha - 1$ is $d_{\alpha-1}$.

Hence, the vehicle state in this work is defined as a triple:

$$\mathbf{state}\left(t\right) = \left\langle \mathbf{p}\left(t\right), \mathbf{v}\left(t\right), \mathbf{a}\left(t\right) \right\rangle, \tag{3.4}$$

where $\mathbf{p}\left(t\right)$, $\mathbf{v}\left(t\right)$ and $\mathbf{a}\left(t\right)$ are all mentioned above.

Additionally, the road information and vehicle surroundings should also be extracted and represented. Before extracting and representing the road information and vehicle surroundings, this work introduces a new concept of a transition between a section and an intersection. The road is divided into three segments, as shown in Fig. 3.1. All predictions in the three segments are integrated into one decision tree.

A transition is a special part of a section with information of the intersection that needs to be considered. In other words, when a vehicle is at a transition, the driver faces the intersection and is able to obtain information such as traffic lights and so forth.

- A certain intersection is defined as a two-dimensional point:

$$\mathbf{intersec} = (x_0, y_0)\,, \tag{3.5}$$

  where (x_0, y_0) is its position. On an intersection, we just consider three behaviors of each vehicle, i.e., left turn, right turn and pass through. Therefore, the intersection is expressed by a point.

- A certain section between two consecutive intersections $\mathbf{intersec}_1$ and $\mathbf{intersec}_2$ that are the ends of the certain section is defined as follows:

$$\mathbf{sec} = \langle \mathbf{intersec}_1, \mathbf{intersec}_2, 0 \rangle\,, \tag{3.6}$$

  where the direction of the vehicle is from intersection $\mathbf{intersec}_1$ to intersection $\mathbf{intersec}_2$.

- Consequently, a certain lane is

$$\mathbf{lane} = \langle \mathbf{sec}, n \rangle\,, \tag{3.7}$$

  where **sec** is the section to which the certain lane belongs and n is the number of the lane. In this work, the width of every lane is the same, and it is a

known constant. When a vehicle faces an intersection, the driver can see three directions. Additionally, this work defines three directions: $\mathbf{lane}_N$ is the direction of the lane in which the driver faces straight forward, $\mathbf{lane}_W$ is the direction of the lane that the driver turns right into, and $\mathbf{lane}_E$ is the direction of the lane that the driver turns left into.

- A certain transition between a certain section and a certain intersection that is an end of the certain section is

$$\mathbf{trans} = \langle \mathbf{intersec}_1, \mathbf{intersec}_2, 1 \rangle , \tag{3.8}$$

where **trans** is very similar to *sec* because a transition is a special part of a section, and when the vehicle is in the transition, the driver is facing intersection $\mathbf{intersec}_2$ and sees the traffic lights in the intersection. The purpose of 0 and 1 in **sec** and **trans** is to distinguish both mathematical definition.

Table 3.1 presents a summary of the aforementioned variables, including the definitions of the position, velocity, acceleration, and vehicle state of a vehicle and of several elementary road environments, such as an intersection, a section, a lane, and a transition.

Note that in this work, the number of lanes is based on zero, and the lane number starts from the central line of the section to which the lane belongs.

3.4 Driving Behavior Modeling

In this work, the driving behaviors of a vehicle are considered as the mean motions of the vehicle, such as some sudden changes including accelerating, decelerating, changing lanes and turning at an intersection. These driving behaviors lead to discontinuous acceleration, which causes the acceleration, velocity and position of the vehicle to be difficult to predict using their history states. The early detection of sudden changes is necessary for predicting the vehicle state. Driving behaviors can be defined as elements in a set, and each behavior is an element of the set. To create the decision tree in all road segments in this work, it is necessary to model the driving behaviors of a vehicle. The driving behavior is divided into three cases: section prediction, intersection prediction and transition prediction. At sections, vehicles accelerate or decelerate, which is caused by the influence of the front vehicles. Additionally, vehicles may change lanes to leave an upcoming jam or to avoid a slow vehicle that is directly in front. Only when adjacent lanes have spacing can lane changes occur. A transition, with some specific characteristics, is a certain area between a section and its intersection. Vehicles at a transition are forbidden from changing lanes, and their behaviors are mainly dependent on the traffic lights. At intersections, vehicles may turn left or right or pass through, depending on the out direction of the lane that the vehicle is in and on the traffic light.

According to the aforementioned road in various situations, this paper classifies driving behaviors into three models: section behaviors, intersection behaviors and transition behaviors. Section behaviors occur in the section, which are relatively simple without considerations of orientation changing. Considerations of intersection

behaviors include changing direction. Transition behaviors are relatively complicated. The transition situation is between section and intersection, and it contains possibilities of section's and intersection's behaviors; thus, it is difficult to predict the coming driving behavior due to the various possibilities.

3.4.1 Section Behaviors

Section behaviors are always when the vehicle is far away from the front intersection and the traffic light is out of the range of the driver.

1) Jam Leaving Intent: When a driver realizes that a jam has occurred in the front of his current lane, he will attempt to enter adjacent lanes to avoid the jam. Lane changing behavior is an important intent and has already been considered in the previous literature, such as by Ahmed [95]. In this work, α refers to the ego vehicle, and $\alpha - m$ refers to the mth vehicle in front of the ego vehicle, for example, the vehicle directly in front of the ego vehicle is $\alpha - 1$. Here, r_α is the range of the driver in the ego vehicle α. The driver could see τ_α vehicles $\alpha - 1, \alpha - 2, \ldots, \alpha - \tau_\alpha$ in the driver's range r_α, but the vehicle $\alpha - \tau_\alpha - 1$ is out of the driver's range. Thus, τ could be represented mathematically by

$$\tau_\alpha = \max_i d_{\alpha - i} < r_\alpha, \tag{3.9}$$

where $d_{\alpha-i}$ is the distance between the vehicle α and the ith front vehicle $\alpha - i$. The jam density ρ_α, which defines an indicator to quantify the congestion level, is as follows:

$$\rho_\alpha = \frac{\tau_\alpha}{r_\alpha}. \tag{3.10}$$

The driver will have a jam leaving intent if the driver cannot tolerate such a jam that $\rho_\alpha > \rho_\alpha^*$, where ρ_α^* is a tolerance threshold for the driver. In this paper, the length of vehicle is 4.3-4.7 meters, we set one vehicle within 5 meters as a tolerance threshold. Thus, this paper uses $\rho_\alpha^* = 0.2$ and $r_\alpha = 400m$ in our later numerical experiments. Then, the driver will change lanes if the condition for changing lanes is satisfied. The driver always prefers to change to the right lane, and when the condition for changing to the right lane is not satisfied, the driver considers changing to the left lane.

2) Overtaking Intent: For this intent, this work considers two aspects: the sizes of and the velocities between the front vehicle and the ego vehicle.

When the front vehicle, such as a truck, is considerably larger than the ego vehicle, the driver always tends to avoid following it. This work simply assumes that the width of every vehicle is the same; thus, this case is simply to compare the lengths and is presented by

$$l_{\alpha-1} > \lambda_{\text{length}} l_\alpha, \tag{3.11}$$

where $\lambda_{\text{length}} > 1$ is the tolerance threshold for the ratio of the length of the directly previous vehicle to the length of the ego vehicle.

In the other case, if the speed of the vehicle ahead is too slow, the driver often attempts to change lanes and overtake the slow vehicle. Mathematically,

$$v_{\alpha-1} < \lambda_{\text{velocity}} v_{\alpha}, \tag{3.12}$$

where $\lambda_{\text{velocity}} < 1$ is the tolerance threshold. This paper uses $\lambda_{\text{length}} = 1.5$ and $\lambda_{\text{velocity}} = 0.8$.

3) Following Intent: In general, the driver of the ego vehicle will follow the front vehicle. However, when the driver follows the front vehicle, the driver will also adapt the ego vehicle such that it will be more comfortable and safe. For example, when the ego vehicle is too close to the front vehicle, the driver tends to brake to avoid driving into it. Mathematically,

$$t_{\text{brake}} = \frac{d_{\alpha-1}}{v_{\alpha} - v_{\alpha-1}}. \tag{3.13}$$

Here, we select a safety braking time t_{safety} as a threshold for the ratio of the velocity of the directly previous vehicle to the velocity of the ego vehicle. When $t_{\text{brake}} \leq t_{\text{safety}}$ is satisfied, the driver will decelerate with an acceleration value. In this paper, $t_{\text{safety}} = 1.5s$ and $\delta = 2$ is a correlation coefficient.

$$a = -\delta \frac{v_{\alpha} - v_{\alpha-1}}{d_{\alpha-1}}. \tag{3.14}$$

4) Free Driving Intent: Otherwise, the state of the driver will be maintained. This case is called *free driving intent*. For free driving, this work will consider that if the velocity of a vehicle is less than the speed limit, then the driver tends to accelerate with a constant acceleration to reach the speed limit.

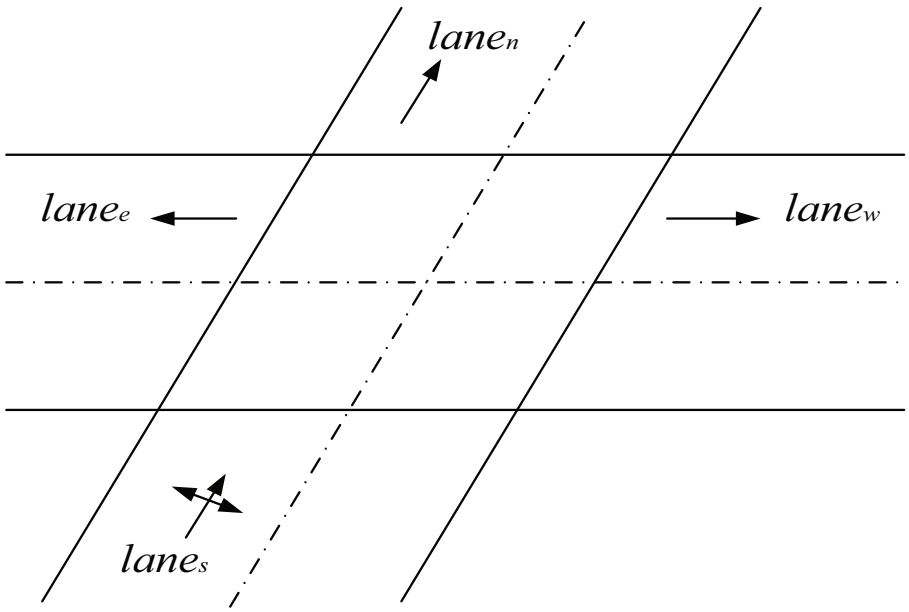

Figure 3.2: Intersection

3.4.2 Intersection Behaviors

Intersection behaviors are to predict the motions when the vehicle is close to or facing the front intersection. In such cases, the driver should consider the information of the front intersection, such as traffic light and so on. Behaviors at intersections are difficult to detect without information, including the out directions of the lane where the ego vehicle is located and the traffic lights. Existing studies always use history trajectories to recognize vehicle behavior using pattern classifications, fuzzy logics, probabilistic finite-state machines or other technologies [115]. However, these technologies all require sufficiently long trajectories, which lead to delayed time, and these technologies have considerable computational requirements, which makes them unsuitable for performing recognition of behaviors at a server with a massive number of vehicles. According to the out direction of the driving lane and traffic light, it is simple and accurate to achieve early detection of whether the vehicle is going to turn

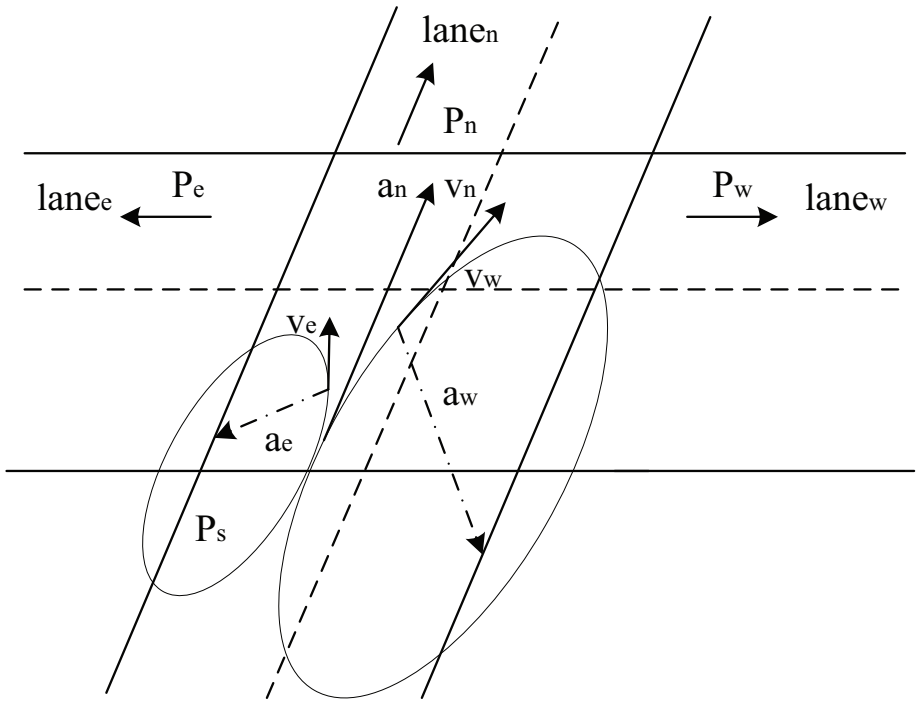

Figure 3.3: Behaviors at intersection

left, right or pass through. However, there is a case in which the motion cannot be detected only using the out direction and traffic lights. This situation arises because the traffic light may occasionally allow the three directions simultaneously. In this case, the motion cannot be determined only by the directions of the lane and the traffic light. As shown in Fig. 3.2, the $\mathbf{lane}_s$ has three out directions: S/E, S/W and S/N. Here, S is marked as the directing lane toward the south, E is marked as the left lanes of the intersection toward the east, W is marked as the right lanes of the intersection toward the west, and N is marked as the lanes across to the north. Hence, S/E means that the driver turns left from the current lane to the east lanes, and S/W and S/N have similar meanings. The motion may be to turn left or right or pass through with the trajectories $\overset{\frown}{P_sP_e}$, $\overset{\frown}{P_sP_w}$, and $\overset{\frown}{P_sP_n}$, as shown in Fig. 3.3. Here, P_e, P_s, P_w, and P_n are four positions standing for positions to the east, south, west,

and north, respectively, of the certain transition. For simplicity, this work considers the curve trajectory of the motion from the ego vehicle, which is similar to 1/4 part of an ellipse, as illustrated in Fig. 3, due to two accelerations changing, in which one's direction is the original direction and the other's direction is the terminal one. The details of the calculation of the two accelerations and corresponding velocity and position will be discussed in Section IV, State Prediction. When a vehicle is arriving from the south, it will have three probabilities: to turn left (go $\mathbf{trans}_e$ in Fig. 3), to turn right (go $\mathbf{trans}_w$ in Fig. 3) and to go straight (go $\mathbf{trans}_n$ in Fig. 3). We will discuss these three cases in the following. If

$$\left| \frac{\mathbf{a}^T \cdot \mathbf{v}}{|\mathbf{a}|\,|\mathbf{v}|} \right| > 1 - \epsilon, \tag{3.15}$$

the motion is to pass through. If

$$\mathbf{a}^T \cdot \mathbf{lane}_E > \epsilon \text{ or } \mathbf{a}^T \cdot \mathbf{lane}_W < -\epsilon, \tag{3.16}$$

the motion is to turn left. If

$$\mathbf{a}^T \cdot \mathbf{lane}_W > \epsilon \text{ or } \mathbf{a}^T \cdot \mathbf{lane}_E < -\epsilon, \tag{3.17}$$

the motion is to turn right. $\mathbf{lane}_E$ and $\mathbf{lane}_W$ are column vectors. The aforementioned ϵ is a positive value close to zero, indicating that it is sufficiently small. In this paper, $\epsilon = 0.01$.

