Improved Swarm Intelligent Optimization Based on Immunological and Evolutionary Algorithms

by

Yu YANG

A dissertation

submitted to the Graduate School of Science and Engineering for Education,

University of Toyama

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Engineering

University of Toyama

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

2020

(Submitted January 17, 2020)

Acknowledgements

As a Chinese proverb goes, when drinking water, one should never forget those who dug the well. I would like to take this opportunity to thank a number of people who have offered invaluable assistance during my study and research. Without their help and encouragement, this thesis would likely not have finished.

To my supervisor professor Zheng Tang, who introduced me to the fascinating and challenging field, for his inspiration and his continuing support. Without his help and encouragement, I would never have completed this degree.

Besides, I would like to thank associate professor Shangce Gao, for his open mind, extensive knowledge and rigorous attitude, which deeply infected us. He gave me directions in my work and study and patiently answered my questions. Thank for his valuable opinions and suggestions on the framework and details of my thesis.

I would like to thank my thesis referees, Prof. Akio Ando, Prof. Hiroyuki Okada, Prof. Tang and Associate Prof. Gao, from University of Toyama for a review and qualification of my thesis and giving various valuable comments and suggestions.

To all the members of the Intelligent Information Systems Research Lab, thanks for their great encouragement and help.

Furthermore, I am also grateful to all the members of the faculty and staff of School of Computer and Engineering and related departments in Jiangsu Ocean University. I am also like to give a special thanks to Yamaguchi family for their encouragement and help.

I want to thank my parents, my husband and my lovely son, as well as all the relatives, who have given me the most selfless love, the greatest support. Thank them for everything they have done for me.

Once again, I would like to thank all the teachers, classmates, friends and relatives who care, support and help me, and wish them health, safety and happiness forever.

Abstract

In this thesis, I studied several advanced immunological and evolutionary algorithms, and their applications on optimization problems. The work of this thesis can be summarized as in the following.

First and foremost, an improved quantum immunological algorithm for optimization is presented. Clonal selection mechanism, the theoretical foundation of clonal selection algorithm (CSA) and its variants, was proposed for explaining the essential features of adaptive immune responses: adequate diversity, discrimination of self and non-self, and sustaining immunologic memory. On the basis of the clonal selection theory, only the high affinity immune cells are chosen to proliferate. Those cells with low affinity must be efficiently eliminated. However, the ability of receptor editing to salvage from deletion low affinity immune cells via changing their receptor specificity realized the clonal selection process anew. By combining clonal selection theory and receptor editing, a complete receptor editing operation based quantum clonal selection algorithm is proposed for traveling salesman problem and holes machining path planning problem. Two receptor editing operators (inversion and deletion) work together to improve the performance of CSA. Furthermore, in order to overcome the drawback of asexual proliferation during the immune maturation process, a complete receptor editing operation based quantum interference crossover is used. The effectiveness of the improved algorithm is evaluated on optimization problems including traveling salesman problems and holes machining path planning. The experimental results are also compared with other clonal selection theory based methods.

Secondly, a novel clonal selection algorithm for resource scheduling optimization problem in cloud computing is introduced. Cloud computing, a computing paradigm

that provides a variety of virtualized resources to end users in pay-as-you-go fashion over the internet, has attracted much attention during recent years. Meanwhile, resources scheduling becomes the primary problematic issue in cloud computing due to rapid growth of services, demand, and requirements. To handle this NP-complete resources scheduling problem, an improved Clonal Selection Algorithm (CSA) is proposed in this paper. An improvement operation with vaccine injection is designed to enhance the diversity of solutions. On the other hand, Gauss mutation is used to improve ability to escape from local optima. Moreover, the effects of the proposed algorithm are analyzed and evaluated by comparison with other resource scheduling methods by simulation toolkit - CloudSim. The comparative results show that the proposed algorithm outperforms these algorithms in terms of execution time.

Thirdly, a hybrid ant lion differential evolution for optimization is presented. Ant lion optimization algorithm (ALO) is a swarm-based metaheuristic for optimization inspired by the nature of ant lion hunting. One of the main step of hunting is the random walk of ants around the ant lion, which ensures ALO to possess a good local searching ability. Differential evolution (DE) is an evolutionary algorithm with a structure including mutation, crossover, and selection. The operations of DE are randomly executed which makes DE suffering from week exploiting ability. In this paper, a hybrid differential evolution based on the random walk of ants around the ant lion is presented, which combines the advantages of ant lion optimization algorithm and differential evolution, aiming to well balance the exploitation and exploration of the search. The hybrid algorithm is tested on CEC'17 benchmark suit and clustering problems. Experimental results verify the superiority of the proposed algorithm in comparison with other related algorithms.

Finally, a novel hypercube and spherical evolution for optimization is depicted. In recent two decades, nature-inspired metaheuristic algorithms have been paid more and more attention. Although many new algorithms have been proposed, there is no quintessential difference among classic metaheuristic algorithms. Therefore, some researchers have focused on the essence of search operators in these optimizers. Spherical Evolution (SE) is one of the recent studies on the search style in metaheuristic

algorithms. Contrary to other algorithms, SE adopts a spherical search instead of a hypercube search. In this paper, we focus on the advantages of the two search styles, We propose a hybrid optimizer based on the two complementary search styles, and design a rule to control their utilization. Experimental results based on 30 benchmark functions of CEC2017 show that the proposed optimizer outperforms other state-of-the-art algorithms in terms of effectiveness and robustness.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In real life, many important issues are solved by optimization methods which involve selecting an optimal solution from a number of options, further improving production efficiency without changing existing conditions, and determining the best solution. With the development of science and technology and production management, optimization problems are almost everywhere, including engineering, medicine, finance, physics, chemistry, bioinformatics and other fields. Optimization methods are gradually being achieved in scientific research, engineering technology and economic management. More and more attention and promotion are paid to optimization methods and thus emerging many successful applications, such as intelligent systems, system control, production scheduling, pattern recognition, software engineering and management engineering, etc.

Swarm intelligence is an emerging technology field that has developed rapidly in recent years. Through in-depth observation and analysis of natural phenomena such as biology, society and microphysics, people gradually discover that individuals in nature have simple behavior and limited ability, but when they work together, they exhibit powerful and complex behavioral characteristics. For example, *immunological algorithms* motivated by human biological adaptive immune system, *ant colony algorithms and particle swarm optimization* inspired by biological phenomena, *gravity algorithms* inspired by physical mechanisms, and *colonial competition algorithms* inspired by sociology have been successfully applied to various national production systems, respectively.

The introduction of swarm intelligence has important theoretical value for the development of artificial intelligence and machine learning. It has broad application prospects in many fields such as function optimization, data mining, pattern recognition, information security, anomaly detection and robotics. It is another milestone in human understanding of nature and learning of nature. Therefore, the research of swarm intelligence method provides effective technical assistance and theoretical support for solving optimization problems such as engineering application and production scheduling.

Although the algorithms and systems based on swarm intelligence have been well applied in some complex optimization problems, traditional swarm intelligence has become more common as the scale of the problem has expanded, and some random factors and noise effects in the application environment have been adversely affected. Research on algorithms has encountered bottlenecks in development. In the case of complex optimization problems in dynamic environments, the global optimization ability of traditional swarm intelligence algorithms is greatly reduced. Due to the interference of noise in the environment, it generates a large number of redundant solutions or even infeasible solutions, thus reducing the algorithm and the actual application performance.

For the uncertain and noisy random complex environment, previous research proposes an aggregated intelligent optimization algorithm based on population structure and information interaction network to solve the complex optimization problem of high-dimensional multimodal and stochastic feedback. Firstly, by using the statistical information contained in the prior data, the algorithm is first hierarchically clustered to form a multi-layered population model for the specific search space, and the dynamics of the evolutionary search are analyzed from the perspective of the information interaction network formed by the complex network. First, the optimal structural features under the layering are found. Then, it is used that the collaborative computing function between the populations under the optimal population structure to eliminate the hypothesis requirements of learning independent and identical distribution of the environment, and then adaptively control the parameters

to enhance the surface. Finally, it is abstracted resolution capabilities for random problems.

Specifically, the implementation of this research has theoretical support for the innovation of traditional swarm intelligence methods in terms of statistics, structure and abstraction. As Tenenbaum et al. [3] wrote that the main gaps between intelligent learning algorithm and learning with humans are in terms of statistics, structure, and abstraction. In this thesis, we mainly concentrate our attentions on two kinds of swarm intelligent technologies: one is immunological clonal selection algorithms, and the other is the hybridization of different swarm intelligent algorithms, and their applications.