3.4.3 Transition Behavior

When a vehicle is in a transition, the driver can see the traffic light. Different traffic lights can lead to distinct behaviors of the vehicle. Here, we consider that when a driver faces a red or yellow traffic light, the driver will give the vehicle a constant acceleration $\mathbf{a}_\delta$. We also provide a vector $\mathbf{e}$ to represent the traffic light information:

$$\mathbf{e} = \begin{cases} (1,0,0)^T & , \text{ if the traffic light is red} \\ (0,1,0)^T & , \text{ if the traffic light is yellow} \\ (0,0,1)^T & , \text{ if the traffic light is green} \end{cases} \cdot$$

Thus, the constant acceleration vector can be represented as

$$\delta_a = (\mathbf{a}_\delta, \mathbf{a}_\delta, 0)^T , \tag{3.18}$$

where each dimension indicates the acceleration of vehicle in corresponding traffic light.

Hence, when the driver faces the traffic light, the acceleration that the driver will provide is

$$\mathbf{a}_\Delta = \delta_a^T \cdot \mathbf{e}. \tag{3.19}$$

3.5 State Prediction

Now, driving behaviors are modeled, and a decision tree can be created based on the surroundings and driving behaviors in varieties of road segments. Our decision tree

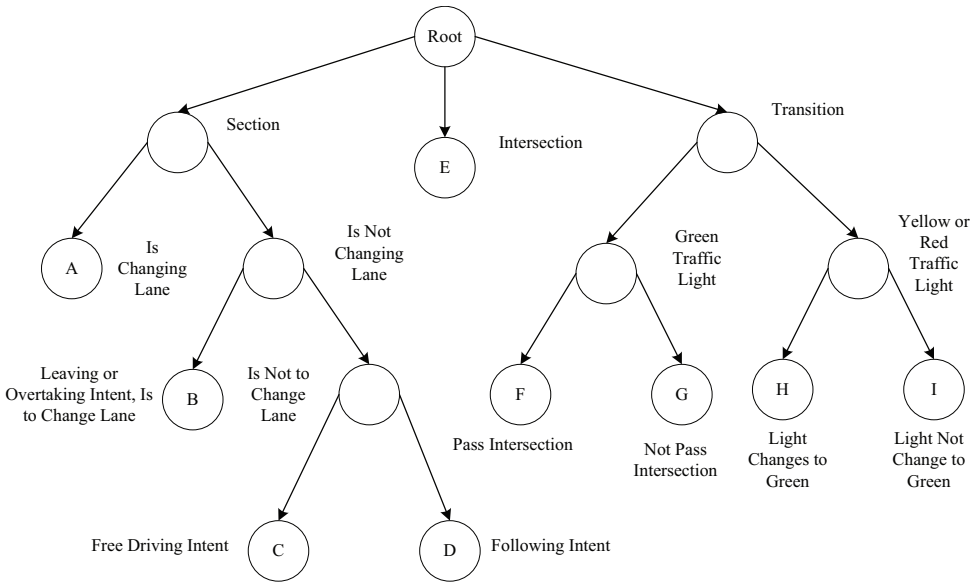

Figure 3.4: Decision tree based on driving behaviors

is illustrated in Fig. 3.4. The decision tree in Fig. 4 represents the aforementioned situations in various road segments and their judgment conditions, and it will help provide quick and easy determination and extension. First, it will be considered that the ego vehicle is in a section, an intersection or a transition, and these cases will be discussed individually. Note that in this work, when the ego vehicle is in the intersection, it means that the ego vehicle has passed the beginning line and will no longer consider traffic lights.

3.5.1 Prediction in Section

When the ego vehicle is in a section, it is considered whether the vehicle is changing lanes. This is because if the vehicle is changing lanes, its velocity and acceleration are not in the same direction, which will lead to a different trajectory. If the ego vehicle is not changing lanes, it is considered whether the vehicle will change lanes based on the aforementioned *jam leaving intent* and *overtaking intent*. These two intents are

very common in reality. If the ego vehicle does not choose to change lanes, then there are two intents for the driver of the ego vehicle: *free driving intent* and *following intent*. To summarize, the prediction in a section could have four cases, **A**, **B**, **C** and **D**, as indicated in Fig. 4.

1) A: When a vehicle is in a section **sec** and it is changing lanes, it has a velocity $\mathbf{v}_{\perp}(t)$ and an acceleration $\mathbf{a}_{\perp}(t)$ whose directions are both perpendicular to **sec**. During the lane change, the vehicle is supposed to have an acceleration $\mathbf{a}_{\perp}$ to first accelerate and then $-\mathbf{a}_{\perp}$ to decelerate, where $|\mathbf{a}_{\perp}| > 0$ could be calculated from the data set. Therefore,

$$\mathbf{a}_{\perp}(t) = \begin{cases} \mathbf{a}_{\perp} & , \ t < t_0 \\ -\mathbf{a}_{\perp} & , \ t > t_0 \end{cases},$$

where t_0 is the time point between accelerating and decelerating and t is during the lane change. Moreover, $\mathbf{v}(t) = \mathbf{v}_{\perp}$ could be calculated from the data set, and the distance of changing the lane equals the lane width, which is also a known constant mentioned in the previous discussion. Hence, the $\mathbf{p}(t)$ during the lane change and the current time point could both be calculated. Then, the position after t_{Δ} can be determined as follows:

$$\mathbf{p}_{\Delta} = \begin{cases} \mathbf{p}(t + t_{\Delta}) & , \text{ if still changing} \\ \mathbf{p}(t + t') + \int_{t+t'}^{t+t_{\Delta}} \mathbf{v}(t)\, dt & , \text{ if changing is done} \end{cases},$$

where $\mathbf{p}_{\Delta}$ is the position at time point $t + t_{\Delta}$ and $t + t'$ is the time point when the vehicle completes the lane change.

2) B: This case is very similar to Case **A**, but the time point t is not during the lane change but rather when starting to change lanes. Moreover, in Case **A**, the acceleration could be calculated from the data set, whereas in this case, the acceleration $\mathbf{a}_{\perp}(t)$ cannot be calculated from the data set. In this case, this work assumes that the time for changing lanes is a known constant; then, the $\mathbf{a}_{\perp}(t)$ could be calculated according to the distance of changing the lane, which equals the lane width. Thus, the position after t_{Δ} could be predicted by the method in Case **A**.

3) C: When a vehicle faces this case, the driver will choose *free driving intent*, which was previously mentioned.

4) D: When a vehicle faces this case, the driver will choose *following intent*, which was previously mentioned.

3.5.2 Prediction in Intersection

1) E: When a vehicle is in an intersection **intersec**, the driver could have three options: to drive straight forward, to turn left and to turn right. The velocity $\mathbf{v}_t$ at the current time point can be calculated from the data set, and it will be compared with $\mathbf{lane}_N$, $\mathbf{lane}_W$ and $\mathbf{lane}_E$ to determine which direction the vehicle will go. Mathematically, the direction that the vehicle will go is

$$\mathbf{direction} = \begin{cases} \mathbf{lane}_W & , \ \mathbf{v}(t) \cdot \mathbf{lane}_W > \epsilon \\ \mathbf{lane}_E & , \ \mathbf{v}(t) \cdot \mathbf{lane}_E > \epsilon \\ \mathbf{lane}_N & , \ \text{otherwise} \end{cases},$$

where $\epsilon > 0$ is a positive value that is sufficiently small, as previously mentioned. $\epsilon = 0.01$ in this paper. If **direction** = $\mathbf{lane}_N$, then the intersection **intersec** could be considered as a section. If **direction** = $\mathbf{lane}_W$ or **direction** = $\mathbf{lane}_E$, then the vehicle has two accelerations that have the straight forward direction and the direction the same as $\mathbf{lane}_W$ or $\mathbf{lane}_E$, respectively. This work denotes the first mentioned acceleration as $\mathbf{a}_o(t)$ and the second mentioned acceleration as $\mathbf{a}_l(t)$. This work assumes that in the intersection **intersec**, $\mathbf{a}_o(t)$ is linearly increasing from zero and $\mathbf{a}_l(t)$ is linearly decreasing to zero. That is, $|\mathbf{a}_o(t)| + |\mathbf{a}_l(t)|$ is a constant during turning. At some certain time point t_0, $\mathbf{a}_o(t_0)$ and $\mathbf{a}_l(t_0)$ could be calculated from the data set; thus, we let

$$a_\Sigma = |\mathbf{a}_o(t_0)| + |\mathbf{a}_l(t_0)|. \tag{3.20}$$

From the data set of the map, the distance between the current time point and the time point when the vehicle completes turning can be calculated. Hence, the time t_s remaining for turning can be obtained. Therefore,

$$\mathbf{a}_o(t_0 + t) = \left(|\mathbf{a}_o(t_0)| - \frac{|\mathbf{a}_o(t_0)|}{t_s}t\right)\mathbf{e}_o, t < t_s, \tag{3.21}$$

$$\mathbf{a}_l(t_0 + t) = \left(|\mathbf{a}_l(t_0)| + \frac{|a_\Sigma - \mathbf{a}_l(t_0)|}{t_s}t\right)\mathbf{e}_l, t < t_s, \tag{3.22}$$

where $\mathbf{e}_o$ is the direction of the original direction and $\mathbf{e}_l$ is the direction of **direction**. The position after t_Δ is

$$\mathbf{p}\left(t_0+t_\Delta\right)=\begin{cases}\mathbf{p}\left(t_0\right)+\int_0^{t_\Delta}\mathbf{v}_0+\mathbf{a}\left(t_0+t\right)dt & ,\ t<t_s\\ \mathbf{p}\left(t_0+t_s\right)+\left(t_\Delta-t_s\right)\mathbf{v}' & ,\ t>t_s\end{cases},$$

where $\mathbf{v}'$ is the velocity when the vehicle completes turning.

3.5.3 Prediction in Transition

1) F: When the driver faces a green traffic light and the vehicle could pass in time, the case could be in a section (when the requested time point is not sufficient to pass) or in a intersection (when the requested time point is sufficient to pass).

2) G: When the driver faces a green traffic light and the vehicle cannot pass in time, the driver will stop the vehicle. The vehicle knows if some vehicle is in front of it. If some vehicle is in front of it, the driver will have *following intent*. If no vehicle is in front of it, the vehicle will calculate the distance between the current position and the final stopped position, which is denoted as d_s. This work assumes that the vehicle will be stopped by a constant acceleration. The constant acceleration can be calculated as

$$\mathbf{a}=-\frac{\left|\mathbf{v}\left(t_0\right)\right|^2}{2d_s}\mathbf{e}, \tag{3.23}$$

where $\mathbf{e}$ is the direction of the vehicle. Thus, the velocity and position are

$$\mathbf{v}\left(t_0+t_\Delta\right)=\mathbf{v}\left(t_0\right)+\mathbf{a}\cdot t_\Delta, \tag{3.24}$$

$$\mathbf{p}\left(t_0 + t_\Delta\right) = \mathbf{p}\left(t_0\right) + \int_0^{t_\Delta} \mathbf{v}\left(t_0 + t\right) dt. \tag{3.25}$$

3) H: This case could be separated into two time intervals: before and after the traffic light turns green. Before the traffic light turns green, the driver would choose *following intent*, while after the traffic light turns red, the case would be Case **F**.

4) I: Because the vehicle will stop as Case **G**, irrespective of whether some vehicle is in front of it, the vehicle have the same intent as Case **G**.

3.5.4 Prediction Summary

We call the above proposed method as driver behavior decision tree (DBDT), which obtains the relatively accurate trajectories of vehicles in a long term according to the sudden changes such as acceleration, deceleration and turn. Moreover, to prevent the prediction from going too far, this work includes the constant yaw rate and acceleration (CYRA) [117] into our approach. CYRA is a physical kinematic-based prediction method. It assumes that within a very short term, the force on a vehicle remains unchanged and the vehicle would keep a constant accelerate vector, including its accelerate direction and value. Thus, CYRA model regards the acceleration and direction of vehicle as a constant to predict the vehicle state. Its constant acceleration $\mathbf{a}_t$ is formulated as follows:

$$\mathbf{a}_t = \mathbf{a}_0, \tag{3.26}$$

where $\mathbf{a}_0$ is a constant value. Next, its velocity and position are calculated as follows:

$$\mathbf{v}_t = \int_0^t \mathbf{a}_t dt, \tag{3.27}$$

$$\mathbf{p}_t = \int_0^t \mathbf{v}_t dt. \tag{3.28}$$

The linearity of its state equation achieves a transmission of state probability distribution. The next vehicle state could be predicted based on this kind of constant accelerate vector.

For a short term, the acceleration of vehicle can be considered as a constant, CYRA can effectually adapt to this situation according to its constant acceleration characteristics. Hence, CYRA can effectively handle the vehicle state prediction in a short term so as to obtain more accurate results. However, it could result in a great error for predicting the vehicle state in a long term because the acceleration of vehicle continually changes. On the contrary, DBDT can detect the sudden change of acceleration of vehicle to instantly adapt to current state so as to obtain better results and avoid a great error, suggesting it is more suitable for predicting the vehicle state in a long term. On the basis of both characteristics, this work finally adopts the following formula to evaluate their performances.

$$T_{\mathrm{DBDT}}(t) = f(t)\, T'_{\mathrm{DBDT}}(t) + (1 - f(t))\, T_{\mathrm{CYRA}}(t), \tag{3.29}$$

where $T'_{\mathrm{DBDT}}(t)$ is the result of our approach and $T_{\mathrm{CYRA}}(t)$ is the result of another approach. $f(t)$ is an increasing function, which means driving behavior recognition is more suitable for long-term prediction and CYRA is more accurate for short-term prediction. In this paper, $f(t) = \frac{1}{4}t$.

3.5.5 Time complexity

A time complexity comparison between DBDT and CYRA is discussed in this subsection. For DBDT, we set C to be the number of vehicles in the same lane. The time complexity regarding prediction in section, intersection and transition is calculated as follows:

1. Prediction in section

- Motion prediction

  1) Jam Leaving Intent: Scanning vehicles in front of itself in the same lane needs the time complexity $C_{m1} = O(c)$.

  2) Overtaking Intent: Considering the vehicle in front of itself requires $C_{m2} = O(1)$.

  3) Following Intent: Calculating the vehicle in front of itself needs $C_{m3} = O(1)$.

  4) Free Driving Intent: This situation takes $C_{m4} = O(1)$.

- Vehicle state prediction

  A) Computing the location data of the lane and state of itself costs $C_{s1} = O(1)$.

  B) Computing the location data of the lane and vehicle states in front of itself needs $C_{s2} = O(c)$.

  C) Computing the vehicle state in front of itself requires $C_{s3} = O(1)$.

  D) Computing the state of itself takes $C_{s4} = O(1)$.

Thus, the time complexity of prediction in section is

$$\begin{aligned} C_{sec} =& (C_{m1} + C_{m2} + C_{m3} + C_{m4}) + max(C_{s1}, C_{s2}, C_{s3}, C_{s4}) \\ =& O(c) \end{aligned} . \tag{3.30}$$

2. Prediction in intersection

- Motion prediction

  Predicting the vehicle motions by the traffic light data and the location data of intersection lanes costs the time complexity $C_m = O(1)$.

- Vehicle state prediction

  E) Computing the vehicle state of itself and intersection lanes data needs $C_s = O(1)$.

  Therefore, the time complexity of prediction in intersection is

$$C_{intersec} = C_m + C_s = O(1). \tag{3.31}$$

3. Prediction in transition

- Motion prediction

  When a vehicle is in a transition, its motion is predicted by the traffic light. This operation needs $C_m = O(1)$.

- Vehicle state prediction

  F) When the driver faces the green traffic light and the vehicle could pass in time, computing the state of itself needs $C_{s1} = O(1)$.

G) When the driver faces the green traffic light and the vehicle could not pass in time, computing the vehicle state and traffic light time requires $C_{s2} = O(1)$.

H) When the driver faces the red traffic and the traffic light turns to green before it passes the transition, computing the vehicle state and traffic light time requires $C_{s3} = O(1)$.

I) When the driver faces the red traffic and the traffic light keeps red before it passes the transition, computing the vehicle state and traffic light time costs $C_{s4} = O(1)$.

Thus, the time complexity of prediction in transition is

$$C_{trans} = C_m + max(C_{s1}, C_{s2}, C_{s3}, C_{s4}) = O(1). \tag{3.32}$$

The number of vehicles is set to be n for prediction. Consequently, the whole time complexity about DBDT is $C = max(C_{sec}, C_{intersec}, C_{trans}) = n * O(c) = O(n)$. For CYRA, each vehicle is predicted by the data of itself. Hence, its time complexity is $O(n)$ [117]. According to both time complexity, we can find that DBDT and CYRA have the same time complexity, suggesting they possess the same efficiency.

3.6 Results and Analysis

To test whether our work is valid, experiments are conducted in a real environment, which is based on the Lankershim Boulevard Dataset of the Next Generation Simulation (NGSIM) program [126]. The Lankershim Boulevard Dataset collects detailed vehicle trajectory data from Lankershim Boulevard in the Universal City neighbor-

Table 3.2: Data Set Parameters

Lankershim Boulevard Dataset	Parameters
Address	Lankershim Boulevard in Los Angeles
Time	8:28-8:45 am & 8:45-9:00 am on 2005.6.16
Road Length	490 m
Intersection Number	4
Sampling Time	1/10 s
Lane Number (Same direction)	1-6
Provides Traffic Light Data	Yes
8:28-8:45 am Data Amount	705294 Records
8:28-8:45 am Vehicle Number	1375
8:45-9:00 am Data Amount	902025 Records
8:45-9:00 am Vehicle Number	1601

hood of Los Angeles. It provides the map of an area of Lankershim Boulevard, including three to four lane segments and covering three signalized intersections. Moreover, the traffic light data and the precise vehicle position, velocity and acceleration in the periods of 8:30 a.m. and 8:45 a.m. on June 16, 2005, are available. The Lankershim Boulevard Dataset covers the driver behavior of lane changing on congested segments, overtaking and behavior at traffic lights, which fits the experimental requirements of this work. The details of the Lankershim Boulevard Dataset are listed in Table 3.2. This work creates a model for the provided map in the Lankershim Boulevard Dataset to extract location data of sections, intersections and transitions. Then, this work extracts traffic light information, and thus, it obtains all road information. By inputting trajectory information of vehicles, this work will compare our approach (DBDT for short) to CYRA [117], DT which is a variant of DBDT by setting $f(t) = 1$ in Eq. (29), and SDT [93, 94] which is a self-selection threshold decision algorithm based on decision tree in four cases: $t_\Delta = 1s$, $t_\Delta = 2s$, $t_\Delta = 3s$ and $t_\Delta = 4s$, respectively.

The results for the accuracy of position prediction are shown in Fig. 3.5, those for the accuracy of velocity prediction are shown in Fig. 3.6, and those for the accuracy

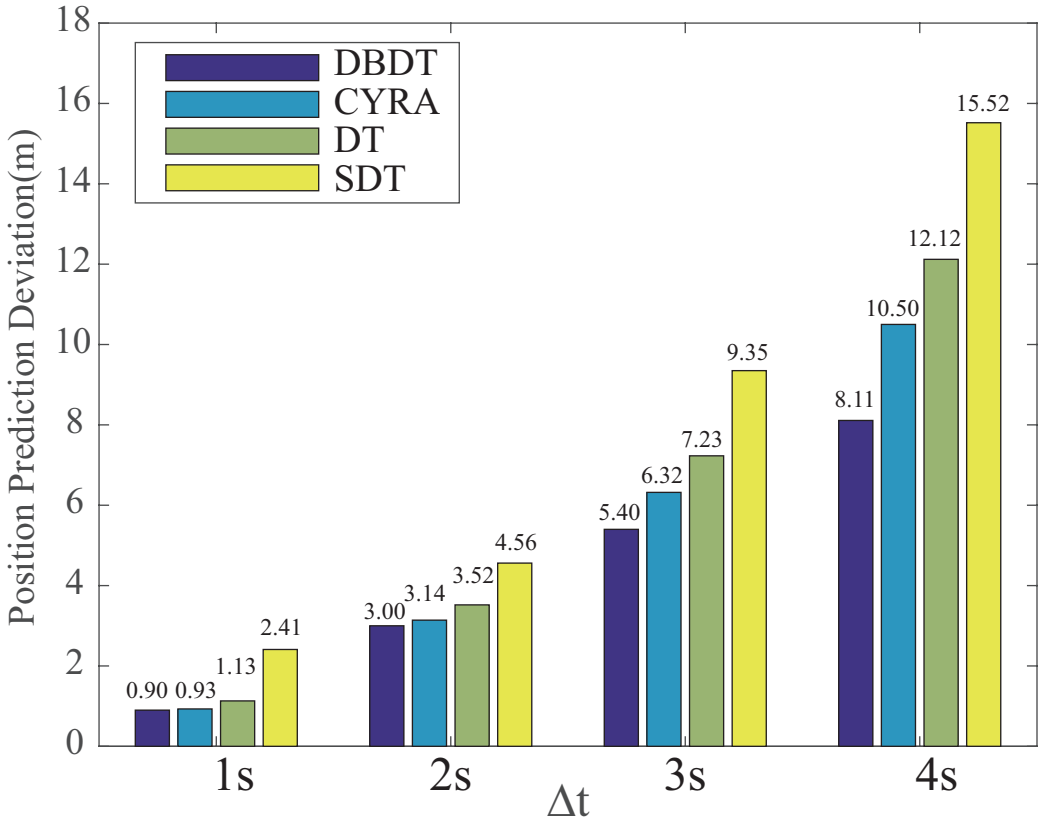

Figure 3.5: Accuracy of position prediction.

of acceleration prediction are shown in Fig. 3.7. The results show that although the state predicted by our approach is not very accurate at the beginning, the state is more accurate than that of CYRA as time passes. This is because our approach provides early detection of the driving behavior, which leads to changing the state at the very beginning of the prediction time point. Moreover, the vehicle state includes the ego vehicle's position, velocity and acceleration, for which the importances are decreasing in many fields. For example, to avoid traffic accidents, the vehicle position prediction is the most essential. Considering the discontinuous acceleration, the three vehicle state components, which are the position, the velocity and the acceleration, are becoming more difficult. Thus, it is expected that from the numerical results, the

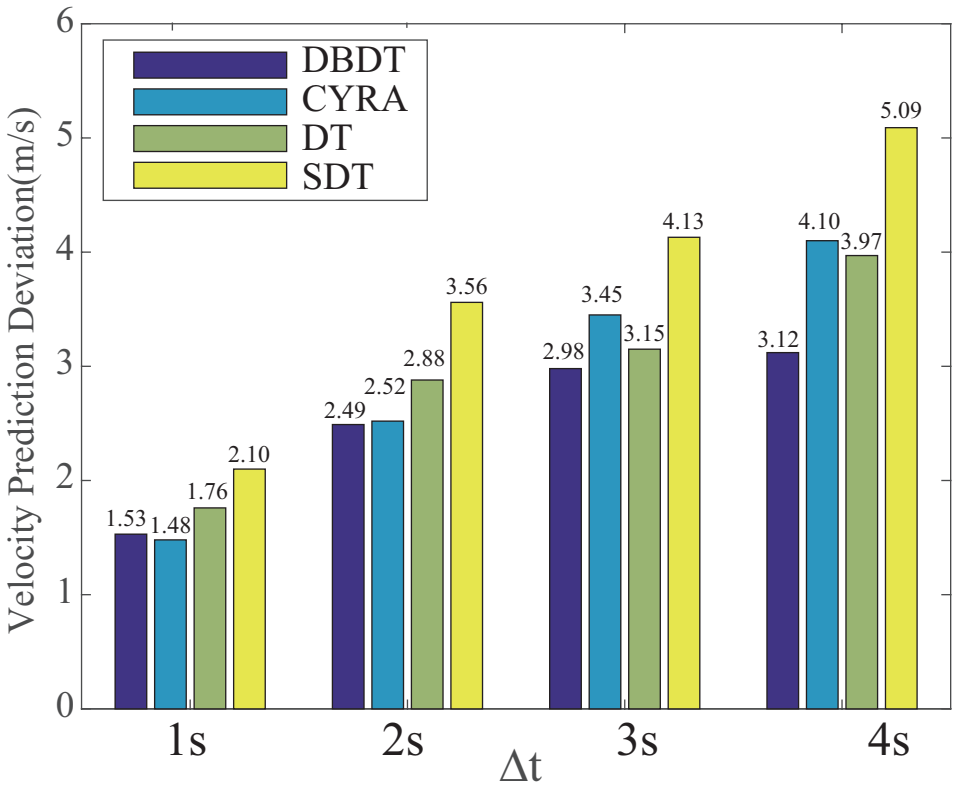

Figure 3.6: Accuracy of velocity prediction.

position prediction is the best, the velocity prediction is not good when $\Delta t = 1s$, and the acceleration prediction is not good when $\Delta t = 1s$ or $\Delta t = 2s$. As time passes, the numerical results become better. From the results, the difference value between the previous one second and the next one second becomes increasingly smaller. Thus, as time lasts past a certain range, the state prediction will be more accurate than that of CYRA.

Additionally, the results of DBDT are better than those of DT, suggesting that the key to the good performance of our proposal is the incorporation and extension of the decision tree and CYRA. In comparison with SDT which generally utilizes thresholds to determine the state selection in a decision tree, our proposal performs better with the aid of accurate modelling of the driving behaviors. Moreover, in light of the

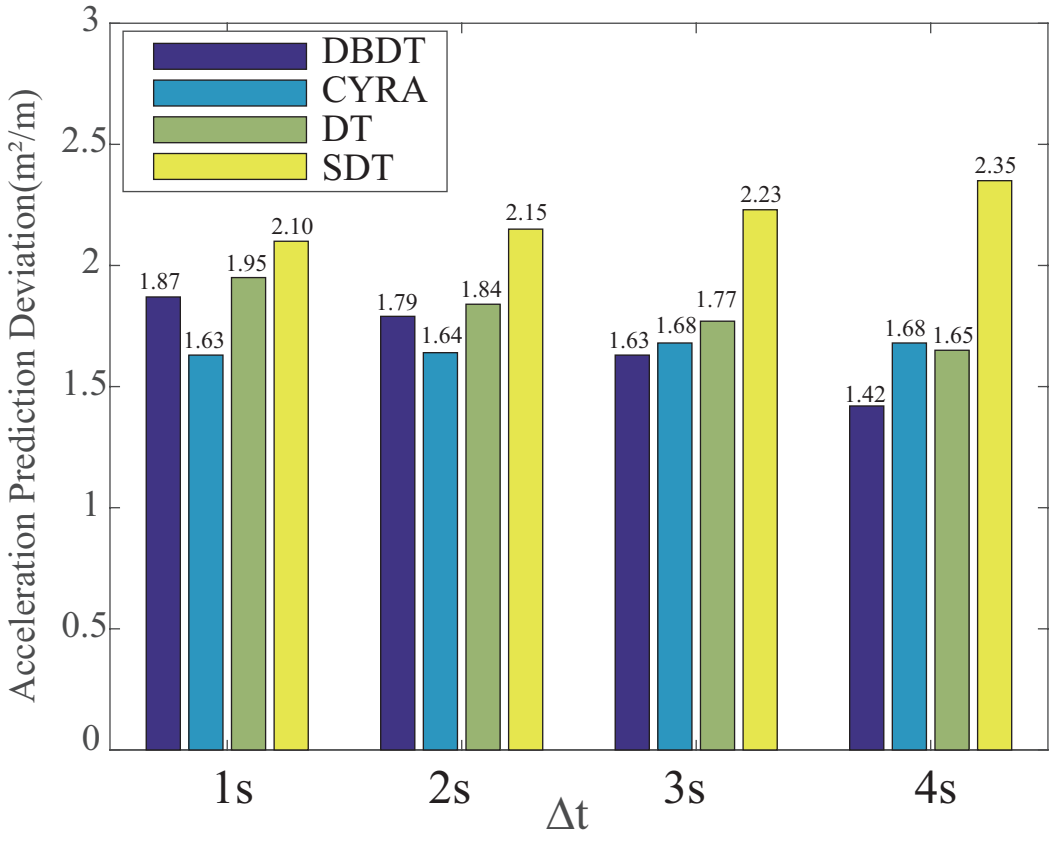

Figure 3.7: Accuracy of acceleration prediction.

results, we believe that more research on how to use driving behaviors of vehicle in the varieties of the all road segments to predict or monitor vehicle drivers by decision trees is warranted.

3.7 Conclusion

This paper highlights that the previous approaches for predicting the vehicle states using the substantial history information have a delayed prediction time. Some trajectory prediction methods based on lane changing recognition are proposed. Although a validation method for complicated environments such as multi-lanes and intersections is not currently available, this paper proposes a new method for the prediction

by using a decision tree in varieties of road segments generated by the driving behaviors. This decision tree helps to detect driving behaviors and predict the vehicle state in all road segments, including sections with multi-lanes, transition segments and intersections. The driving behavior recognition improves the accuracy of vehicle state prediction in long-term cases. Our approach shows advantages in the provided real environments.

Social Internet of Vehicles is an important and intelligent transport network [127]. It has more characteristics and more complicated circumstances. Thus, to predict this kind of IoV is more meaningful and challenging in the future work. Furthermore, the proposed technique might lead to the development of vehicle networking and intelligentialization [128], as well as to provide effective methods to solve vehicle routing problems in dynamic environments [69].

Chapter 4

Chaotic Grey Wolf Algorithm for Numerical Optimization Problems

Grey wolf optimization algorithm (GWO) is a recently proposed meta-heuristics and has shown promising performance in solving complex function optimization and engineering problems. To further enrich the search dynamics of GWO, the chaotic local search (CLS) mechanism is incorporated into GWO to enhance the search by taking the properties of ergodicity and randomness of chaotic maps. Twelve different kinds of chaotic maps are investigated to give some insights into the influence of CLS on GWO. Experimental results based on 29 widely used benchmark functions suggest that CLS indeed enables GWO to possess better performance in terms of solution accuracy, solution distribution, and convergence property. Summarized results also reveal that the performance of the resultant chaotic grey wolf optimization (CGWO) algorithm is effected not only by the characteristics of the embedded chaotic map, but also by the landscape of the solved problems.