(1) The use of statistical methods and techniques to model the swarm intelligence method has important theoretical and practical value for improving the efficiency and robust performance of the algorithm in solving complex problems. In the face of complex problems in a dynamic environment, the search space and parameter range of the problem will change with time. At this time, the global and local optimal solutions of the problem are all in motion. Although traditional swarm intelligence technology has a good effect in dealing with static optimization problems, its ability to find optimal solutions in dynamic uncertain environments is greatly reduced because it does not track and locate mobile feasible solutions. The statistical-based clustering methods, such as support vector machines and estimation distribution algorithms, can generate global estimates for the sample space, so that important information of the solution space at the time of change is stored in the form of statistical data. Through the statistical analysis and modeling of the temporary feasible solutions generated in the swarm intelligence algorithm, the targeted tracking and positioning optimization are formed for different specific solution spaces, so as to improve the global optimization ability of the algorithm in the random environment.

(2) The characteristics of the information interaction network formed during the process of population evolution search is analyzed to find out the general rules of general structural features, and then design and enhance the collaborative computing function between groups, including the design and development of memory and

sharing mechanisms, thereby eliminating the assumptions about the independent and identical distribution of the environment. It is to improve the computational efficiency of the algorithm when solving dynamic optimization problems. The advantage of swarm intelligence algorithm lies in the collaborative computing ability between different individuals. In order to further improve the synergy effect, the swarm intelligence under special population structure (such as hierarchical structure, scale-free structure, small world model structure, etc.) should be studied. The algorithm will carry out targeted calculations for dynamic environments with different specificities, and use the memory mechanism in each group to perform regression analysis on the periodically changing environment in a timely manner, and use the sharing and crowding mechanism among groups to effectively control the scale of each group. The regular information interaction is to improve the diversity of the entire group. The remaining issues is the lack of dynamic environment prior knowledge to improve its own computing and application capabilities.

(3) Through the adaptive adjustment of the parameters in the swarm intelligence algorithm, the independent learning of the algorithm is realized, which improves the abstract generalization ability of the system and can better deal with the uncertain conditions or random noise in the dynamic environment. There are some custom parameters in the swarm intelligence algorithm that affect the execution complexity and computational efficiency of the algorithm. By designing the hooks with the changing environmental parameters, the principle of maximizing the performance of the algorithm at different times can be satisfied, and the parameters can be adaptively adjusted. For example, the size of the group should be expanded rapidly in the event of a sudden change in the environment, so as to cope with various optimization details after the mutation, or the scale is gradually adaptively reduced after the environment is balanced and stable, under the premise of not increasing the complexity of the algorithm time. Thus, the optimization ability in the dynamic environment is improved.

Based on the above consideration, in this thesis, I will present several research results of how to applied improved immunological and evolutionary algorithms to

solve very complex optimization problems.

1.1 Research Background

Human beings have been continually enlightened by nature, and through in-depth and meticulous observation of various biological behavior patterns, macroscopic and microphysical phenomena, and social behavioral mechanisms in nature, condensing the laws of learning or synergy, and conducting mathematical modeling, in order to continue to get inspiration for solving complex problems. At present, the swarm intelligence algorithm has achieved certain results in practical engineering applications such as optimization.

Swarm intelligence refers to the characteristics of individuals with simple intelligence that exhibit group intelligence behavior through mutual cooperation and organization, with natural distributed and self-organizing characteristics. Typical swarm intelligence algorithm models are: (1) an ant colony algorithm that simulates pheromone transfer and optimal shortest path in ant collective foraging; (2) a particle swarm optimization proposed by the mechanism of cooperation and competition in the process of mimicking foraging of birds; (3) a bacterial foraging algorithm that mimics the microscopic behavioral characteristics of bacterial chemotaxis, replication, and migration; (4) a gravity algorithm that simulates gravity under gravitation; (5) a colonial competition algorithm that simulates social behavior; (6) an artificial bee colony algorithm for self-organizing collaborative mode such as feeding, collecting honey, nesting, etc, and (7) an immunological clonal selection algorithm mimics the biological adaptive immune response mechanisms, especially the clonal selection theory.

Bonabeau et al. [4] wrote in the magazine NATURE that the swarm intelligence algorithm has broad application prospects in solving complex problems. At present, swarm intelligence algorithms have been successfully applied to optimization problem solving, pattern clustering, biological information, production control, intelligent health monitoring, signal processing. And geophysics and other fields fully demon-

strate the distribution, simplicity and scalability of group intelligence in solving practical complex problems. Because the swarm intelligence algorithm does not have a central control system, it can use simple and efficient individual behaviors to form a robust and flexible solution to tasks through self-organizing operations and indirect communication with individuals.

Although swarm intelligence algorithms have demonstrated superior computing performance in the face of static problems, the development of swarm intelligence has only just begun in the face of dynamic environments. However, due to the large number of complex optimization problems in engineering design, scientific computing, social economy, network communication and other application fields, the environment is actually changing with time. For example, new tasks in production scheduling problems may arrive at any time, and the quality and design requirements of objects in engineering design may suddenly change. The optimization problems in these applications will change with the design variables, objective functions, constraints, etc. of the problem.

At present, swarm intelligence has made initial progress in the face of complex dynamic optimization problems. Literature [5] analyzed the environmental changes by re-estimating the particles in the particle swarm algorithm and performing random sampling to analyze the environmental detection. Literature [6] uses the diversity of groups in the particle swarm optimization algorithm to re-diverge the groups that tend to converge to cope with environmental changes. In [7], the ant colony optimization algorithm based on random, optimal preservation and mixed immigration strategy is used to deal with the dynamic traveling salesman problem. The results show that the stochastic immigration strategy is better than the other two when the environmental change rate is high, and the optimal retention is when the frequency is low. The immigration strategy is better. In [8], through the memory update method of some particles, the memory traceability and variable-scale random initialization strategy are introduced into the phase particle group, and the ability of the memory strategy to respond to the dynamic environment is discussed. By using a variety of group strategies, the literature [9] uses the mechanism of mutual exclusion of parti-

cles to realize the detection of the unknown solution space. Literature [10] explores the ability of group intelligence to search for direct good solutions in dynamic environment from the perspective of individual evolvability. In practical engineering applications, group intelligence for dynamic optimization has also been successfully applied in dynamic scheduling, dynamic sequencing and routing planning, inventory management, dynamic communication, robotics, grinding wheel cutting, Ad Hoc networks, etc. [11–13].

The theoretical analysis of the swarm computing algorithm can be traced back to the genetic algorithm based on Darwin's evolutionary theory. Compared with the simulated annealing algorithm, the genetic algorithm is an early calculation algorithm based on the population group. Therefore, the theory of the relevant theoretical research results on the swarm computing theory. As an important guiding significance, Holland proposed the well-known model theorem [14] (i.e., Schema theory). The pattern theorem is a model with low order, short defined distance and average fitness higher than the average fitness of the population. It grows exponentially in the offspring, which guarantees The number of excellent models (the optimal solution of the population calculation algorithm) increases exponentially, providing a mathematical basis for explaining the mechanism of population calculations (including genetic algorithms). On this basis, Zhou Zhihua [15] and others proposed a new Markov chain-based method to evaluate the expectation of "First-Hitting-Time" in the population algorithm, and it is theoretically analyzed the probability of the algorithm finding the optimal solution. Goldberg and Deb [16] proposed a method of Takeover-Time-Analysis, which can quantitatively compare the selection pressure between selection strategies in evolutionary populations. Prugel-Bennett and Shapiro [17] proposed a statistical physical analysis method to study the evolutionary process of population fitness distribution. Wolpert and Macready [18] proposed and proved the famous NFLT (No-Free-Lunch-Theory) theorem, pointing out that the general algorithm can not achieve the highest optimization efficiency on all problems, and needs to integrate the specific knowledge related to the problem to improve the specificity and the computational efficiency of the algorithm.

Although up to now, a number of theoretical analysis methods for swarm intelligence calculation algorithms have been proposed to study the convergence of algorithms [19], performance evaluation [20], development and balance in search [21], etc. Relying on specific choices, mutations, and cross-operations in specific algorithms does not form a more unified theoretical research framework. In recent years, more scholars have begun to analyze the characteristics of the algorithm from another perspective, that is, the information interaction network formed in the algorithm search process, thus avoiding the dependence on the specific implementation of the algorithm. Liu Jing [22] et al. proposed a predictive method based on the information interaction network called Motif to judge the difficulty of the population evolution algorithm to solve the problem.

The development of swarm intelligence algorithms for dynamic environment optimization problems shows that it has carried out new research in theoretical concepts, computer science, integrated technology and popularization applications. Among them, the research on the strategy of dealing with the dynamic environment and the construction of the corresponding algorithm model are particularly important. Some problems need to be combined with other technologies to find a solution. Therefore, swarm intelligence computing has the following research trends in dealing with dynamic optimization problems:

(1) The use of statistical methods that are not sensitive to environmental changes to model cluster intelligence will become the mainstream research direction for group intelligence to deal with uncertain environmental changes. Regardless of the method used to detect the externally unknown computing environment, the disadvantage is that the detection methods are both passive and lagging [5]. The swarm intelligence algorithm needs to be directed to the changing frequency of the environment, the intensity of the change, the predictability of the change, and cyclically varying cycle lengths and accuracy are used to develop different detection mechanisms, and even to determine the necessity of changing the algorithm's coding. The integration of statistical methods and the use of efficient population structures (such as hierarchical, distributed, scale-free, etc.) can well avoid the above shortcomings. However,

the use of statistical methods to integrate group intelligence and how to effectively establish a hybrid computing model is one of the new challenges to further improve the application scope and efficiency of swarm intelligence.