4.1 Introduction

In the past two decades, various meta-heuristics have been proposed by taking inspirations from evolutionary theories, physical phenomenon, immunological mechanism, social behaviors, etc. Some famous ones can be listed as follows: genetic algorithm, evolution programming, differential evolution, ant colony optimization, scatter search, simulated annealing, particle swarm optimization [129]. These meta-heuristics can be categorized into classes according to: single-solution-based or population-based, model-based or instance-based, with or without memory. Regardless the differences among meta-heuristics, the unity procedure of their implementation involves four steps: (1) initialization of parameters and solutions and set iteration number to be zero; (2) generate candidate solution set according to generation rules and sampling distributions; (3) Update solution and parameter set; and (4) If termination conditions are satisfied, stop and exit; otherwise return to step (2). Following this general implementation process, meta-heuristics have achieved great success in solving various problems arisen from engineering, bio-informatics, complex networks, etc.

Although dozens of meta-heuristics have been proposed in the literature, it is still a great demand and challenging to design more new meta-heuristics. The famous No Free Lunch theorem [130] demonstrates that there is no meta-heuristics best suited for solving all optimization problems. New meta-heuristics can bring new inspirations, mechanisms, motivations and characteristics for the problem-solving research community. Grey wolf optimization algorithm (GWO) is such an answer for this request. By mimicking the social leadership and hunting behavior of grey wolves in nature, GWO performs the search in problem's landscape with a distinct character-

Table 4.1: Typical meta-heuristics and the corresponding chaotic meta-heuristics.

Algorithm	Reference	
	Original	Chaotic Version
Evolutionary Algorithm	[131]	[132]
Differential Evolution	[133]	[134, 135]
Artificial Bee Colony Algorithm	[136]	[137–139]
Ant Colony Optimization	[69]	[140]
Artificial Fish-Swarm Algorithm	[141]	[142–144]
Monkey Search Algorithm	[145]	[146]
Cuckoo Search Algorithm	[147]	[148–150]
Firefly Algorithm	[151]	[152–155]
Krill Herd Algorithm	[156]	[157]
Biogeography-Based Optimization	[158]	[159–161]
Particle Swarm Optimization	[162]	[163–165]
Gravitational Search Algorithm	[166]	[167–169]
Bat Swarm Optimization	[170]	[171, 172]

istic of well-balance of exploration and exploitation, and has shown very competitive results compared to other well-known meta-heuristics [1]. Nevertheless, compared to some state-of-art optimization algorithms, GWO is usually outperformed due to its oversimplified search dynamics. Thus, in this study we for the first time incorporate chaotic search mechanism into GWO to further improve its performance.

In the field of optimization, it is widely accepted that the ergodicity and randomness of chaos are powerful mechanisms to avoid falling into the local search process. In the literature, most meta-heuristics have been attempted to combine with chaotic mechanisms. Table 4.1 summarizes some typical meta-heuristics and their improved ones by incorporated chaos. It should be pointed out that Table 4.1 is not aiming to give a comprehensive summary of such chaotic combinations, but to reveal that chaotic mechanisms indeed enable algorithm to possess better performances. Thus, it is natural and interesting to find out the effect of chaos on GWO. This work considers 12 kinds of chaotic maps to perform the chaotic local search (CLS). It is apparent that different chaotic maps have distinct distribution characteristics and the related

CLS thus manipulates different search dynamics. The motivation of this study is two-fold: (1) the verification of the effects of CLS on GWO should be made; and (2) the insights of which chaotic map is most suitable for GWO should also be given. To realize these, extensive experiments are conducted based on 29 widely used benchmark functions. Simulation results suggest that CLS indeed enables GWO to possess better performance in terms of solution accuracy, solution distribution, and convergence property. Further discussions also reveal that the performance of CGWO is effected not only by the characteristics of the embedded chaotic map, but also by the landscape of the solved problems.

4.2 Grey Wolf Optimization

GWO is a population-based meta-heuristic which is inspired by the mechanism of the hunting behavior and social leadership rule of grey wolves. GWO involves five procedures to perform the search for an optimization problem: social hierarchy division, encircling prey, hunting, attacking prey, and search for prey. Fig. 4.1 gives a conceptual graph of GWO. In social hierarchy procedure, the whole population of wolves is divided into four groups. The first three leader wolves (i.e., with highest fitness for an optimization problem) α, β, and δ can guide other wolves (denoted by ω) to move toward promising search areas. In encircling prey procedure, grey wolves encircle prey during the hunt (as shown in Fig. 4.1) according to the following rules:

$$\vec{D} = |\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)|, \tag{4.1}$$

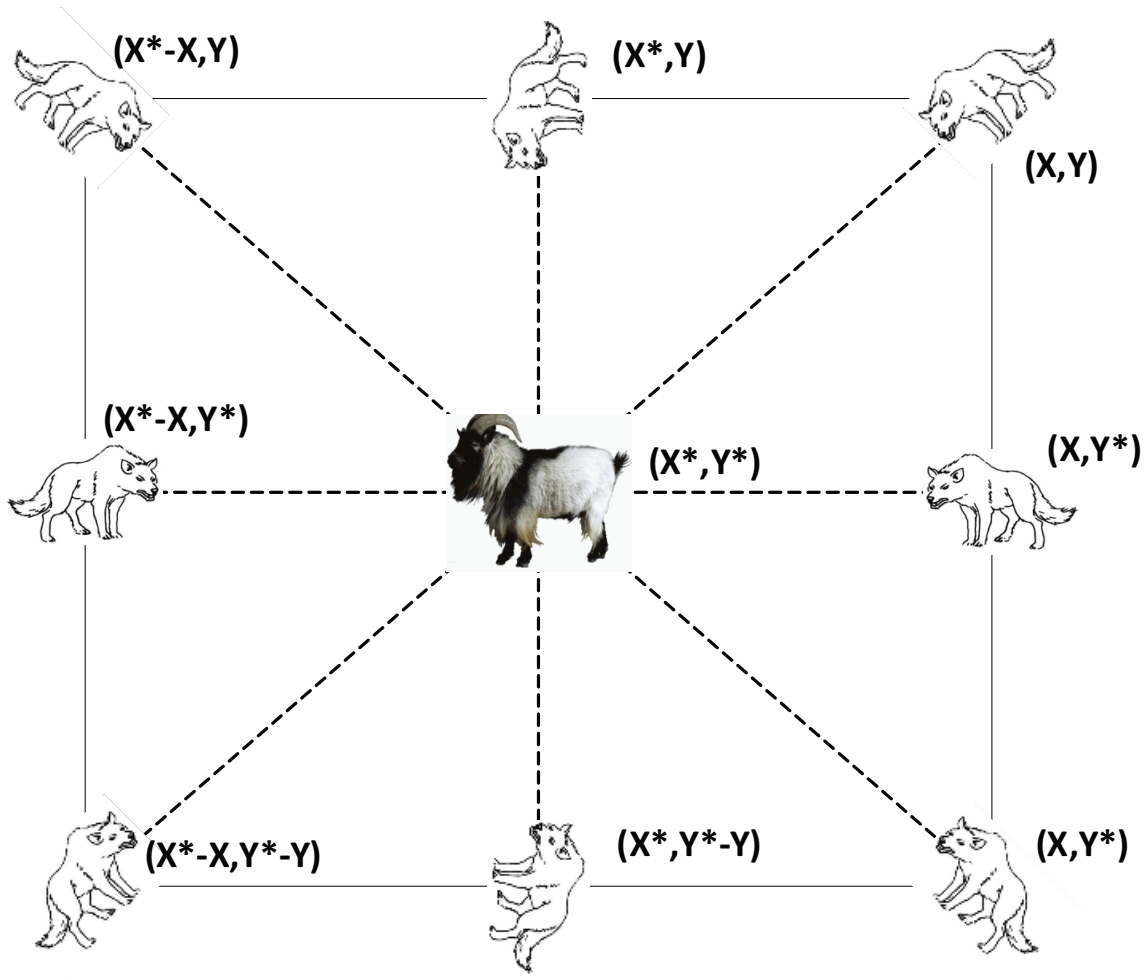

Figure 4.1: Conceptual graph of the mechanism of grey wolf optimization.

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D}, \tag{4.2}$$

where t denotes the iteration number, $\vec{X}_p$ and $\vec{X}$ indicates the position of the prey and a grey wolf respectively. $\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a}$, $\vec{C} = 2\vec{r}_2$, the components of parameter $\vec{a}$ are linearly decreased from 2 to 0 along with the iteration, and $\vec{r}_1, \vec{r}_2$ are random vectors in [0, 1].

After recognizing the position of the prey, a hunting process is carried out by moving the ω wolves which are guided by the leader wolves based on the following

equations:

$$\vec{D}_i = |\vec{C}_i \cdot \vec{X}_i(t) - \vec{X}(t)|, \qquad i = \{\alpha, \beta, \delta\}, \tag{4.3}$$

$$\vec{X}(t+1) = \frac{\sum_{i=\{\alpha,\beta,\delta\}} \vec{X}_i(t) - \vec{A}_i \cdot \vec{D}_i}{3}, \tag{4.4}$$

where $\vec{X}_i$ ($i = \{\alpha, \beta, \delta\}$) denotes the position of the leader wolves α, β, and δ respectively. $\vec{C}_i$ and $\vec{A}_i$ are random vectors. After hunting, the ω wolves update their positions randomly around the prey by the guided leader wolves who estimate the position of the prey.

The attacking prey and search for prey procedures are automatically realized depending on the values of two vectors $\vec{A}$ and $\vec{C}$. From the optimization perspective, the attacking prey is an exploitation process which aims to make the search converge to promising solutions, while the search for prey process is an exploration to diverge the search and thus has ability of jumping out of the local optimal solution. As illustrated in Fig. 4.1, the exploration takes place when $\vec{A}$ is greater than 1 or less than -1, or when $\vec{C}$ is greater than 1. On the contrary, the exploitation is performed when $|A| < 1$ and $|C| < 1$. More details of GWO should be referred to as in [1].

4.3 Chaotic Grey Wolf Optimization

In this paper, we for the first time incorporate chaotic search mechanisms into GWO to further improve its search performance. Twelve different chaotic maps shown in Table 4.2 are utilized to generate chaotic variable sequences. In the literature, there

Table 4.2: The definition of twelve chaotic maps.

Name	Equation
Logistic map	$z_{k+1} = 4z_k(1 - z_k)$
PWLCM	$z_{k+1} = \begin{cases} z_k/0.7, & z_k \in (0, 0.7) \\ (1 - z_k)(1 - 0.7), & z_k \in [0.7, 1) \end{cases}$
Singer map	$z_{k+1} = 1.073(7.86z_k - 23.31z_k^2 + 28.75z_k^3 - 13.302875z_k^4)$
Sine map	$z_{k+1} = \sin(\pi z_k)$
Sinusoidal map	$z_{k+1} = 2.3z_k^2 sin(\pi z_k)$
Tent map	$z_{k+1} = \begin{cases} z_k/0.4, & 0 < z_k \leq 0.4 \\ (1 - z_k)/0.6, & 0.4 < z_k \leq 1 \end{cases}$
Bernoulli map	$z_{k+1} = \begin{cases} z_k/0.6, & 0 < z_k \leq 0.6 \\ (z_k - 0.6)/0.4, & 0.6 < z_k < 1 \end{cases}$
Chebyshev map	$z_{k+1} = \cos(0.5 \cos^{-1} z_k)$
Circle map	$z_{k+1} = z_k + 0.5 - \frac{1.1}{\pi} \sin(2\pi z_k) \text{mod}(1)$
Cubic map	$z_{k+1} = 2.59z_k(1 - z_k^2)$
Gaussian map	$z_{k+1} = \begin{cases} 0, & z_k = 0 \\ (1/z_k) \text{mod}(1) & z_k \neq 0 \end{cases}$
ICMIC	$z_{k+1} = \sin(70/z_k)$

are two kinds of incorporation schemes of chaos when embedding into meta-heuristics. One is to use chaotic maps to generate chaotic sequences to substitute the random numbers in the original algorithm, e.g. [132]. The other scheme which is usually considered to be more effective is to use chaos to perform a chaotic local search (CLS). In this study, twelve variants of CLS derived from twelve different chaotic maps are proposed to embed into GWO. The pseudo code of CLS is shown as in Algorithm 1.

Algorithm 1: CLS

begin
- Set the parameters of a chaotic system;
- According to the selected chaotic system, get a chaotic sequence;
- Choose the best individual $\vec{X}_\alpha$ in the current population;
- Superimpose an item of the chaotic sequence on $\vec{X}_\alpha$ in any dimension to form a new individual that is marked as $\vec{X}_n$, using $\vec{X}_n = \vec{X}_\alpha + r(U - L)(z - 0.5)$;
- Calculation the fitness value of the new individual $\vec{X}_n$;
- **for** *the optimization function* f **do**
  - **if** $f(\vec{X}_\alpha) > f(\vec{X}_n)$ **then**
    - $\vec{X}_\alpha \leftarrow \vec{X}_n$

In Algorithm 1, the search neighborhood of $\vec{X}_n$ is conducted in a hypercube whose center is $\vec{X}_\alpha$ with a radius r. U and L are the search upper and lower boundary of the search space respectively. z is a chaotic variable generated by the selected chaotic map. The search range is narrowed along with the iteration of the algorithm via $r(t+1) = 0.988r(t)$. Based on the GWO and CLS, the CGWO is proposed as shown in Algorithm 2. It is worth emphasizing that the local search is only applied to the current best wolf $\vec{X}_\alpha$ obtained from the GWO. Compared with carrying out local search on all wolves, it is expected that this scheme can not only save computational times, but also produce competitive good solutions.

Algorithm 2: CGWO

begin
- Initialize the grey wolf population $\vec{X}_i (i = 1, 2, ..., N)$;
- Initialize $\vec{a}$, $\vec{A}$, and $\vec{C}$;
- Calculate the fitness of each search agent;
- $\vec{X}_\alpha$ = the best search wolf;
- $\vec{X}_\beta$ = the second best search wolf;
- $\vec{X}_\delta$ = the third best search wolf;
- **while** $t <$ *Maximal number of iterations* **do**
  - **for** *Each search wolf* **do**
    - Update the position of the current search wolf by Eq. (4);
  - Update $\vec{a}$, $\vec{A}$, and $\vec{C}$;
  - Calculate the fitness of all search wolves;
  - Update $\vec{X}_\alpha$, $\vec{X}_\beta$, and $\vec{X}_\delta$;
  - Implement chaotic local search approach (CLS) based on a selected chaotic map;
  - Decrease chaotic local search radius using $r(t+1) = 0.988r(t)$;
  - $t = t + 1$;
- Return $\vec{X}_\alpha$;

4.4 Experimental Results and Discussions

To verify the performance of the proposed CGWO, 29 widely used benchmark functions are used where the fist 23 ones are classical functions [173] and the last 6 ones

are taken from CEC 2005 [174]. To be specific, F1 and F5 are unimodal functions; F6 is a step function which has only one minimum and is discontinuous; F7 is a noisy quartic function; F8-F13 are multimodal functions with plenty of local minima and the number of the local minima in these functions increase exponentially with the dimension of the function; F14-F23 are low dimensional functions which only have a few local minima. These functions can successfully test the searching capacity of algorithms in terms of convergence speed and global exploration ability. In other words, unimodal functions are able to reflect the convergence speed of the algorithm in a direct manner, and multimodal ones are likely to estimate the algorithms' ability of escaping from local minima. F24-F29 are shifted or rotated functions where global minima lie at a fixed location of the search range (difficult to be predicted in advance) and linkage among the variables exists to make these variables are non-separated, and thus make the optimal solutions for these functions hard to be found.

The GWO algorithm and the proposed 12 variants of CGWO algorithms are implemented for 30 independent runs to make a performance comparison. The parameters in these algorithms are set to be identical. That is: the population size of wolves is $N = 30$, the maximal number of iteration is $T_{max} = 500$, the parameter $a = 2 - t \times \frac{2}{T_{max}}$ where t denotes the iteration number, the initial value of chaotic number $z_0 = 0.152$, and the initial search radius $r(0) = 0.0001$. The experimental results for 29 functions are summarized in Tables 4.3- 4.7, where average values (Average) and standard deviation (SD) of 30 runs are recorded in the form of "Average $\pm$ SD". The best values among all compared 13 algorithms are emphasized using bold fonts. From Tables 4.3- 4.7, it is apparent that CGWO outperform GWO on 27 out of 29 functions in

terms of solution accuracy. It suggests that CLS indeed enables GWO to find better solutions by enriching the search dynamics in a local search manner. Not surprisingly, it is difficult to find out which chaotic map consistently improve the performance of GWO over all tested functions, inferring that the search performance of CGWO is effected not only by the characteristics of the embedded chaotic map, but also by the landscape of the solved problems.

To further give some insights into the search results of the compared algorithms, the final solutions' distribution obtained from 30 runs for F27 and F29 are plotted in Fig. 4.2 using the box-and-whisker diagram. Both two functions are very hard to be solved, thus the robustness of all algorithms is sensitive, i.e., many outlier solutions exits. Nevertheless, CGWO generally possesses smaller values in terms of minimum, median, first quartile, third quartile and maximum. On the other hand, Figs. 4.3 and 4.4 depict the convergence graph of all compared algorithms for F27 and F29 respectively. From Figs. 4.3(a) and 4.4(a), it is clear that all algorithms converge quickly in the early search phases and slow down in latter phases, and CGWO generally converges faster than GWO. In addition, we define the ratio of best-so-far solutions found by 12 chaotic maps to those found by GWO verse the iteration. For Figs. 4.3(b) and 4.4(b), the values above the horizontal line (i.e., represented GWO) means worse solutions found by the chaotic algorithm than those by GWO, and below the line indicates a better performance. These two figures clearly show that CGWO significantly outperform GWO in the latter search phases, and thus overall has a better performance in terms of solution accuracy, solution distribution, and convergence property.

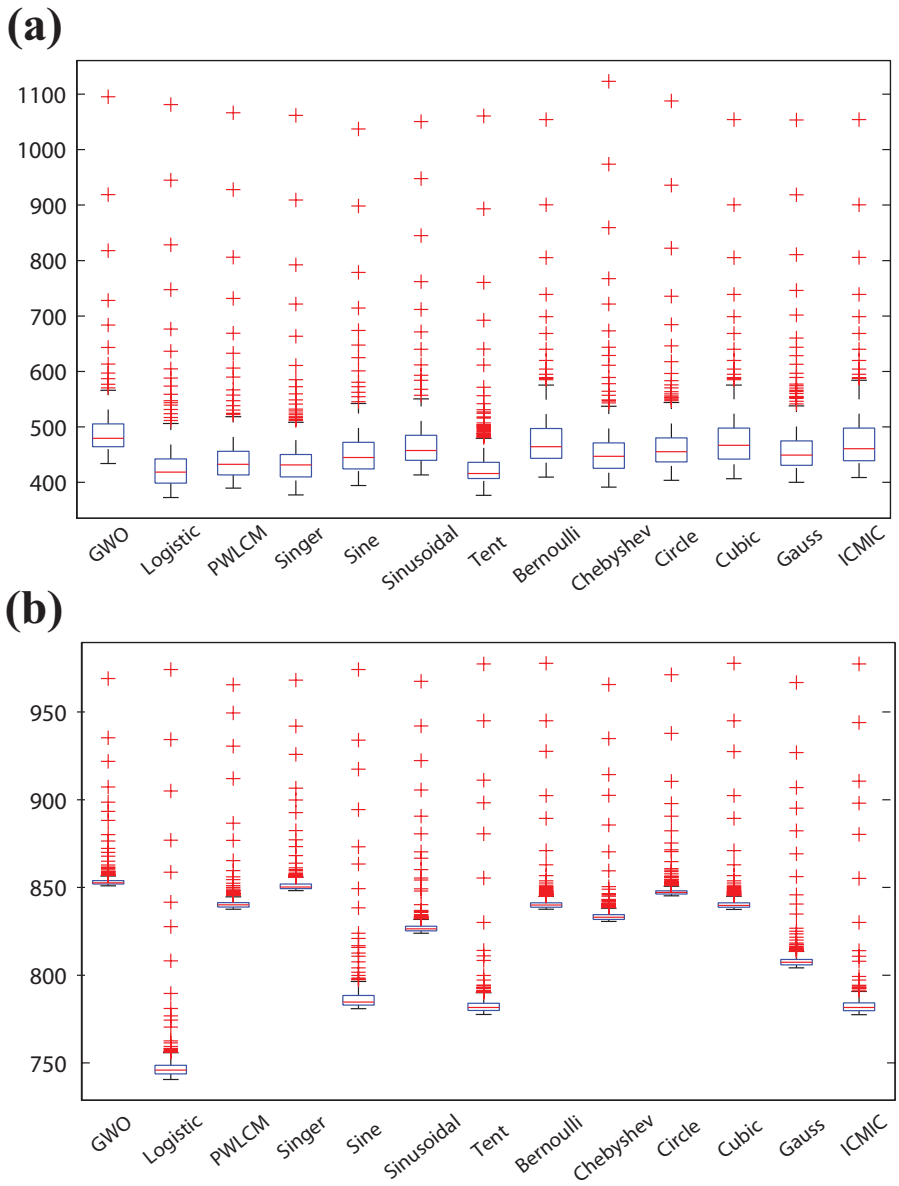

Figure 4.2: Solution distribution for (a) F27 and (b) F29.

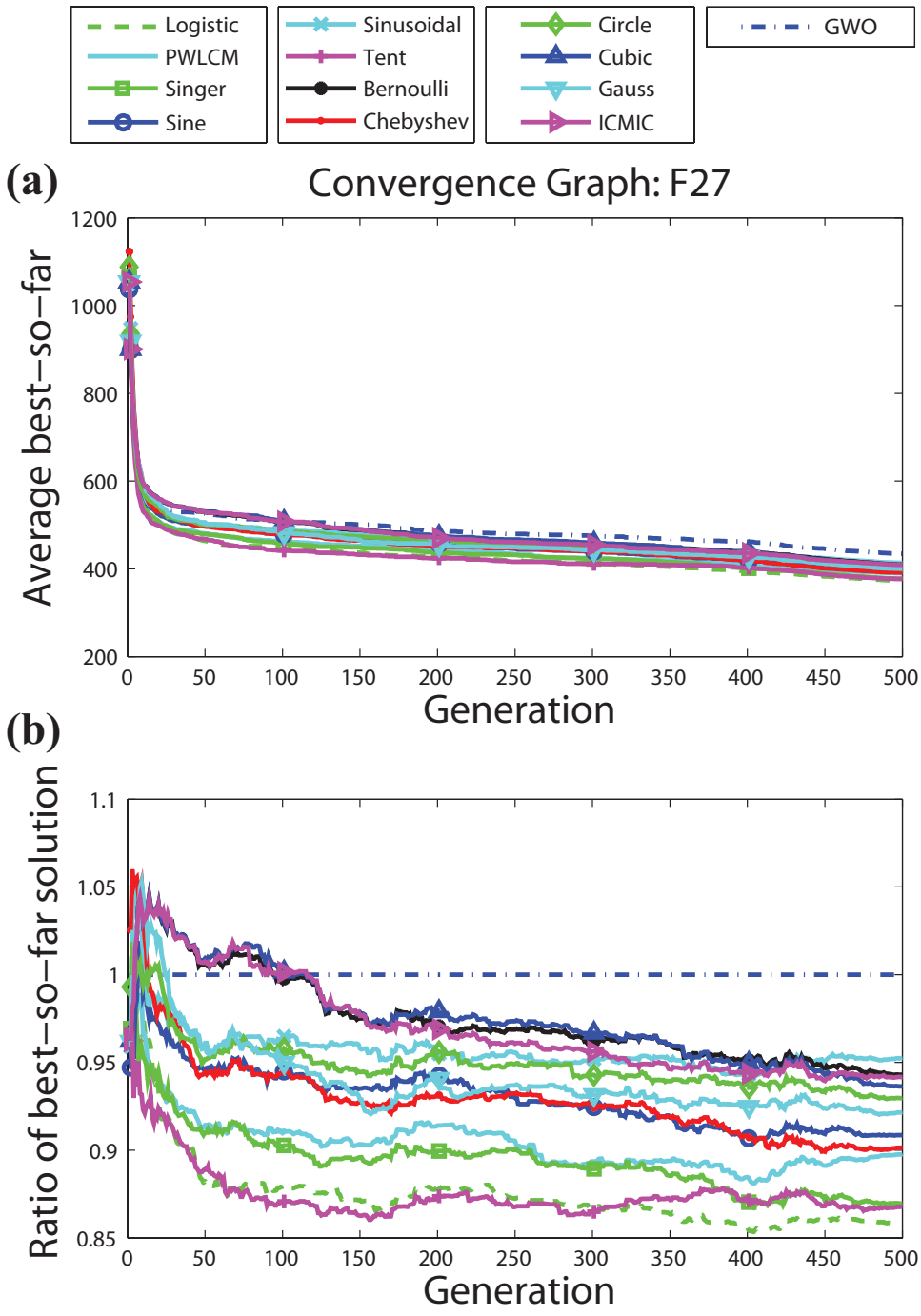

Figure 4.3: Search performance of the algorithms for comparison on F27.

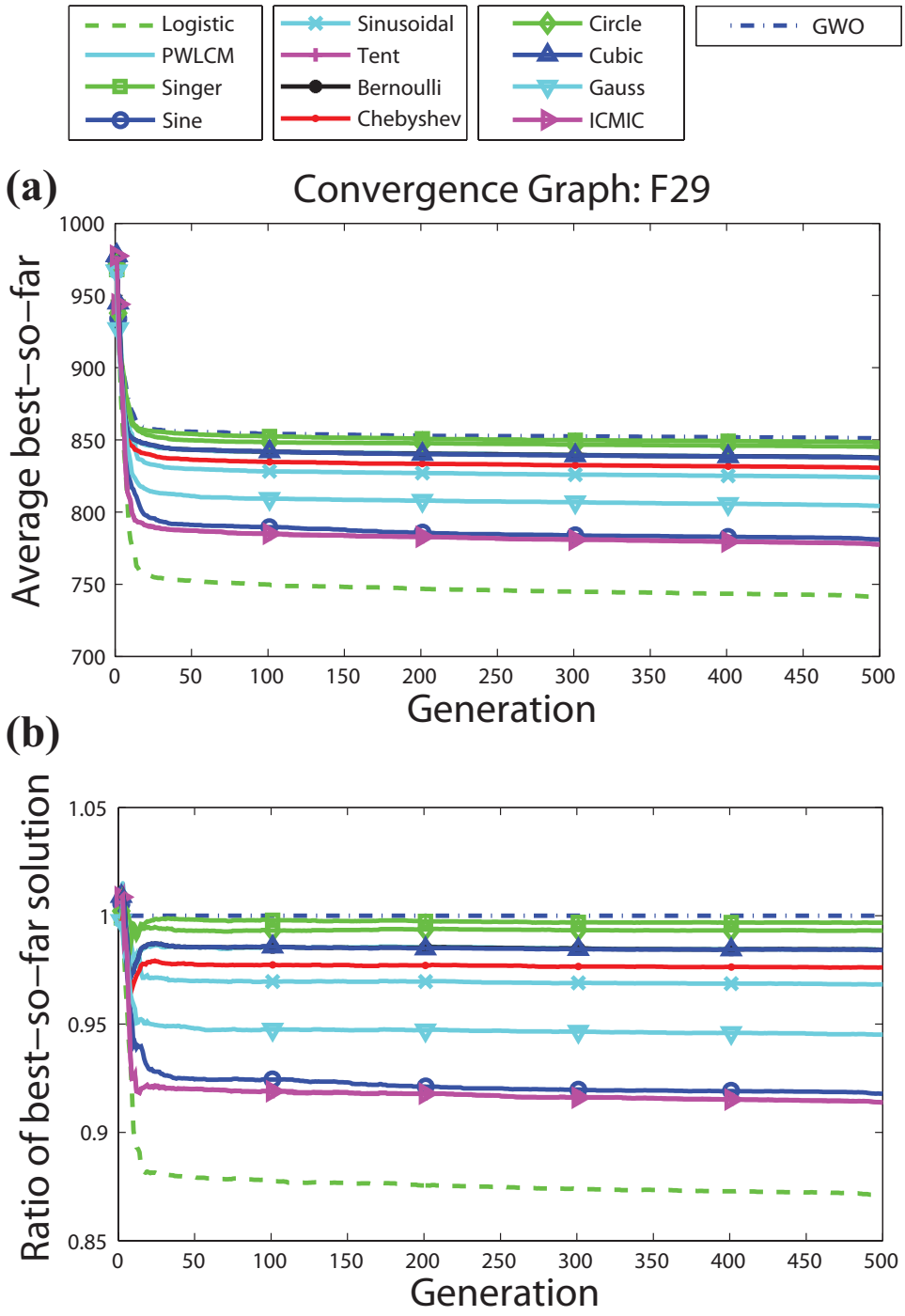

Figure 4.4: Search performance of the algorithms for comparison on F29.

Table 4.3: Experimental results of benchmark functions (F1-F6) using GWO and CGWO with 12 different chaotic maps.