(2) Two aspects which includes further maintaining the ability of the swarm intelligence algorithm to adapt to environmental changes and improving the addressing ability of the algorithm to the optimal solution after the change is an innovative drive to solve the dynamic optimization problem. Traditional swarm intelligence algorithms deal with dynamic environment changes by maintaining and increasing population diversity strategies [6, 7], memory strategies [8], multi-group strategies [9], and enhancing individual evolution [10]. None of these strategies can dominate, that is, any strategy has its own advantages and disadvantages. For example, based on the strategy of increasing group diversity, after detecting changes in the environment, it temporarily increases diversity by restarting techniques such as search and excessive mutation, and improves the global search ability of the algorithm. The disadvantage of this strategy is that the difficulty of monitoring changes, the blindness of processes that increase diversity, and the difficulty of transmitting and measuring information between old and new populations. Therefore, how to proceed from the model structure and self-adaptation of the swarm intelligence algorithm itself, continue to explore the computational strategies that can improve the response to complex environmental changes, is the second challenge to realize the computational efficiency of swarm intelligence itself.

In summary, swarm intelligence uses distributed individuals to solve problems by simulating biological natural and social phenomena and mechanisms, and then uses the characteristics of group collaboration to improve algorithms to solve the performance of various complex problems. Practical engineering applications provide a broad development prospect. Based on the above research background, we mainly developed two kinds of swarm intelligent algorithms, including two immunological clonal selection algorithms, and two kinds of hybridization of different single swarm intelligent algorithms. The performance of these improved algorithms are verified on several optimization problems, arising from numerical optimization problems or

combinatorial optimization problems, to complex resource scheduling optimization in cloud computing.

1.2 Organization of the Thesis

In Chapter 1, several basic theories and research background, research motivation, and ideas are introduced. In Chapter 2, an improved quantum immunological algorithm for optimization is presented. The greatly improved algorithm is applied on two important combinatorial optimization problems, including traveling salesman problem and holes machining path planning problem. Experimental results and performance comparison suggested that the proposed method is effective and efficient. In Chapter 3, I elaborate a novel clonal selection algorithm for resource scheduling optimization problem in cloud computing. An improvement operation with vaccine injection is designed to enhance the diversity of solutions. On the other hand, Gauss mutation is used to improve ability to escape from local optima. Moreover, the effects of the proposed algorithm are analyzed and evaluated by comparison with other resource scheduling methods by simulation toolkit - CloudSim. The comparative results show that the proposed algorithm outperforms these algorithms in terms of execution time. In Chapter 4, a hybrid ant lion differential evolution for optimization is presented. In Chapter 5, a novel hypercube and spherical evolution for optimization is depicted. In chapter 6, some general remarks and future works are given.

Chapter 2

Improved Quantum Clonal Selection Algorithm

2.1 Introduction

A great deal of real-world problems, such as job shop scheduling problem [23], frequency assignment problem [24], machining parameter selection problem [25], and structural topology configuration [26], can be treated as optimization problems (OPs).Various attempts from different perspectives have been made to resolve these optimization problems.

Traditional methods such as steepest decent, linear programming, dynamic programming, etc. commonly are indeed producing some satisfactory results under certain conditions. However, these conventional methods confirm certain negative situations according to the powerful constraints on the objective function, continuity, differentiable, unimodal and so forth [27]. In addition, these conventional methods have been facing common problems such as the premature convergence.

As we know, many engineering problems generally are complicated with a high-dimensional space, multiobjective and multiconstraint functions and difficult to be solved by traditional methods. Thus, efficient and effective optimization algorithms are still necessary to be proposed.

In the last few decades, substantial operational researchers have proposed a wide variety of approaches to handle complex optimization problems. However, it was

until relatively recently that researchers utilized evolutionary algorithms (EAs) and other population-based heuristics for their global optimization, parallelism, and effectiveness [28]. These novel methods have the distinctive advantage for resolving multiobjective and multiconstraint problems that traditional approaches have failed to tackle.

Among population-based optimization methods, genetic algorithm (GA), an bio-inspired intelligent model of Darwinian evolution based on the mechanism composed of genetic mutation and natural selection, was first proposed by Holland. Different genetic algorithms have been successfully used to a extensive filed of problems including control [29], optimization problems [30], 3D metamaterial design [31] and software test data generation [32], [33].

Although GAs have broad applicability and great potential for global search, they have to face some problems such as premature convergence and a lack of local search abilities. A number of improved genetic algorithms and hybrid approaches have been proposed to overcome these problems [34], [35], [36].

Besides, some other bio-inspired methods such as particle swarm optimization (PSO) which depends on the foraging behavior of bird swarms [37], artificial bee colony (ABC) which simulates intelligent behaviors of honey bee swarms [38], and ant colony optimization (ACO) which emulates the foraging behavior of ant colonies [39] have also been thoroughly studied and successfully applied in many places.

Apart from the above mentioned methods, there has been an increasing interest in researching the immune systems to develop new methods for addressing different engineering problems in recent years. Artificial immune system (AIS), one of the most attractive nature inspired computation models, has been proposed to deal with different problems [40]. This emerging computational intelligence simulates the learning and adaptive mechanisms of living organisms in protecting themselves against antigens. Numerous artificial immune systems and their modified versions, such as negative selection algorithm [41], and immune network theory based algorithm [42], have been proposed to apply for both theoretical and practical problems [43], [44], [45], [40]. Particularly, clonal selection algorithm (CSA) [46] which is based on the clonal selec-

tion principle proposed by Burnet [47] has attracted more and more interest and has been validated as a powerful method to optimization problems [48], [49], [50], [51].

Although CSA provides an alternative approach for problems that are difficult to be handled by traditional optimization methods, it has to undergo a fact of falling into local minima but not global optimum for complex problems. This circumstance called premature convergence takes place when the population of CSA reaches such a local minimum state that the immune operators can not generate new solutions with a high affinity (degree of match) anymore [52]. To deal with the premature convergence problem, several receptor editing based CSAs have been proposed by us or other researchers [53], [54], [55], [56].

As mentioned above, sufficient diversity is one of the essential features of adaptive immune responses. This is implemented by the distinctive structure of the immunoglobulin molecule, which consists of two chains, i.e., heavy and light chains [57]. At the heavy chain locus, receptor editing takes place mainly by the deletion of intervening gene segments. At the light chain locus, receptor editing can take place by either deletion or inversion of the intervening gene sequences. However, almost all receptor editing based on immune algorithms have ignored the deletion operation.

In this paper, a complete receptor editing based on clonal selection algorithm is proposed. The novel algorithm with complete receptor editing will cover the solution space more efficiently and is less likely to be trapped on local optima. Moreover, a quantum interference crossover is also embedded for exchanging knowledge among different solutions.

The rest of this paper is structured as follows. Section 2.2 gives a brief overview of the immune response, clonal selection theory, and receptor editing. Section 2.3 introduces the novel clonal selection algorithm in detail. Traveling salesman problems and holes machining path planning problems are briefly introduced and then solved to verify the performance of the novel approach in Section 2.4. Section 2.5 draws conclusions.

2.2 Natural Immune System

As described above, artificial immune system can be defined as a computational model based on the biological immune system. In order to develop a new artificial immune system for optimization problems, the natural immune system needs to be explained.

2.2.1 Natural Immune Response and Clonal Selection Theory

Immune system which is a highly parallel, evolved and distributed adaptive system protects the body against bacteria and viruses [58]. This system can automatically adapt or learn to discern different antigens, even those never met before. Anything that can trigger this immune response is called an antigen (Ag). From a pattern recognition perspective, the most attractive attribute of the immune system is the existence of receptor molecules on the surface of immune cells, which can discern the nearly limitless range of antigenic patterns. Two major immune cells, i.e., B cells and T cells, can be recognized by the receptor molecules. When the structures of an immune cell receptor and an antigen are complementary, the antigen is discerned by the immune cell. The more complementary the structures are, the higher the affinity between the immune cell and the antigen is [45].

Clonal selection theory proposed by Burnet [47] explains the essential properties of immune system including adequate diversity, discrimination of self and non-self, and sustaining immunologic memory. In essence, the clonal selection theory is a pattern of natural selection. The clonal selection principle is shown in Fig. 2.1 and the details are described as follows.