Algorithm	F1	F2	F3	F4	F5	F6
GWO	1.95E-27 ± 4.89E-27	8.71E-17 ± 6.43E-17	2.73E-05 ± 6.42E-05	5.36E-07 ± 4.76E-07	2.71E+01 ± 6.06E-01	7.88E-01 ± 3.14E-01
Logistic	2.03E-27 ± 4.75E-27	1.02E-16 ± 5.61E-17	2.59E-05 ± 7.05E-05	8.45E-07 ± 7.40E-07	2.71E+01 ± 7.76E-01	7.73E-01 ± 4.26E-01
PWLCM	1.88E-27 ± 4.31E-27	8.33E-17 ± 6.21E-17	1.03E-05 ± 2.85E-05	1.06E-06 ± 1.44E-06	2.71E+01 ± 8.63E-01	8.47E-01 ± 3.53E-01
Singer	1.17E-27 ± 2.39E-27	8.73E-17 ± 6.22E-17	**2.96E-06 ± 6.41E-06**	7.41E-07 ± 9.87E-07	2.72E+01 ± 8.09E-01	7.44E-01 ± 3.22E-01
Sine	**7.92E-28 ± 8.32E-28**	**8.27E-17 ± 5.57E-17**	1.17E-04 ± 5.63E-04	**5.24E-07 ± 4.81E-07**	2.70E+01 ± 7.15E-01	6.88E-01 ± 3.32E-01
Sinusoidal	1.30E-27 ± 1.94E-27	1.12E-16 ± 1.05E-16	1.29E-05 ± 4.77E-05	1.07E-06 ± 1.50E-06	**2.69E+01 ± 7.96E-01**	8.59E-01 ± 4.24E-01
Tent	1.48E-27 ± 3.65E-27	9.60E-17 ± 8.27E-17	2.90E-05 ± 1.05E-04	5.72E-07 ± 4.51E-07	2.72E+01 ± 9.25E-01	8.24E-01 ± 3.73E-01
Bernoulli	1.57E-27 ± 2.95E-27	1.05E-16 ± 8.21E-17	1.22E-05 ± 3.99E-05	8.81E-07 ± 8.82E-07	2.70E+01 ± 7.98E-01	8.46E-01 ± 4.02E-01
Chebyshev	2.40E-27 ± 4.37E-27	1.14E-16 ± 9.92E-17	2.73E-05 ± 8.50E-05	1.10E-06 ± 1.63E-06	2.71E+01 ± 6.93E-01	7.76E-01 ± 4.38E-01
Circle	2.75E-27 ± 4.89E-27	1.05E-16 ± 8.38E-17	3.98E-05 ± 1.11E-04	8.24E-07 ± 9.55E-07	2.72E+01 ± 8.37E-01	8.34E-01 ± 4.06E-01
Cubic	1.53E-27 ± 2.16E-27	9.50E-17 ± 5.28E-17	1.51E-05 ± 3.84E-05	8.41E-07 ± 9.88E-07	2.72E+01 ± 7.16E-01	**6.67E-01 ± 3.95E-01**
Gauss	1.23E-27 ± 1.53E-27	9.82E-17 ± 5.95E-17	1.23E-05 ± 2.55E-05	1.17E-06 ± 1.26E-06	2.71E+01 ± 6.49E-01	7.99E-01 ± 3.97E-01
ICMIC	1.30E-27 ± 1.97E-27	1.28E-16 ± 1.02E-16	1.60E-05 ± 4.97E-05	8.62E-07 ± 7.17E-07	2.71E+01 ± 6.46E-01	7.18E-01 ± 2.61E-01

Table 4.4: Experimental results of benchmark functions (F7-F12) using GWO and CGWO with 12 different chaotic maps.

Algorithm	F7	F8	F9	F10	F11	F12
GWO	1.90E-03 ± 1.04E-03	-6.07E+03 ± 6.10E+02	3.12E+00 ± 3.53E+00	**1.01E-13 ± 1.39E-14**	2.65E-03 ± 6.17E-03	4.52E-02 ± 1.98E-02
Logistic	1.24E-03 ± 7.99E-04	-6.16E+03 ± 7.58E+02	2.92E+00 ± 3.64E+00	1.03E-13 ± 1.68E-14	3.78E-03 ± 9.02E-03	4.47E-02 ± 2.96E-02
PWLCM	9.95E-04 ± 4.56E-04	-5.96E+03 ± 7.84E+02	2.94E+00 ± 3.80E+00	1.05E-13 ± 2.27E-14	2.63E-03 ± 5.59E-03	4.77E-02 ± 2.98E-02
Singer	9.14E-04 ± 4.58E-04	-5.78E+03 ± 8.69E+02	**2.47E+00 ± 3.11E+00**	1.03E-13 ± 1.54E-14	3.83E-03 ± 8.36E-03	4.77E-02 ± 2.77E-02
Sine	1.09E-03 ± 6.23E-04	-5.99E+03 ± 9.17E+02	2.75E+00 ± 3.50E+00	1.01E-13 ± 1.40E-14	4.20E-03 ± 1.25E-02	4.28E-02 ± 2.42E-02
Sinusoidal	1.10E-03 ± 6.10E-04	**-6.36E+03 ± 5.04E+02**	2.56E+00 ± 3.72E+00	1.05E-13 ± 2.01E-14	2.16E-03 ± 6.02E-03	4.69E-02 ± 1.96E-02
Tent	1.02E-03 ± 5.65E-04	-6.07E+03 ± 8.49E+02	2.54E+00 ± 3.52E+00	1.02E-13 ± 1.49E-14	5.31E-03 ± 7.98E-03	5.02E-02 ± 3.09E-02
Bernoulli	9.03E-04 ± 4.17E-04	-6.27E+03 ± 8.51E+02	3.15E+00 ± 6.08E+00	1.04E-13 ± 1.78E-14	5.26E-03 ± 9.08E-03	4.16E-02 ± 2.29E-02
Chebyshev	**8.78E-04 ± 4.60E-04**	-6.20E+03 ± 8.85E+02	3.40E+00 ± 3.94E+00	1.08E-13 ± 2.41E-14	2.59E-03 ± 6.35E-03	4.83E-02 ± 2.23E-02
Circle	1.01E-03 ± 5.20E-04	-6.04E+03 ± 8.94E+02	2.79E+00 ± 3.54E+00	1.09E-13 ± 2.25E-14	9.14E-03 ± 1.35E-02	4.96E-02 ± 2.31E-02
Cubic	1.17E-03 ± 5.91E-04	-6.08E+03 ± 6.35E+02	2.60E+00 ± 3.05E+00	1.07E-13 ± 2.24E-14	2.63E-03 ± 6.10E-03	4.88E-02 ± 2.39E-02
Gauss	9.91E-04 ± 5.35E-04	-5.90E+03 ± 8.15E+02	2.70E+00 ± 4.22E+00	1.04E-13 ± 1.57E-14	**1.94E-03 ± 7.04E-03**	4.08E-02 ± 1.73E-02
ICMIC	9.68E-04 ± 4.33E-04	-5.72E+03 ± 9.91E+02	2.51E+00 ± 3.97E+00	1.05E-13 ± 1.82E-14	4.28E-03 ± 8.05E-03	**3.75E-02 ± 1.49E-02**

Table 4.5: Experimental results of benchmark functions (F13-F18) using GWO and CGWO with 12 different chaotic maps.

Algorithm	F13	F14	F15	F16	F17	F18
GWO	6.41E-01 ± 2.55E-01	3.58E+00 ± 3.41E+00	5.06E-03 ± 8.59E-03	-1.03E+00 ± 4.13E-08	3.98E-01 ± 2.28E-06	3.00E+00 ± 3.30E-05
Logistic	6.05E-01 ± 1.96E-01	5.24E+00 ± 4.33E+00	6.45E-03 ± 9.27E-03	-1.03E+00 ± 1.75E-08	3.98E-01 ± 2.58E-04	3.00E+00 ± 4.58E-05
PWLCM	6.76E-01 ± 2.66E-01	5.43E+00 ± 4.65E+00	4.44E-03 ± 8.10E-03	**-1.03E+00 ± 1.19E-08**	3.98E-01 ± 1.88E-04	3.00E+00 ± 4.52E-05
Singer	6.28E-01 ± 1.85E-01	5.33E+00 ± 4.70E+00	6.39E-03 ± 9.30E-03	-1.03E+00 ± 2.71E-08	3.98E-01 ± 2.69E-06	3.00E+00 ± 3.47E-05
Sine	6.29E-01 ± 1.93E-01	4.78E+00 ± 4.50E+00	5.05E-03 ± 8.59E-03	-1.03E+00 ± 2.36E-08	3.98E-01 ± 6.01E-06	3.00E+00 ± 3.76E-05
Sinusoidal	6.74E-01 ± 2.01E-01	6.11E+00 ± 4.75E+00	5.14E-03 ± 8.54E-03	-1.03E+00 ± 2.24E-08	3.98E-01 ± 2.52E-06	5.70E+00 ± 1.48E+01
Tent	6.85E-01 ± 1.45E-01	4.07E+00 ± 3.85E+00	3.71E-03 ± 7.58E-03	-1.03E+00 ± 2.28E-08	3.98E-01 ± 2.01E-04	3.00E+00 ± 4.30E-05
Bernoulli	6.49E-01 ± 2.69E-01	5.01E+00 ± 4.20E+00	**3.12E-03 ± 6.89E-03**	-1.03E+00 ± 1.96E-08	3.98E-01 ± 2.37E-06	5.70E+00 ± 1.48E+01
Chebyshev	6.85E-01 ± 2.44E-01	3.91E+00 ± 3.83E+00	8.42E-03 ± 9.92E-03	-1.03E+00 ± 2.30E-08	3.98E-01 ± 5.20E-06	3.00E+00 ± 4.48E-05
Circle	6.47E-01 ± 2.50E-01	4.32E+00 ± 4.35E+00	5.09E-03 ± 8.57E-03	-1.03E+00 ± 4.32E-08	3.98E-01 ± 1.46E-04	3.00E+00 ± 4.87E-05
Cubic	6.37E-01 ± 2.85E-01	5.07E+00 ± 4.65E+00	3.79E-03 ± 7.55E-03	-1.03E+00 ± 2.78E-08	3.98E-01 ± 9.96E-05	3.00E+00 ± 4.73E-05
Gauss	**5.92E-01 ± 1.99E-01**	4.33E+00 ± 3.68E+00	5.07E-03 ± 8.58E-03	-1.03E+00 ± 3.12E-08	**3.98E-01 ± 2.22E-06**	**3.00E+00 ± 2.51E-05**
ICMIC	6.34E-01 ± 1.94E-01	**3.48E+00 ± 3.62E+00**	3.14E-03 ± 6.88E-03	-1.03E+00 ± 3.14E-08	3.98E-01 ± 1.97E-05	5.70E+00 ± 1.48E+01

Table 4.6: Experimental results of benchmark functions (F19-F23) using GWO and CGWO with 12 different chaotic maps.

Algorithm	F19	F20	F21	F22	F23
GWO	-3.86E+00 ± 2.37E-03	-3.24E+00 ± 7.05E-02	-9.81E+00 ± 1.28E+00	-1.04E+01 ± 8.16E-04	-1.03E+01 ± 1.48E+00
Logistic	-3.86E+00 ± 2.48E-03	-3.26E+00 ± 7.87E-02	-9.23E+00 ± 2.13E+00	-1.04E+01 ± 1.04E-03	**-1.05E+01 ± 9.65E-04**
PWLCM	-3.86E+00 ± 2.25E-03	-3.27E+00 ± 7.38E-02	-9.40E+00 ± 2.00E+00	-1.02E+01 ± 9.70E-01	-1.01E+01 ± 1.75E+00
Singer	-3.86E+00 ± 2.12E-03	**-3.29E+00 ± 5.45E-02**	-8.81E+00 ± 2.53E+00	**-1.04E+01 ± 7.75E-04**	-1.02E+01 ± 1.37E+00
Sine	-3.86E+00 ± 2.73E-03	-3.23E+00 ± 8.46E-02	-9.64E+00 ± 1.55E+00	-1.04E+01 ± 9.82E-04	-1.03E+01 ± 1.48E+00
Sinusoidal	-3.86E+00 ± 2.48E-03	-3.24E+00 ± 8.21E-02	-9.64E+00 ± 1.55E+00	-1.04E+01 ± 1.11E-03	-1.03E+01 ± 1.48E+00
Tent	**-3.86E+00 ± 2.11E-03**	-3.26E+00 ± 8.30E-02	-9.56E+00 ± 1.84E+00	-1.02E+01 ± 9.70E-01	-9.82E+00 ± 2.23E+00
Bernoulli	-3.86E+00 ± 2.85E-03	-3.27E+00 ± 7.89E-02	-9.64E+00 ± 1.55E+00	-9.79E+00 ± 1.89E+00	-1.05E+01 ± 1.27E-03
Chebyshev	-3.86E+00 ± 2.51E-03	-3.27E+00 ± 7.29E-02	-9.23E+00 ± 2.14E+00	-1.02E+01 ± 9.70E-01	-1.03E+01 ± 1.48E+00
Circle	-3.86E+00 ± 2.21E-03	-3.28E+00 ± 7.62E-02	-8.80E+00 ± 2.54E+00	-1.04E+01 ± 1.07E-03	-1.05E+01 ± 1.10E-03
Cubic	-3.86E+00 ± 2.96E-03	-3.28E+00 ± 6.39E-02	-8.47E+00 ± 2.67E+00	-1.00E+01 ± 1.34E+00	-9.99E+00 ± 2.06E+00
Gauss	-3.86E+00 ± 2.86E-03	-3.27E+00 ± 8.61E-02	**-9.98E+00 ± 9.23E-01**	-1.04E+01 ± 8.11E-04	-1.05E+01 ± 1.08E-03
ICMIC	-3.86E+00 ± 2.25E-03	-3.23E+00 ± 8.28E-02	-9.64E+00 ± 1.55E+00	-1.00E+01 ± 1.34E+00	-9.72E+00 ± 2.48E+00

Table 4.7: Experimental results of Composite benchmark functions (F24-F29) using GWO and CGWO with 12 different chaotic maps.

Algorithm	F24	F25	F26	F27	F28	F29
GWO	9.41E+01 ± 9.68E+01	1.75E+02 ± 1.08E+02	**1.99E+02 ± 8.51E+01**	4.34E+02 ± 1.62E+02	1.38E+02 ± 1.67E+02	8.51E+02 ± 1.38E+02
Logistic	8.24E+01 ± 9.53E+01	1.64E+02 ± 1.16E+02	2.36E+02 ± 1.12E+02	**3.73E+02 ± 7.87E+01**	1.85E+02 ± 1.67E+02	**7.41E+02 ± 2.05E+02**
PWLCM	1.04E+02 ± 9.58E+01	1.79E+02 ± 9.73E+01	2.41E+02 ± 1.41E+02	3.90E+02 ± 1.17E+02	1.59E+02 ± 1.24E+02	8.38E+02 ± 1.52E+02
Singer	9.84E+01 ± 1.20E+02	1.83E+02 ± 1.01E+02	2.20E+02 ± 9.80E+01	3.78E+02 ± 1.07E+02	1.17E+02 ± 1.50E+02	8.48E+02 ± 1.38E+02
Sine	9.18E+01 ± 9.53E+01	**1.37E+02 ± 1.20E+02**	2.10E+02 ± 1.18E+02	3.94E+02 ± 1.15E+02	1.91E+02 ± 1.66E+02	7.81E+02 ± 1.84E+02
Sinusoidal	9.43E+01 ± 1.16E+02	1.48E+02 ± 9.95E+01	2.43E+02 ± 1.14E+02	4.14E+02 ± 1.31E+02	1.64E+02 ± 1.37E+02	8.24E+02 ± 1.62E+02
Tent	**7.60E+01 ± 8.54E+01**	1.74E+02 ± 1.16E+02	2.49E+02 ± 1.20E+02	3.77E+02 ± 1.16E+02	2.18E+02 ± 1.85E+02	7.78E+02 ± 1.91E+02
Bernoulli	8.22E+01 ± 9.06E+01	1.52E+02 ± 9.97E+01	2.07E+02 ± 9.77E+01	4.09E+02 ± 1.35E+02	1.68E+02 ± 1.58E+02	8.38E+02 ± 1.51E+02
Chebyshev	7.94E+01 ± 9.75E+01	1.55E+02 ± 1.02E+02	2.46E+02 ± 1.01E+02	3.91E+02 ± 1.18E+02	**1.09E+02 ± 1.06E+02**	8.31E+02 ± 1.51E+02
Circle	1.21E+02 ± 1.05E+02	1.69E+02 ± 1.13E+02	2.06E+02 ± 8.94E+01	4.04E+02 ± 1.20E+02	1.60E+02 ± 1.43E+02	8.45E+02 ± 1.36E+02
Cubic	8.16E+01 ± 9.05E+01	1.55E+02 ± 9.86E+01	2.07E+02 ± 9.16E+01	4.07E+02 ± 1.40E+02	1.65E+02 ± 1.60E+02	8.37E+02 ± 1.51E+02
Gauss	9.41E+01 ± 1.07E+02	1.52E+02 ± 9.16E+01	2.41E+02 ± 1.18E+02	4.00E+02 ± 1.37E+02	1.85E+02 ± 1.48E+02	8.04E+02 ± 1.69E+02
ICMIC	8.21E+01 ± 9.04E+01	1.55E+02 ± 1.01E+02	2.10E+02 ± 9.41E+01	4.09E+02 ± 1.37E+02	1.73E+02 ± 1.61E+02	7.78E+02 ± 1.91E+02

4.5 Conclusion

In this paper, we incorporated twelve kinds of chaotic maps into grey wolf optimization (GWO) to further improve its search performance. Experimental results suggested that chaotic local search definitely improves the search capacity of GWO by taking advantage of the ergodicity and randomness of chaos, especially on latter search phases. The resultant chaotic grey wolf optimization (CGWO) thus is more capable of jumping out the local minimum and well balance the exploration and exploitation. The results motivated us to further investigate the different influence of different chaotic maps on GWO, and thereafter give some insights into the design of other chaotic meta-heuristics.

Chapter 5

General Conclusions and Remarks

Natural Computing is a field of research in the field of computational computing technology for human beings. This computing technology is inspired both by nature and by computers in nature, that is, it studies naturally inspired models and computational techniques, studies information processing what happened in nature. The field of natural computing has been the focus of much research in recent decades. In this dissertation, I will give a brief review of the natural algorithms and artificial intelligence.

In chapter 1, we systematically reviewed Natural computing, Artificial intelligence, Machine learning, Heuristic and Metaheuristic.

In chapter 2, we introduce two kinds of nature-inspired algorithms: Decision tree and Gray wolf optimization.

And in this dissertation, we introduced two nature-inspired intelligent algorithms for optimization and prediction problems. They are summarized as follows.

In chapter 3, points out that the previous approaches for predicting the vehicle states by the heavy history information and has a delayed prediction time. Some trajectory prediction methods based on lane changing recognition are proposed. While

a validation method for complicated environment such as multi-lanes and intersection is not available now, this paper proposes a new method for the prediction by using a decision tree in varieties of road segments generated by the driving behaviors. Such decision tree helps to detect driving behaviors and predict the vehicle state in all road segments including sections with multi-lane, transition segment and intersections. The driving behavior recognition improves the accuracy of vehicle state prediction in long term case. Our approach shows advantages in the provided real environments. Furthermore, the proposed technique might lead to the development of vehiclesą́s networking and intelligentialization, and also for providing effective methods to solve vehicles routing problems in dynamic environments.

In chapter 4, we incorporated twelve kinds of chaotic maps into gray wolf optimization (GWO) to further improve its search performance. Experimental results suggested that chaotic local search definitely improves the search capacity of GWO by taking advantage of the ergodicity and randomness of chaos, especially on latter search phases. The resultant chaotic grey wolf optimization (CGWO) thus is more capable of jumping out the local minimum and well balance the exploration and exploitation. The results motivated us to further investigate the different influence of different chaotic maps on GWO, and thereafter give some insights into the design of other chaotic meta-heuristics.

Based on the work I have done, I will focus on the following points in my future research.

First of all, go a step further on various kinds of algorithm mechanisms of Nature-inspired intelligent algorithms. That is because only with deeper understanding of

Nature-inspired intelligent algorithms, it could be possible to keep digging in the study of the construction (or architecture) of new algorithm. Then, improve the performance of the current existent Nature-inspired intelligent algorithms and apply them to solve problems in new fields.

Last but not least, Nature-inspired intelligent algorithms can combine with other computational intelligence algorithms for solving much complex problems.

Bibliography

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

[2] F. Jeanson and A. White, "Dynamic memory for robot control via delay neural networks," in *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*, 2013, pp. 29–30.

[3] A. Brabazon, M. O'Neill, and S. McGarraghy, *Natural computing algorithms*. Springer, 2015.

[4] S. Russell, P. Norvig, and A. Intelligence, "A modern approach," *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, vol. 25, p. 27, 1995.

[5] R. Kurzweil, *The singularity is near: When humans transcend biology*. Penguin, 2005.

[6] M. R. Genesereth and N. J. Nilsson, "Logical foundations of artificial," *Intelligence. Morgan Kaufmann*, vol. 2, p. 27, 1987.

[7] G. Weiss, *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT press, 1999.

[8] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara, "The distributed constraint satisfaction problem: Formalization and algorithms," *IEEE Transactions on knowledge and data engineering*, vol. 10, pp. 673–685, 1998.

[9] D. Poole, A. Mackworth, and R. Goebel, "Computational intelligence," *IEEE Transactions on knowledge and data engineering*, 1998.

[10] N. J. Nilsson, *Artificial intelligence: a new synthesis*. Elsevier, 1998.

[11] C. M. Van der Walt and E. Barnard, "Data characteristics that determine classifier performance," 2006.

[12] E. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg, "Hyper-heuristics: An emerging direction in modern search technology," *Handbook of metaheuristics*, pp. 457–474, 2003.

[13] M. Minsky, "Society of mind: a response to four reviews," *Artificial Intelligence*, vol. 48, pp. 371–396, 1991.

[14] I. Kononenko, "Machine learning for medical diagnosis: history, state of the art and perspective," *Artificial Intelligence in medicine*, vol. 23, pp. 89–109, 2001.

[15] T. M. Mitchell, "Machine learning. 1997," *Burr Ridge, IL: McGraw Hill*, vol. 45, pp. 870–877, 1997.

[16] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[17] A. A. Freitas, "A survey of evolutionary algorithms for data mining and knowledge discovery," in *Advances in evolutionary computing*. Springer, 2003, pp. 819–845.

[18] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," *AI magazine*, vol. 17, p. 37, 1996.

[19] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, pp. 81–106, 1986.

[20] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[21] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 609–616.

[22] F. V. Jensen, *An introduction to Bayesian networks*. UCL press London, 1996.

[23] D. E. Goldberg, *Genetic algorithms*. Pearson Education India, 2006.

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

[25] R. Groner, *Methods of heuristics, Publisher=Routledg, Year=2014,*.

[26] L. Bianchi, M. Dorigo, L. M. Gambardella, and W. J. Gutjahr, "A survey on metaheuristics for stochastic combinatorial optimization," *Natural Computing*, vol. 8, pp. 239–287, 2009.

[27] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Computing Surveys (CSUR)*, vol. 35, pp. 268–308, 2003.

[28] Q. Cao, S. Gao, J. Zhang, Z. Tang, and H. Kimura, "A stochastic dynamic local search method for learning multiple-valued logic networks," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 90, no. 5, pp. 1085–1092, 2007.

[29] S. Gao, Q. Cao, C. Vairappan, J. Zhang, and Z. Tang, "An improved local search learning method for multiple-valued logic network minimization with bi-objectives," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 92, no. 2, pp. 594–603, 2009.

[30] S. Gao, J. Zhang, Z. Tnag, and Q. Cao, "A chaotic dynamic local search method for learning multiple-valued logic networks." *Journal of Multiple-Valued Logic & Soft Computing*, vol. 15, 2009.

[31] S. Gao, Q. Cao, M. Ishii, and Z. Tang, "Local search with probabilistic modeling for learning multiple-valued logic networks," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 94, no. 2, pp. 795–805, 2011.

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

[33] C. Vairappan, H. Tamura, S. 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, no. 7-9, pp. 1870–1877, 2009.

[34] C. Vairappan, S. Gao, H. Tamura, and Z. Tang, "An improved learning of local search for fuzzy controller network," *International Journal of Innovative Computing, Information and Control*, vol. 5, no. 4, pp. 1101–1113, 2009.

[35] J. Yi, G. Yang, S. Gao, and Z. Tang, "A shrinking chaotic maximum neural network for maximum clique problem," *International Journal of Innovative Computing, Information and Control*, vol. 5, no. 5, pp. 1213–1229, 2009.

[36] ——, "Transiently chaotic neural network based on switched cooling and its application to maximum clique problem," *International Journal of Innovative Computing, Information and Control*, vol. 5, no. 6, pp. 1569–1586, 2009.

[37] Z. Zhang, S. Gao, G. Yang, F. Li, and Z. Tang, "An algorithm of supervised learning for elman neural network," *International Journal of Innovative Computing, Information and Control*, vol. 5, no. 10, 2009.

[38] Z. Zhang, Z. Tang, S. Gao, and G. Yang, "Training elman neural network for dynamical system identification using stochastic dynamic batch local search algorithm," *International Journal of Innovative Computing, Information and Control*, vol. 6, pp. 1883–1892, 2010.

[39] ——, "Training elman neural network for dynamic system identification using an adaptive local search algorithm," *International Journal of Innovative. Computing, Information and Control*, vol. 6, no. 5, 2010.

[40] R.-L. Wang, S.-C. Gao, and Z. Tang, "Solving the maximum cut problem using two-phase hopfeild neural network," *International Journal of Innovative Computing, Information and Control*, vol. 6, pp. 3573–3583, 2010.

[41] Z. Zhang, Z. Tang, S. Gao, and G. Yang, "An algorithm of chaotic dynamic adaptive local search method for elman neural network," *International Journal of Innovative Computing, Information and Control*, vol. 7, no. 2, pp. 647–656, 2011.

[42] Z. Sha, L. Hu, Y. Todo, J. Ji, S. Gao, and Z. Tang, "A breast cancer classifier using a neuron model with dendritic nonlinearity," *IEICE Transactions on Information and Systems*, vol. 98, no. 7, pp. 1365–1376, 2015.

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

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

[45] T. Jiang, S. Gao, D. Wang, J. Ji, Y. Todo, and Z. Tang, "A neuron model with synaptic nonlinearities in a dendritic tree for liver disorders," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 12, no. 1, pp. 105–115, 2017.

[46] W. Chen, J. Sun, S. Gao, J. Cheng, J. Wang, and Y. Todo, "Using a single dendritic neuron to forecast tourist arrivals to japan," *IEICE Transactions on Information and Systems*, vol. 100, no. 1, pp. 190–202, 2017.

[47] Y. Yu, Y. Wang, S. Gao, and Z. Tang, "Statistical modeling and prediction for tourism economy using dendritic neural network," *Computational Intelligence and Neuroscience*, vol. 2017, 2017.

[48] Y. Tang, J. Ji, S. Gao, H. Dai, Y. Yu, and Y. Todo, "A pruning neural network model in credit classification analysis," *Computational Intelligence and Neuroscience*, vol. 2018, 2018.

[49] S. Gao, H. Dai, G. Yang, and Z. Tang, "A novel clonal selection algorithm and its application to traveling salesman problem," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 90, no. 10, pp. 2318–2325, 2007.

[50] S. Gao, Z. Tang, H. Dai, and J. Zhang, "An improved clonal selection algorithm and its application to traveling salesman problems," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 90, no. 12, pp. 2930–2938, 2007.

[51] ——, "A hybrid clonal selection algorithm," *International Journal of Innovative Computing, Information and Control*, vol. 4, no. 4, pp. 995–1008, 2008.

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

[53] S. Gao, H. Dai, J. Zhang, and Z. Tang, "An expanded lateral interactive clonal selection algorithm and its application," *IEICE Transactions on Fundamentals*

of Electronics, Communications and Computer Sciences, vol. 91, no. 8, pp. 2223–2231, 2008.

[54] H. Dai, Y. Yang, C. Li, J. Shi, S. Gao, and Z. Tang, "Quantum interference crossover-based clonal selection algorithm and its application to traveling salesman problem," *IEICE Transactions on Information and Systems*, vol. 92, no. 1, pp. 78–85, 2009.

[55] S. Gao, Z. Tang, and C. Vairappan, "An effective immune algorithm for multiple-valued logic minimization problems," *International Journal of Innovative Computing, Information and Control*, vol. 5, no. 11, 2009.

[56] W. Wang, S. Gao, and Z. Tang, "Improved pattern recognition with complex artificial immune system," *Soft Computing*, vol. 13, no. 12, pp. 1209–1217, 2009.

[57] S. Gao, R.-L. Wang, H. Tamura, and Z. Tang, "A multi-layered immune system for graph planarization problem," *IEICE Transactions on Information and Systems*, vol. 92, no. 12, pp. 2498–2507, 2009.

[58] S. Gao, Q. Cao, Z. Zhang, and Z. Tang, "A chaotic clonal selection algorithm and its application to synthesize multiple-valued logic functions," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 5, no. 1, pp. 105–114, 2010.

[59] S. Gao, R.-L. Wang, M. Ishii, and Z. Tang, "An artificial immune system with feedback mechanisms for effective handling of population size," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 93, no. 2, pp. 532–541, 2010.

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

[61] Z. Xu, Y. Wang, S. Li, Y. Liu, Y. Todo, and S. Gao, "Immune algorithm combined with estimation of distribution for traveling salesman problem," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 11, no. S1, 2016.

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

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

[64] S. Gao, S. Song, J. Cheng, Y. Todo, and M. Zhou, "Incorporation of solvent effect into multi-objective evolutionary algorithm for improved protein structure prediction," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2017.

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

[66] 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.

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

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

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

[70] S. Wang, Aorigele, G. Liu, and S. Gao, "A hybrid discrete imperialist competition algorithm for fuzzy job-shop scheduling problems," *IEEE Access*, vol. 4, pp. 9320–9331, 2016.

[71] Aorigele, Z. Tang, Y. Todo, and S. Gao, "A hybrid discrete imperialist competition algorithm for gene selection for microarray data," *Current Proteomics*, vol. 15, no. 2, pp. 99–110, 2018.

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

[73] Y. Wang, S. Gao, Y. Yu, and Z. Xu, "The discovery of population interaction with a power law distribution in brain storm optimization," *Memetic Computing*, pp. 1–23, 2017.

[74] Y. Yu, S. Gao, S. Cheng, Y. Wang, S. Song, and F. Yuan, "CBSO: a memetic brain storm optimization with chaotic local search," *Memetic Computing*, pp. 1–15, 2017.

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

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

[77] J. Zhang, L. Ni, C. Xie, S. Gao, and Z. Tang, "Inertial estimator learning automata," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 95, no. 6, pp. 1041–1048, 2012.

[78] J. Cheng, S. Gao, C. Vairappan, R.-L. Wang, and A. Ylä-Jääski, "A buffer overflow based algorithm to conceal software watermarking trigger behavior," *IEICE Transactions on Information and Systems*, vol. 97, no. 3, pp. 524–532, 2014.

[79] X. Jiang, M. H. Lee, S. Gao, and Y. Wu, "Optimized geometric ldpc codes with quasi-cyclic structure," *Journal of Communications and Networks*, vol. 16, no. 3, pp. 249–257, 2014.

[80] C. Liu, J. Cheng, Y. Wang, and S. Gao, "Time performance optimization and resource conflicts resolution for multiple project management," *IEICE Transactions on Information and Systems*, vol. 99, no. 3, pp. 650–660, 2016.