When an antigen occurs in an immune system, it will impose a selective pressure on the antibody (Ab) population. That is to say, only those immune cells which specifically discern the antigen are chosen to proliferate and differentiate. We call these immune cells high affinity cells. In the following selection stage, immune cells with high affinity to the antigen are activated, and then stimulated to proliferate generating numerous clones. Finally, these clones can mutate or become plasma cells

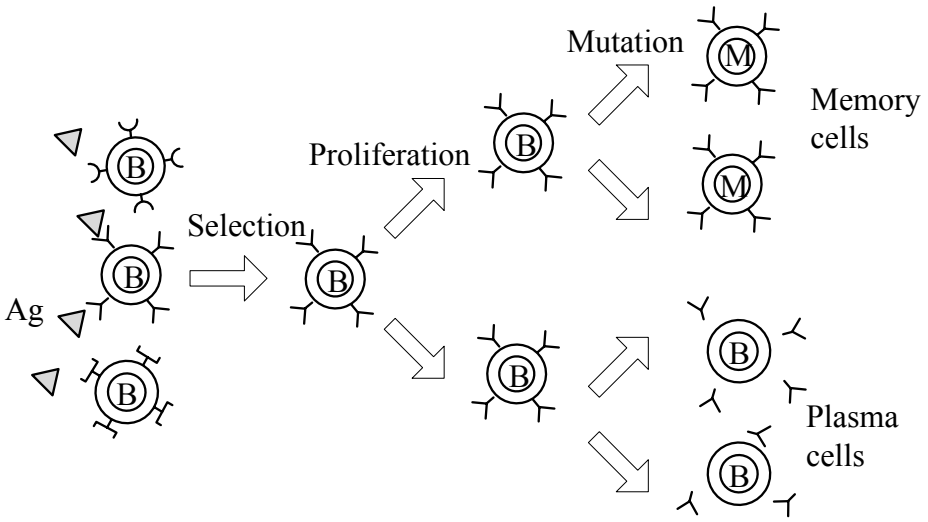


Figure 2.1: Illustration of the clonal selection process.

which release a great deal of antibodies or memory cells that maintain the antigenic status for future infections.

2.2.2 Receptor Editing

On the basis of the clonal selection theory, immune cells with low affinity with antigen do not divide and will be discarded or deleted. However, recent investigations suggest that clonal deletion, formerly regarded as one of the most important mechanisms of clonal selection theory, handle next and only when receptor editing (RE) can not offer a high affinity specificity [59], [60], [61].

As noted former, diversity is one important feature of adaptive immune system. It is implemented by the distinctive structure of antibody molecule, which contains heavy chain and light chain formed by the rearrangement of diverse gene segments. Fig. 2.2 shows the Y structure of an antibody receptor molecule.

As can be seen in Fig. 2.2, the antibody molecule consists of two light (L) chains and two heavy (H) chains. Each chain contains the variable (V) region and constant (C) region. The V region (VH, VL) is mainly in charge of antigen recognition and contains special variable subregions where the residues have contacted with the actual antigen. The C region (CH, CL) is in charge of diverse effector functions and defines

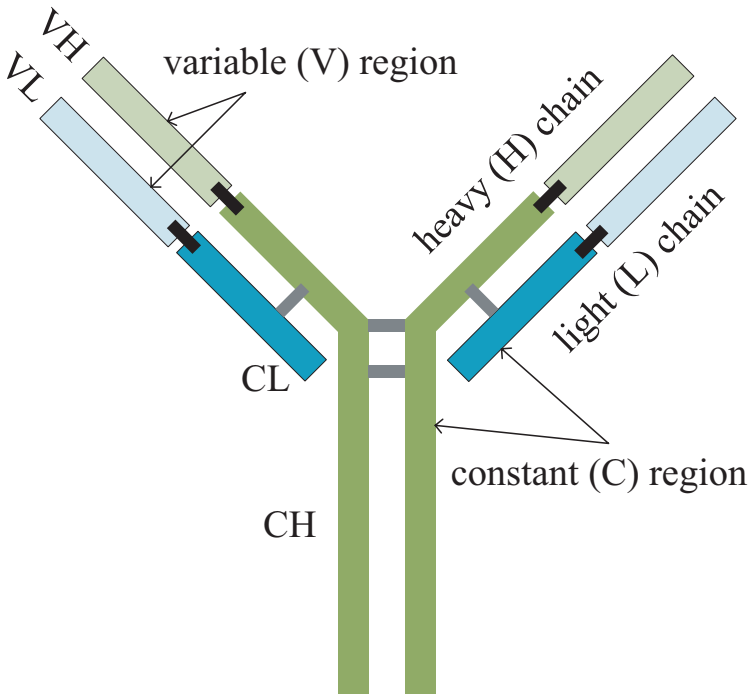


Figure 2.2: Y-type structure of antibody molecule.

the chain types or classes [62].

In the elementary rearrangement, the heavy chain locus is realigned to generate a VDJ segment whose locus is composed of numerous variable (V), diversity (D) and junction (J) segments. The VDJ is generated by the combination between a V segment and the assembled DJ. In the secondary rearrangement, B lymphocyte tolerance is governed by the light chain where only V and J segments are assembled to generate VJ segment. Thereafter, the upstream V segments can be further combined with the downstream J segment so as to delete or replace the previous VJ segment. Moreover, substantial V regions in reverse direction of the chromosome are realigned by the inversion instead of the deletion in the intervening sequences, thus retaining the V segments [61], [57].

Fig. 2.3 indicates the different arrangements on heavy and light chains.

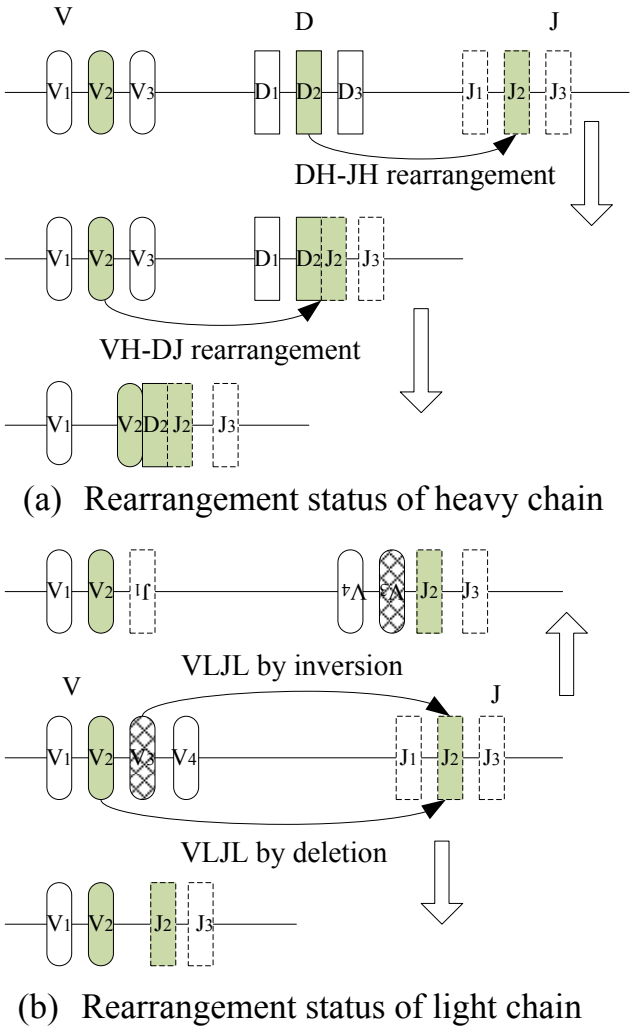


Figure 2.3: Gene rearrangement of receptor editing on the heavy chain (a) and light chain (b).