[81] C. Liu, Y. Wang, and S. Gao, "Adaptive shape kernel-based mean shift tracker in robot vision system," *Computational Intelligence and Neuroscience*, vol. 2016, 2016.

[82] J. Cheng, L.-F. Hu, S. Gao, J. Liu, and Y. Yang, "An optimized collaborative filtering method to construct spatial-temporal behavior pattern-based user interest model," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 12, no. 2, pp. 221–227, 2017.

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

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

[85] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees, Publisher=CRC press, Year=1984,*.

[86] B. Kamiński, M. Jakubczyk, and P. Szufel, "A framework for sensitivity analysis of decision trees," *Central European Journal of Operations Research*, pp. 1–25.

[87] T. Taleb, M. Ochi, A. Jamalipour, N. Kato, and Y. Nemoto, "An efficient vehicle-heading based routing protocol for vanet networks," in *Wireless Com-*

munications and Networking Conference, 2006. WCNC 2006. IEEE, 2006, pp. 2199–2204.

[88] T. Taleb, E. Sakhaee, A. Jamalipour, K. Hashimoto, N. Kato, and Y. Nemoto, "A stable routing protocol to support its services in vanet networks," *IEEE Transactions on Vehicular Technology*, vol. 56, pp. 3337–3347, 2007.

[89] M. Dixit, R. Kumar, and A. K. Sagar, "Vanet: Architectures, research issues, routing protocols, and its applications," in *Computing, Communication and Automation (ICCCA), 2016 International Conference on*, 2016, pp. 555–561.

[90] J. Liu, J. Wan, Q. Wang, P. Deng, K. Zhou, and Y. Qiao, "A survey on position-based routing for vehicular ad hoc networks," *Telecommunication Systems*, vol. 62, pp. 15–30, 2016.

[91] R. Alsaqour, M. Abdelhaq, and T. Abdullah, "Modeling the position information inaccuracy in manet position-based routing protocols," *Research Journal of Applied Sciences, Engineering and Technology*, vol. 3, pp. 971–976, 2011.

[92] R. A. Alsaqour, M. S. Abdelhaq, and O. A. Alsukour, "Effect of network parameters on neighbor wireless link breaks in gpsr protocol and enhancement using mobility prediction model," *EURASIP Journal on Wireless Communications and Networking*, vol. 2012, p. 171, 2012.

[93] S. Wang, C. Fan, C. H. Hsu, Q. Sun, and F. Yang, "A vertical handoff method via self-selection decision tree for internet of vehicles," *IEEE Systems Journal*, vol. 10, pp. 1183–1192, 2016.

[94] M. Bin, S. Cheng, X. Xie *et al.*, "Modeling and analysis for vertical handoff based on the decision tree in a heterogeneous vehicle network," *IEEE Access*, vol. 5, pp. 8812–8824, 2017.

[95] K. I. Ahmed, "Modeling drivers' acceleration and lane changing behavior," Ph.D. dissertation, Massachusetts Institute of Technology, 1999.

[96] C. Kedowide, C. Gouin-Vallerand, and É. Vallières, "Recognizing blind spot check activity with car drivers based on decision tree classifier approach," in *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014, pp. 22–26.

[97] S. Sivaraman and M. M. Trivedi, "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 4, pp. 1773–1795, 2013.

[98] A. Jazayeri, H. Cai, J. Y. Zheng, and M. Tuceryan, "Vehicle detection and tracking in car video based on motion model," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, pp. 583–595, 2011.

[99] A. Geiger and B. Kitt, "Object flow: A descriptor for classifying traffic motion," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*, 2010, pp. 287–293.

[100] S. Cherng, C. Y. Fang, C. P. Chen, and S. W. Chen, "Critical motion detection of nearby moving vehicles in a vision-based driver-assistance system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, pp. 70–82, 2009.

[101] F. Garcia, P. Cerri, A. Broggi, A. de la Escalera, and J. M. Armingol, "Data fusion for overtaking vehicle detection based on radar and optical flow," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*, 2012, pp. 494–499.

[102] A. Barth and U. Franke, "Tracking oncoming and turning vehicles at intersections," in *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, 2010, pp. 861–868.

[103] D. Kasper, G. Weidl, T. Dang, G. Breuel, A. Tamke, A. Wedel, and W. Rosenstiel, "Object-oriented bayesian networks for detection of lane change maneuvers," *IEEE Intelligent Transportation Systems Magazine*, vol. 4, pp. 19–31, 2012.

[104] Y. Zhu, D. Comaniciu, M. Pellkofer, and T. Koehler, "Reliable detection of overtaking vehicles using robust information fusion," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, pp. 401–414, 2006.

[105] J. Wang, G. Bebis, and R. Miller, "Overtaking vehicle detection using dynamic and quasi-static background modeling," in *Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*, 2005, pp. 64–64.

[106] A. Barth and U. Franke, "Estimating the driving state of oncoming vehicles from a moving platform using stereo vision," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, pp. 560–571, 2009.

[107] B. Barrois and T. . D. B. . c. P. . S. P. . . Y. . . Wöhler, Christian.

[108] J. Wiest, M. Höffken, U. Kreßel, and K. Dietmayer, "Probabilistic trajectory prediction with gaussian mixture models," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*, 2012, pp. 141–146.

[109] S. Sivaraman, B. Morris, and M. Trivedi, "Learning multi-lane trajectories using vehicle-based vision," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, 2011, pp. 2070–2076.

[110] C. Hermes, J. Einhaus, M. Hahn, C. Wöhler, and F. Kummert, "Vehicle tracking and motion prediction in complex urban scenarios," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*, 2010, pp. 26–33.

[111] C. Hermes, C. Wohler, K. Schenk, and F. Kummert, "Long-term vehicle motion prediction," in *Intelligent Vehicles Symposium, 2009 IEEE*. IEEE, 2009, pp. 652–657.

[112] M. Dagdelen, G. Reymond, A. Kemeny, M. Bordier, and N. Maïzi, "Model-based predictive motion cueing strategy for vehicle driving simulators," *Control Engineering Practice*, vol. 17, pp. 995–1003, 2009.

[113] J. Sorstedt, L. Svensson, F. Sandblom, and L. Hammarstrand, "A new vehicle motion model for improved predictions and situation assessment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, pp. 1209–1219, 2011.

[114] R. Pandita and D. Caveney, "Preceding vehicle state prediction," in *Intelligent Vehicles Symposium (IV), 2013 IEEE*. IEEE, 2013, pp. 1000–1006.

[115] T. Hülnhagen, I. Dengler, A. Tamke, T. Dang, and G. Breuel, "Maneuver recognition using probabilistic finite-state machines and fuzzy logic," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*. IEEE, 2010, pp. 65–70.

[116] B. Morris, A. Doshi, and M. Trivedi, "Lane change intent prediction for driver assistance: On-road design and evaluation," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, 2011, pp. 895–901.

[117] A. Houenou, P. Bonnifait, V. Cherfaoui, and W. Yao, "Vehicle trajectory prediction based on motion model and maneuver recognition," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 4363–4369.

[118] D. Petrich, T. Dang, D. Kasper, G. Breuel, and C. Stiller, "Map-based long term motion prediction for vehicles in traffic environments," in *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*. IEEE, 2013, pp. 2166–2172.

[119] P. Kumar, M. Perrollaz, S. Lefevre, and C. Laugier, "Learning-based approach for online lane change intention prediction," in *Intelligent Vehicles Symposium (IV), 2013 IEEE*. IEEE, 2013, pp. 797–802.

[120] W. Yao, H. Zhao, F. Davoine, and H. Zha, "Learning lane change trajectories from on-road driving data," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*. IEEE, 2012, pp. 885–890.

[121] R. Schubert, E. Richter, and G. Wanielik, "Comparison and evaluation of advanced motion models for vehicle tracking," in *Information Fusion, 2008 11th International Conference on*. IEEE, 2008, pp. 1–6.

[122] A. Berthelot, A. Tamke, T. Dang, and G. Breuel, "Handling uncertainties in criticality assessment," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE, 2011, pp. 571–576.

[123] A. Tamke, T. Dang, and G. Breuel, "A flexible method for criticality assessment in driver assistance systems," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE, 2011, pp. 697–702.

[124] N. Mattern, R. Schubert, and G. Wanielik, "High-accurate vehicle localization using digital maps and coherency images," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*, 2010, pp. 462–469.

[125] M. Nitti, V. Pilloni, G. Colistra, and L. Atzori, "The virtual object as a major element of the internet of things: a survey," *IEEE Communications Surveys & Tutorials*, vol. 18, pp. 1228–1240, 2016.

[126] V. Punzo, M. T. Borzacchiello, and B. Ciuffo, "Estimation of vehicle trajectories from observed discrete positions and next-generation simulation program (ngsim) data," in *TRB 2009 Annual Meeting*, 2009.

[127] K. M. Alam, M. Saini, and A. El Saddik, "Toward social internet of vehicles: Concept, architecture, and applications," *IEEE Access*, vol. 3, pp. 343–357, 2015.

[128] Y. Fangchun, W. Shangguang, L. Jinglin, L. Zhihan, and S. Qibo, "An overview of internet of vehicles," *China Communications*, vol. 11, no. 10, pp. 1–15, 2014.

[129] F. W. Glover and G. A. Kochenberger, *Handbook of metaheuristics*. Springer Science & Business Media, 2006.

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

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

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

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

[134] L. dos Santos Coelho and V. C. Mariani, "Combining of chaotic differential evolution and quadratic programming for economic dispatch optimization with valve-point effect," *IEEE Trans. Power Systems*, vol. 21, no. 2, pp. 989–996, 2006.

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

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

[137] B. Alatas, "Chaotic bee colony algorithms for global numerical optimization," *Expert Systems with Applications*, vol. 37, no. 8, pp. 5682–5687, 2010.

[138] C. Xu, H. Duan, and F. Liu, "Chaotic artificial bee colony approach to uninhabited combat air vehicle (ucav) path planning," *Aerospace Science and Technology*, vol. 14, no. 8, pp. 535–541, 2010.

[139] W.-C. Hong, "Electric load forecasting by seasonal recurrent svr (support vector regression) with chaotic artificial bee colony algorithm," *Energy*, vol. 36, no. 9, pp. 5568–5578, 2011.

[140] J. Cai, X. Ma, L. Li, Y. Yang, H. Peng, and X. Wang, "Chaotic ant swarm optimization to economic dispatch," *Electric Power Systems Research*, vol. 77, no. 10, pp. 1373–1380, 2007.

[141] X. Li, "A new intelligent optimization-artificial fish swarm algorithm," *Doctor thesis, Zhejiang University of Zhejiang, China*, 2003.

[142] H. Ma and Y. Wang, "An artificial fish swarm algorithm based on chaos search," in *Fifth International Conference on Natural Computation*, 2009, pp. 118–121.

[143] K. Zhu and M. Jiang, "An improved artificial fish swarm algorithm based on chaotic search and feedback strategy," in *International Conference on Computational Intelligence and Software Engineering*, 2009, pp. 1–4.

[144] W. Guo, G. Fang, and X. Huang, "An improved chaotic artificial fish swarm algorithm and its application in optimizing cascade hydropower stations," in *International Conference on Business Management and Electronic Information*, vol. 3, 2011, pp. 217–220.

[145] A. Mucherino and O. Seref, "Monkey search: a novel metaheuristic search for global optimization," in *Data Mining, Systems Analysis and Optimization in Biomedicine*, vol. 953, no. 1, 2007, pp. 162–173.

[146] L. Zheng, "An improved monkey algorithm with dynamic adaptation," *Applied Mathematics and Computation*, vol. 222, pp. 645–657, 2013.

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

[148] L. Xiang-Tao and Y. Ming-Hao, "Parameter estimation for chaotic systems using the cuckoo search algorithm with an orthogonal learning method," *Chinese Physics B*, vol. 21, no. 5, p. 050507, 2012.

[149] P. Nasa-ngium, K. Sunat, and S. Chiewchanwattana, "Enhancing modified cuckoo search by using mantegna lévy flights and chaotic sequences," in *10th International Joint Conference on Computer Science and Software Engineering*, 2013, pp. 53–57.

[150] G. Wang, S. Deb, A. Gandomi, Z. Zhang, and A. Alavi, "Chaotic cuckoo search," *Soft Computing*, vol. 20, pp. 3349–3362, 2016.

[151] X.-S. Yang, "Firefly algorithm," *Engineering Optimization*, pp. 221–230, 2010.

[152] A. Gandomi, X.-S. Yang, S. Talatahari, and A. Alavi, "Firefly algorithm with chaos," *Communications in Nonlinear Science and Numerical Simulation*, vol. 18, no. 1, pp. 89–98, 2013.

[153] L. dos Santos Coelho, D. L. de Andrade Bernert, and V. C. Mariani, "A chaotic firefly algorithm applied to reliability-redundancy optimization," in *IEEE Congress of Evolutionary Computation (CEC)*, 2011, pp. 517–521.

[154] X.-S. Yang, "Chaos-enhanced firefly algorithm with automatic parameter tuning," *Int. J. Swarm Intell. Res.*, vol. 2, no. 4, pp. 125–36, 2012.

[155] L. dos Santos Coelho and V. C. Mariani, "Firefly algorithm approach based on chaotic tinkerbell map applied to multivariable pid controller tuning," *Computers & Mathematics with Applications*, vol. 64, no. 8, pp. 2371–2382, 2012.

[156] A. H. Gandomi and A. H. Alavi, "Krill herd: a new bio-inspired optimization algorithm," *Communications in Nonlinear Science and Numerical Simulation*, vol. 17, no. 12, pp. 4831–4845, 2012.

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

[158] D. Simon, "Biogeography-based optimization," *IEEE Trans. Evolutionary Computation*, vol. 12, no. 6, pp. 702–713, 2008.

[159] L. Wang and Y. Xu, "An effective hybrid biogeography-based optimization algorithm for parameter estimation of chaotic systems," *Expert Systems with Applications*, vol. 38, no. 12, pp. 15 103–15 109, 2011.

[160] W. Zhu and H. Duan, "Chaotic predator–prey biogeography-based optimization approach for ucav path planning," *Aerospace Science and Technology*, vol. 32, no. 1, pp. 153–161, 2014.

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

[162] K. James and E. Russell, "Particle swarm optimization," in *Proceedings of International Conference on Neural Networks*, 1995, pp. 1942–1948.

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

[164] B. Alatas, E. Akin, and A. B. Ozer, "Chaos embedded particle swarm optimization algorithms," *Chaos, Solitons & Fractals*, vol. 40, no. 4, pp. 1715–1734, 2009.

[165] J. Chuanwen and E. Bompard, "A hybrid method of chaotic particle swarm optimization and linear interior for reactive power optimisation," *Mathematics and Computers in Simulation*, vol. 68, no. 1, pp. 57–65, 2005.

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

[167] C. Li, J. Zhou, J. Xiao, and H. Xiao, "Parameters identification of chaotic system by chaotic gravitational search algorithm," *Chaos, Solitons & Fractals*, vol. 45, no. 4, pp. 539–547, 2012.

[168] 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.

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

[170] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature Inspired Cooperative strategies for Optimization (NICSO 2010)*. Springer, 2010, pp. 65–74.

[171] A. R. Jordehi, "Chaotic bat swarm optimisation (cbso)," *Applied Soft Computing*, vol. 26, pp. 523–530, 2015.

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

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

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