2.3 Complete Receptor Editing Based on Quantum Clonal Selection Algorithm

In this section, the proposed complete receptor editing based on novel clonal selection algorithm (CRECSA) is presented. The initial idea of CRECSA is inspired by the natural immune system where the purpose is to efficiently eliminate the antigens. The evolutionary process of CRECSA is composed of initialization, affinity evaluation, clone operator, hypermutation, receptor editing and quantum crossover operation. The algorithm will evolve over iterations and not be ended until the termination

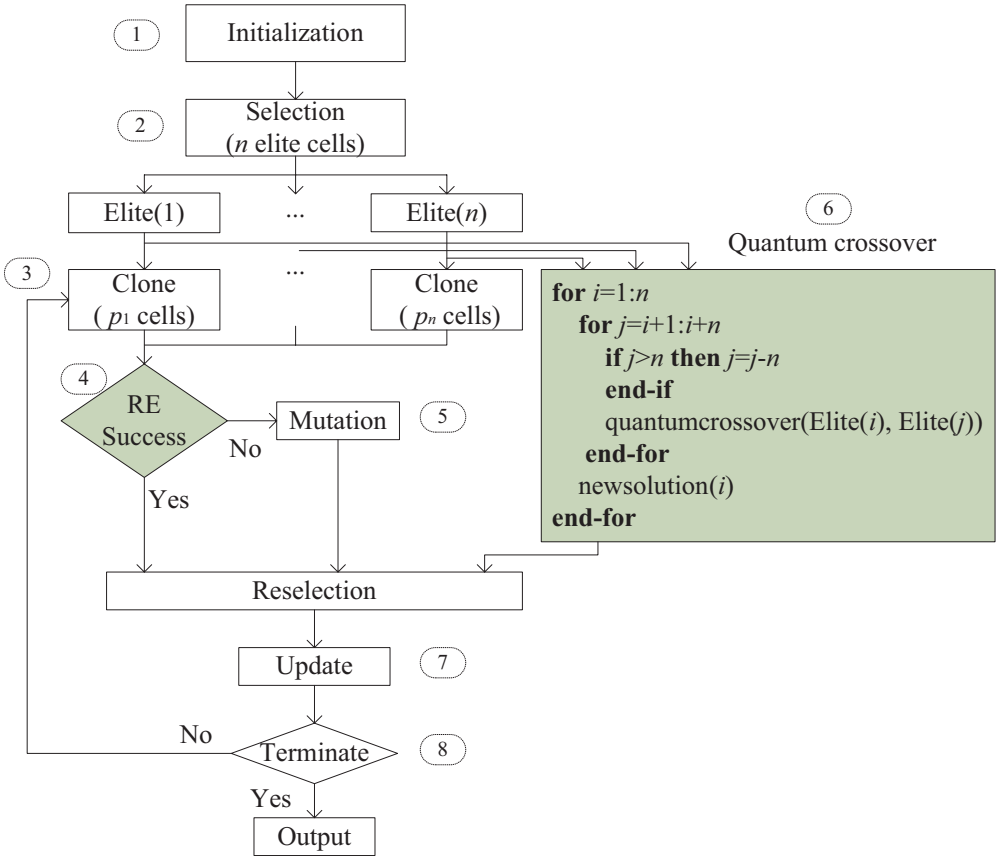


Figure 2.4: The flowcharts of the proposed algorithm CRECSA.

condition is satisfied.

Like other CSAs, the proposed CRECSA is also based on CLONALG, which is proposed by Castro et al. [63]. The flowcharts of novel CRECSA are illustrated in Fig. 2.4 in a comparative perspective.

Since each step of CLONALG is included in CRECSA, we just describe the procedure of CRECSA as follows:

Step 1 Generate an initial population S including m cells randomly $(S_1, S_2, ..., S_m)$ and give the termination criterion.

The termination criteria can be determined by the limited CPU time, the number of generations, or the number of consecutive iterations without the enhancement of the best solution. In this paper, a maximum generation number G_{max} is used as the termination criteria for fair comparison with other algorithms.

Step 2 Compute individual affinity $A(S_1)$, $A(S_2)$, ..., $A(S_m)$ and then rank these

cells via a descending order, where $A(.)$ is a affinity computation function. Then choose n ($n \leq m$) high affinity antibodies as elite cells, which depends on their affinities from m original antibodies. And place each elite in n separate and different elite pools ($EP_1, EP_2, ..., EP_n$).

Step 3 Duplicate the elites in each elite solution pool with a rate proportional to its fitness as Eq.(2.1):

$$p_i = round(\frac{(n-i)}{n} \times M), \tag{2.1}$$

where i indicates the ordinal of elite pools and M denotes a multiplying factor. $round(.)$ is to round its argument towards the closest integer.

Step 4 Change the clones in each pool by the complete receptor editing. According to the shape-space model [64], a group of coordinates in a N-dimensional shape-space can embody an antigen or an immune cell. Thus, the receptor gene sequence of immune cell can be formulated as $S = (s_1, s_2, ..., s_N)$. The receptor editing process is described as follows:

$$s_1... \rightarrow s_{i-1} \rightarrow \mathbf{s_i} \rightarrow \mathbf{s_j} \rightarrow s_{j+1}... \rightarrow s_N$$

$$\Uparrow$$

Deletion operation

$$\Uparrow$$

$$s_1... \rightarrow s_{i-1} \rightarrow \mathbf{s_i} \rightarrow s_{i+1}...s_{j-1} \rightarrow \mathbf{s_j} \rightarrow s_{j+1}... \rightarrow s_N$$

$$\Downarrow$$

Inversion rearrangement

$$(\overrightarrow{\mathbf{s_i} \rightarrow s_{i+1}...s_{j-1}} \Rightarrow \overrightarrow{s_{j-1}...s_{i+1} \rightarrow \mathbf{s_i}})$$

$$\Downarrow$$

$$s_1... \rightarrow s_{i-1} \rightarrow s_{j-1}...s_{i+1} \rightarrow \mathbf{s_i} \rightarrow \mathbf{s_j} \rightarrow s_{j+1}... \rightarrow s_N$$

It should be noticed that the algorithm in this paper is proposed to solve traveling salesman problems (TSP) and holes machining path planning problems which can be transformed into TSP. The solution of TSP is encoded in a permutation of cities. The deletion operator destroys a closed feasible solution. Consequently, almost all

receptor editing based immune algorithms ignored the deletion operation. However, the ability of receptor editing to salvage from deletion low affinity immune cells via changing their receptor specificity realized the clonal selection process anew [57]. By simulating the recombination mechanism in nature immune system, a rearrangement operation based receptor deletion editing is proposed. Suppose the solution sequence is $S = (s_1, s_2, ..., s_N)$. Two points x and y are selected randomly. Algorithm 1 shows the pseudo code of rearrangement operation.

begin
  Input x, y
  d_{sum}=0,p_{selSum}=0,m_{sel}=0
  for i=$x : y$ **do**
    d_{max}=argmax$\{cityDis(s_i, s_z),\ z = i+1, i+2, ..., y\}$
    d_{sum}=$\sum_{z=i+1}^{y} cityDis(s_i, s_z)$
    for j=*i+1:y* **do**
      $p_{sel}[j]$=$\frac{d_{max} - cityDis(s_i, s_j)}{d_{sum}}$
      p_{selSum}+=$p_{sel}[j]$
    end
    s_{rand}=$rnd(0, P_{selSum})$
    for j =*i+1:y* **do**
      m_{sel}+=p_{sel}[j]
      if $m_{sel} \geq s_{rand}$ **then**
        $cityI$=j
        break
      end
    end
    $s_{i+1} = s_{cityI}$
  end
end

Algorithm 1: Deletion rearrangement operation

Here, $cityDis(s_x, s_z)$ returns the distance between city s_x and city s_z. d_{max} means the maximal distance between city s_x and other cities in sub-segment. d_{sum} means the sum of distance between city s_x and other cities in sub-segment. $p_{sel}[j]$ is the probability of city s_j to be selected. $rnd(0, p_{selsum})$ is a random function which returns a value between 0 and p_{selsum}. From Algorithm 1, the city close to the preselected city will be selected with a higher probability.

In this step, inversion rearrangement operation or deletion rearrangement opera-

tion is randomly selected according to the threshold probability p_s.

Without loss of generality, we suppose that $s_1, s_2, ..., s_i, s_{i+1}, ..., s_j, s_{j+1}, ..., s_N$ is the original solution, $s_1, ..., s_{i-1}, s_{x1}, s_{x2}, ..., s_{i-j+1}, s_{j+1}, ..., s_N$ is the new solution generated by receptor editing operation.

If $cityDis(s_{i-1}, s_i)+cityDis(s_i, s_{i+1})+...+cityDis(s_{j-1}, s_j)$

$+cityDis(s_j, s_{j+1}) > cityDis(s_{i-1}, s_{x1})+cityDis(s_{x1}, s_{x2})$

$+...+cityDis(s_{j-i}, s_{j-i+1})+cityDis(s_{j-i+1}, s_{j+1})$, we can confirm the receptor editing process is success.

Step 5 Subject the clones which fail at the receptor editing process through random point mutation. This process can be illustrated as follows:

$$s_1 \rightarrow s_2... \rightarrow \mathbf{s_i} \rightarrow s_{i+1}... \rightarrow \mathbf{s_j} \rightarrow s_{j+1}... \rightarrow s_N$$

$$\Downarrow$$

$$s_1 \rightarrow s_2... \rightarrow \mathbf{s_j} \rightarrow s_{i+1}... \rightarrow \mathbf{s_i} \rightarrow s_{j+1}... \rightarrow s_N$$

Step 6 In order to enhance the information exchange ability among different solutions, an improved quantum crossover operator is added into this CSA. Subject n selected original solutions $(S_1, S_2, ..., S_n)$ through improved quantum crossover operation. Through this operation, we will get n new solutions. The pseudo code of quantum crossover operation is shown as Algorithm 2. Because every city in the new solution comes from different original solutions, we need to determine the candidate city according to Algorithm 2. In this algorithm, the parameter *pos* is the index of candidate solution. $cityName(x, j)$ is a function which returns the city name of jth city in S_x. $cityIndex(x, CN)$ returns the index of preselected city CN in S_x. $cityDis(s_x, s_y)$ returns the distance between city s_x and city s_y. LCI and RCI are the index of the left and right city of CN in candidate solution. LC and RC are the left and right neighbour city of CN in candidate solution.

For clarity, a simple TSP instance including A, B, C, D, E, F six cities is used for explaining the quantum crossover. Six solutions are:

$S_1 : A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow F \rightarrow A$

$S_2 : C \rightarrow E \rightarrow A \rightarrow B \rightarrow F \rightarrow D \rightarrow C$

$S_3 : B \rightarrow F \rightarrow D \rightarrow C \rightarrow A \rightarrow E \rightarrow B$

$S_4 : A \to C \to E \to F \to D \to B \to A$

$S_5 : B \to D \to F \to E \to C \to A \to B$

$S_6 : F \to B \to A \to C \to E \to D \to F$

```
begin
    for i=1:N do
        for j=1:N do
            if j==1 then
                CN=cityName(i+j-1,j)
            end
            else
                pos=(i+j > N+1)?(i+j-(N+1)):(i+j-1)
                LCI=cityIndex(pos, CN)-1
                RCI=cityIndex(pos, CN)+1
                LC=cityName(pos, LCI)
                RC=cityName(pos, RCI)
                if cityDis(CN, LC) ≤ cityDis(CN, RC) then
                    CN = LC
                end
                else
                    CN = RC
                end
            end
            sQC[i][j]=CN
        end
    end
end
```

Algorithm 2: Improved quantum interference crossover operation

Fig. 2.5 demonstrates the construction procedure of a quantum crossover. The following describes an example of the process of how a new crossover solution is constructed.

(1) Without loss of generality, city B is selected from solution S_3. Then jump to S_4, two neighbour cities D and A are candidate cities. If $cityDis(B, A) < cityDis(B, D)$, city A is selected. The new solution under construction is $B \to A$.

(2) Then jump to S_5, C and B are two neighbour cities of A. If $cityDis(C, A) < cityDis(A, B)$, city C is selected. The new solution under construction is $B \to A \to C$.

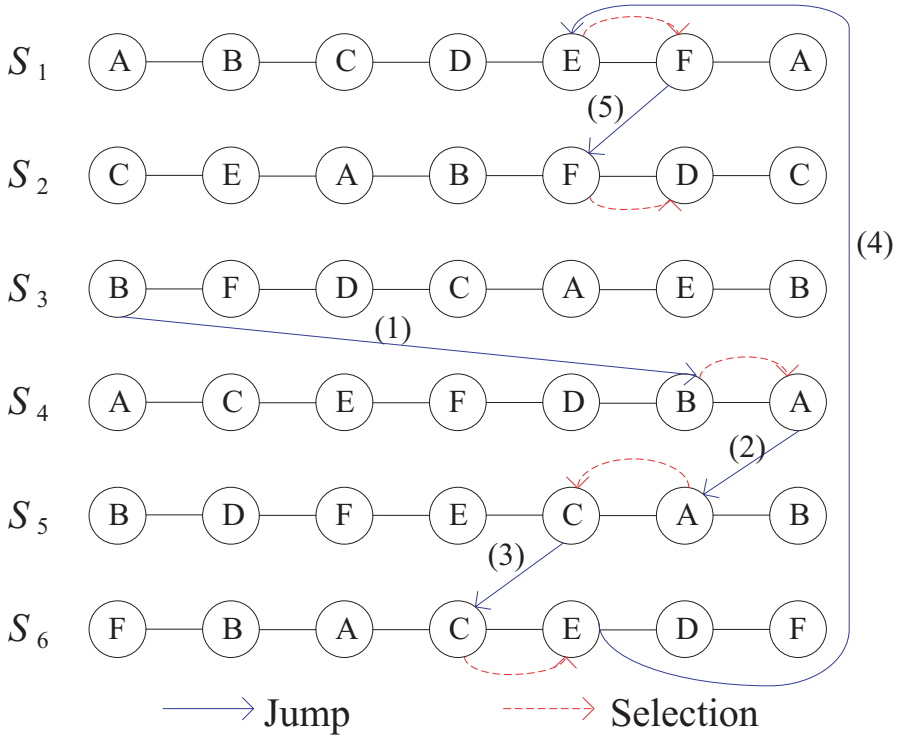


Figure 2.5: Illustration of construction process of quantum interference crossover solution.

(3) Then jump to S_6, A and E are two neighbour cities of C. If $cityDis(C, E) < cityDis(A, E)$, city E is selected. The new solution under construction is $B \to A \to C \to E$.

This way, a new solution will be constructed step by step. If a city already appears in the new solution under manipulation, it will be replaced by the closest city which is not used in current solution.

Step 7 Select the best cell from each elite pool and solutions generated by the quantum crossover and then update current elite pool with the best solutions.

Step 8 If the generation number reaches the maximum generation number G_{max}, the algorithm is ended. Otherwise, return to Step 3 for next iteration.

2.4 Simulation and Discussion

In this section, in order to evaluate the performance of CRECSA, computational experiments are conducted to optimize two benchmark problems containing TSP and holes machining path planning problem. In the following subsections, these problems are firstly described. Then, the parameter settings for CRECSA are discussed. Simulation results are presented finally. The CRECSA is programmed by C++ and all the computational results demonstrated in the following figures and tables are obtained over 10 runs unless there is a special statement.

2.4.1 Traveling Salesman Problem (TSP)

TSP, one of the most widely discussed combinatorial optimization problems, is a famous NP-hard problem. Generally, the optimal solution of TSP is a minimum Hamiltonian path. Thus, various practical problems such as intelligent transportation, flow shop scheduling, and logistics, can be regarded as TSP or its variants. As a result, TSP attracts many attention from the scientific and industrial fields through the years.

Generally, the TSP can be described as follows: an optimal route is constructed via visiting n cities only once and finally returning to the initial point. It is assumed that C_i represents the ith city in a route. $\pi = \{C_1, C_2, ..., C_i, ..., C_j, ..., C_N\}$ $(1 \leq i \leq j \leq N)$ indicates a feasible route containing all the visited cities. Therefore, a typical TSP can be expressed mathematically as follow [65]:

$$\begin{cases} \min f(\pi) & = & \sum_{i=1}^{N-1} cityDis(C_i, C_{i+1}) \\ & & +cityDis(C_1, C_N) \\ s.t. \;\; C_i & \neq & C_j (i \neq j) \\ C_i & \in & \{1, 2, ..., N\}, \end{cases} \tag{2.2}$$

where N indicates the total number of cities. $f(\pi)$ denotes the path length of a feasible route π. If $cityDis(C_i, C_j) = cityDis(C_j, C_i)$, the problem is called a symmetric TSP

Table 2.1: TSP instances used in simulation.

Problem	City number	Optimum	G_{max}
eil51	51	426	1000
st70	70	675	1000
eil76	76	538	1000
rd100	100	7910	1000
eil101	101	629	1000
lin105	105	14379	1000
pr107	107	44303	1000
pr124	124	59030	1000
bier127	127	118282	2000
pr136	136	96772	2000
pr152	152	73682	2000
rat195	195	2323	2000
kroA200	200	29368	5000
lin318	318	42029	10000

Table 2.2: Parameters setup in simulation.

Parameter	Meaning	Values
N	city number	51∼318
m	number of initial antibodies	N
n	number of elite pools	N
M	proliferation rate	50
p_s	threshold probability of revision and deletion rearrangement	0.5
G_{max}	maximum number of generation	*

*:see Table 2.1

(STSP). If $cityDis(C_i, C_j) \neq cityDis(C_j, C_i)$, the problem is called an asymmetric TSP (ATSP). In this paper, all tested traveling salesman problems are STSP and available at TSPLIB [66]. Table 2.1 lists the specific results of several TSP instances. Meaning and value of each parameter used in simulation are given in Table 2.2. In this table, the value 0.5 of p_s means inversion or deletion rearrangement opeartion has the same probability to be selected during Step 4.

2.4.2 Simulation Results of TSP Instances

In terms of the structure of algorithm, the proposed method is an improved version of the receptor editing based on clonal selection algorithm RECSA [53]. According to

Table 2.3: Comparison results of the three algorithms RECSA, RECSA+IQC, and RECSA+DEL on TSP instances.

TSP	RECSA		RECSA+IQC		RECSA+DEL	
Instance	*PDB*	*PDM*	*PDB*	*PDM*	*PDB*	*PDM*
eil51	1.41	2.63	1.17	2.16	0.70	1.27
eil76	3.35	4.54	1.86	3.09	1.67	2.84
rd100	4.55	5.85	2.40	3.30	2.34	3.53
pr107	2.11	3.40	2.44	2.77	1.57	2.44
pr124	1.65	4.42	0.64	1.64	0.94	2.97

RECSA, the improved quantum crossover IQC is added to improve the global search performance through exchanging information between different solutions. Deletion operation DEL, one of the receptor editing methods, is embedded into RECSA to further boost the local search ability. Table 2.3 shows the experimental results of three algorithms RECSA, RECSA+IQC, and RECSA+DEL.

As shown in Table 2.3, the improved quantum crossover IQC and the deletion receptor editing operation DEL can enhance the performance of RECSA. In this table, *PDM* and *PDB* indicate the percentage deviation of the average result D_m and the best result D_b over the optimal solution D_{opt} respectively. Here, D_m is the average distance of all 10 trials, and D_b is the shortest distance among 10 runs.

$$PDM = \frac{D_m - D_{opt}}{D_{opt}} \times 100, \tag{2.3}$$

$$PDB = \frac{D_b - D_{opt}}{D_{opt}} \times 100. \tag{2.4}$$

Like other evolutionary algorithms, population diversity plays a very important role in clonal selection algorithms. In this kind of algorithm, the diversity is represented by the average edge-distance between the best solution and other solutions. Edge-distance indicates diverse edges between two solutions [67].

The average population diversity of each generation of four algorithms RECSA, RECSA+IQC, RECSA+DEL, and CRECSA for eil51 problem are shown in Fig. 2.6.

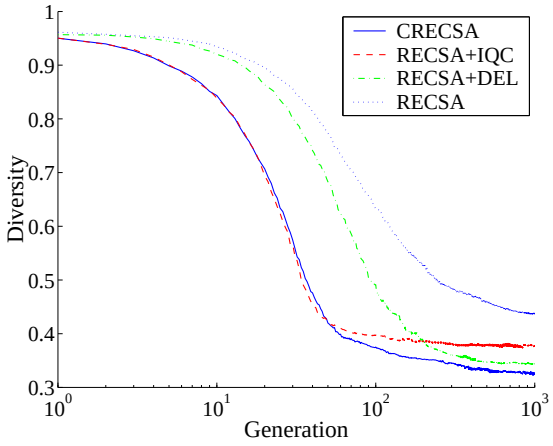
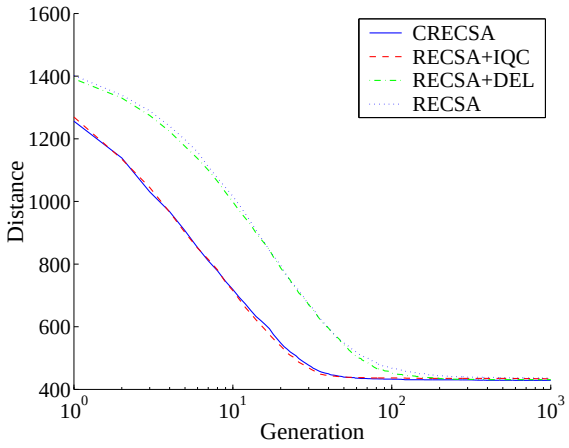


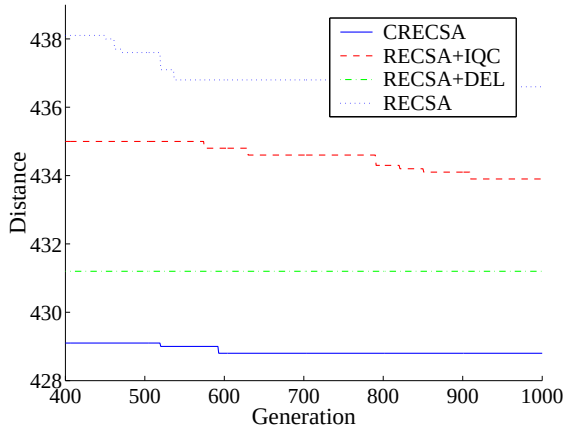
Figure 2.6: Comparison of population diversity of the four algorithms for eil51 problem.

From Fig. 2.6, the diversity of the proposed algorithm is lower than others. The usual view is that higher diversity is beneficial for avoiding premature convergence and escaping from local optima of algorithm [68]. However, based on our experimental results, the diversity is not the only key to the performance of algorithm. It seems that a proper control of balancing the exploration and exploitation process of algorithm is needed. In general, a good population diversity is helpful for exploring the search space widely, and a low diversity leads to the convergence of algorithm. Consequently, the algorithm should possess a higher diversity in the beginning of the searching process and a lower one in the end. As Fig. 2.6 shown, the proposed algorithm possesses equivalent diversity maintenance ability at the beginning of searching process and a lower diversity at the end stage. That is to say, the proposed algorithm maintains a higher population diversity at the initial stage to explore the search space widely, and a lower diversity easy to converge to the optimal value at the end stage.

Fig. 2.7 illustrates the convergence process of RECSA, RECSA+IQC, RECSA+DEL, and CRECSA. The distance in Figs. 2.7 and 2.8 is the average distance of each generation of 10 runs. From Fig. 2.6 and Fig. 2.7, it can be confirmed that the CRECSA has better performance of balancing the exploration search and exploitation search than other methods. The same conclusion can be drawn from another TSP instance of eil76 in Fig. 2.8.



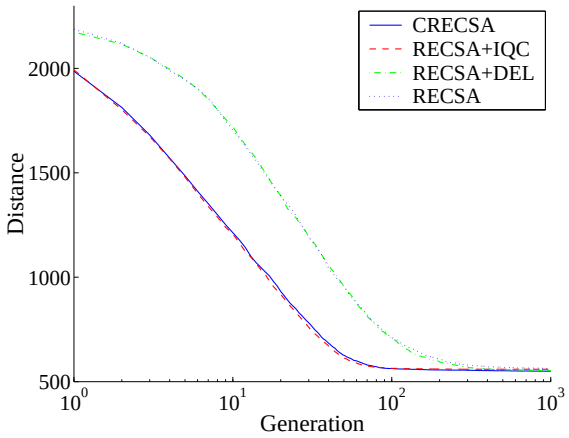
(a)



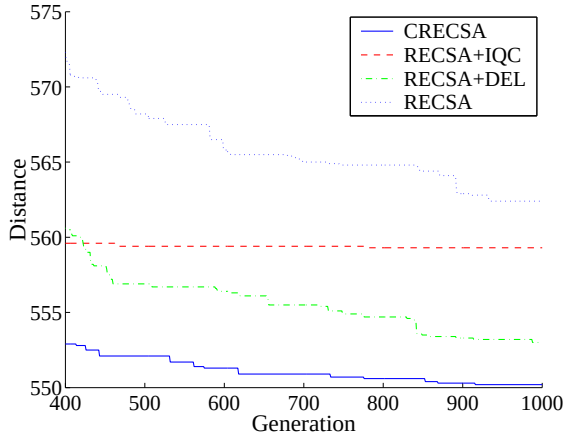
(b)

Figure 2.7: (a) Comparison of convergence process of the four algorithms for eil51 problem (b) the last 600 generations of the convergence process.

In order to further confirm the performance of the CRECSA, we apply our method for TSP instances with the city number from 51 to 318 and also compare CRECSA with other CSAs [53]. Table 2.4 shows the experimental results. t is CPU time. From this table, it can be confirmed that the proposed algorithm CRECSA outperforms RECSA, RECSA+IQC, and RECSA+DEL. RECSA indicates clonal selection algorithm including receptor inversion editing operation. RECSA+IQC manifests the improved quantum crossover based on RECSA [53]. RECSA+DEL denotes the RECSA combined with receptor deletion editing operation. The computational results of RECSA+IQC are recalculated by the same hardware PC so as to make fair comparison.



(a) convergence process



(b) last 600 generations convergence process

Figure 2.8: (a) Comparison of convergence process of the four algorithms for eil76 problem (b) the last 600 generations of the convergence process.

2.4.3 Holes Machining Path Planning (HMPP) Problem

Most of the earlier algorithms of AISs were used for solving benchmark cases. However, the number of resolving engineering optimization problems such as I-beam design problem [55], structural layout problem [25] has been increasing lately. HMPP is a crucial mission for machining process planning of multi-hole parts.

HMPP aims to minimize the overall machining costs, including the moving cost of tools (C_{M}), cutting tools cost(C_{T}), and tools replacement cost (C_{R}) respectively. The mathematical formulation of the case is as follows [69].

$$C_{\mathrm{M}} = \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} \eta \cdot D_{ij} \cdot e_{ij}, \tag{2.5}$$

where η is the moving cost of the cutting tools for unit length. $e_{ij} \in \{0, 1\}$. $e_{ij} = 1$ denotes the edge(i, j) is in the optimization solution.

$$C_{\mathrm{R}} = \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} \lambda \cdot p_{ij} \cdot e_{ij}, \tag{2.6}$$

where p_{ij} is the tools replacement time from point i to point j and λ is the replacement cost in unit time.

The tool cost (C_{T}) is defined as follow:

$$C_{\mathrm{T}} = \sum_{k=1}^{N_{\mathrm{T}}} C_{\mathrm{T}k},$$

$$C_{\mathrm{T}k} = \sum_{i=1}^{N} \left(\frac{t_{ki}}{T_{ki}} Q_k + c \cdot t_{ki} \right), \tag{2.7}$$

where $k \in \{0, 1, 2, ..., N_{\mathrm{T}}\}$, N_{T} is the number of tools which to be used for machining N holes, t_{ki} is the cutting time of tool k when machining the hole i, T_{ki} is the life-span of tool k machining the hole i, c is the machining cost for unit time, and Q_k is the economic cost of the cutting tool k.

Obviously, the objective of HMPP is to minimize the overall machining costs as follow:

$$\min(C_{\mathrm{M}} + C_{\mathrm{R}} + C_{\mathrm{T}}). \tag{2.8}$$

2.4.4 Computational Results of HMPPs

In order to compare the proposed algorithm with other methods fairly, the aim in experimental simulation is to minimize the moving cost of the machining tools. Fig. 2.9 shows the dimensional drawing of the part with 26 holes. The coordinates of holes

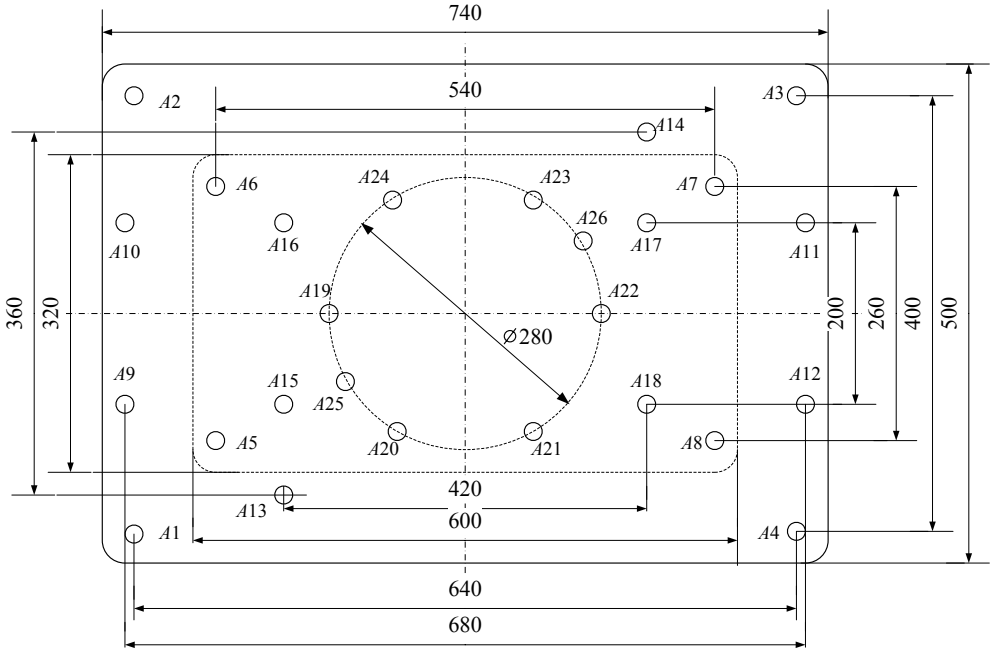


Figure 2.9: Illustration of 26-hole machining part.

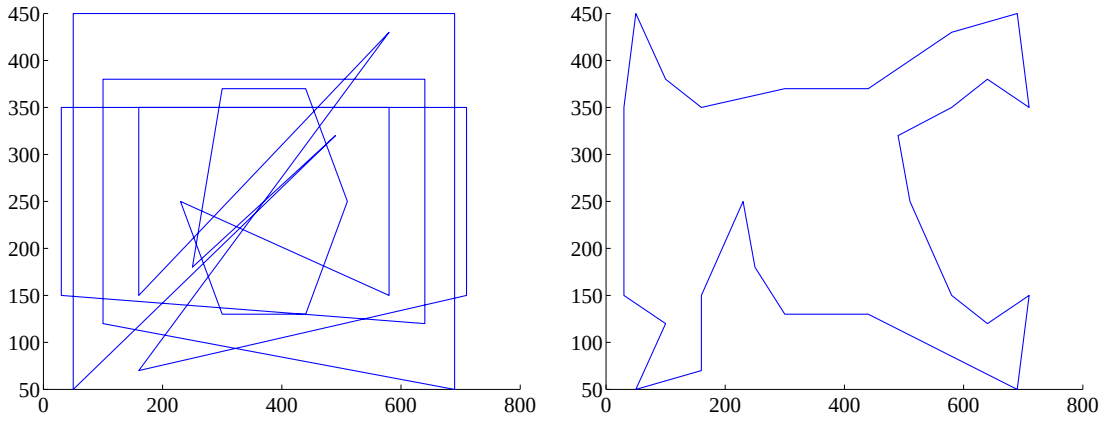
are:

$X = [x_i]$ = [50, 50, 690, 690, 100, 100, 640, 640, 30, 30, 710, 710, 160, 580, 160, 160, 580, 580, 230, 300, 440, 510, 440, 300, 250, 490], i=1,2,...,26.

$Y = [y_i]$ = [50, 450, 450, 50, 120, 380, 380, 120, 150, 350, 350, 150, 70, 430, 150, 350, 350, 150, 250, 130, 130, 250, 370, 370, 180, 320], i=1,2,...,26.

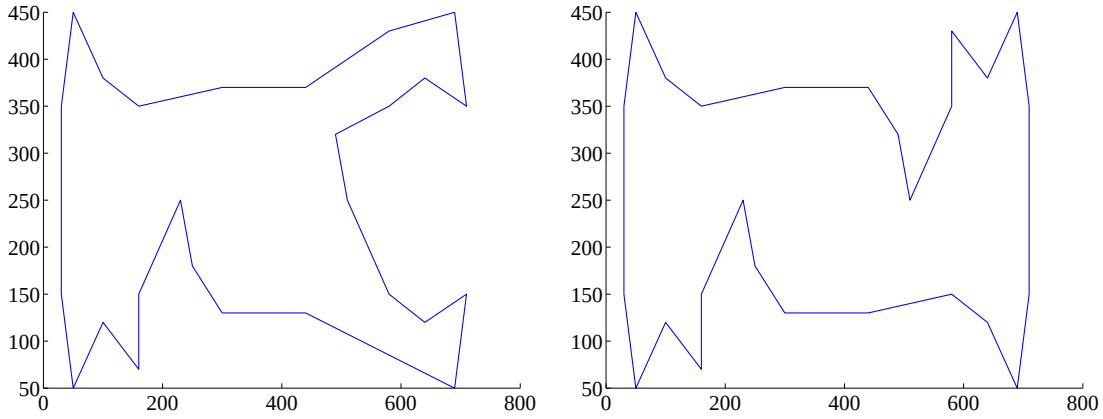
Parameters of the proposed algorithm are set as follows: the number of initial solutions $m = 10$, the number of clone solution $M = 8$, and the maximum generation $G_{max} = 100$. The algorithm is tested by 100 independent runs. Optimization results are presented in Table 2.5 and also compared with other algorithms. In this table, D_m is the average distance of all 100 trials, and D_b is the shortest distance among 100 runs. Success rate means the probability to find the optimal solution. Improvements of the mean results compared with machining path length in sequence are also shown in this table.

It should be noted that the computational results of the machining path distance in [2], [1] are conflicted with our re-tested results. Fig. 2.10(b) and 2.10(c) illustrate the optimal machining paths of these two algorithms and present the correct distance.



(a) path length: 9269.95

(b) path length: 2802.85



(c) path length: 2794.97

(d) path length: 2696.38

Figure 2.10: Comparison of holes machining path planning results of four methods. (a) machining path in sequence (b) EACS algorithm [1] (c) GA [2] (d) the proposed algorithm CRECSA.

Fig. 2.10(a) shows the machining path in sequence $A1 \rightarrow A2 \rightarrow ... \rightarrow A26 \rightarrow A1$. To the best of our knowledge, the best result achieved by the proposed algorithm is shown in Fig.2.10(d). The best machining path is $A1 \rightarrow A9 \rightarrow A10 \rightarrow A2 \rightarrow A6 \rightarrow A16 \rightarrow A24 \rightarrow A23 \rightarrow A26 \rightarrow A22 \rightarrow A17 \rightarrow A14 \rightarrow A7 \rightarrow A3 \rightarrow A11 \rightarrow A12 \rightarrow A4 \rightarrow A8 \rightarrow A18 \rightarrow A21 \rightarrow A20 \rightarrow A25 \rightarrow A19 \rightarrow A15 \rightarrow A13 \rightarrow A5 \rightarrow A1$, and the path length is 2696.38. The same results were also confirmed in the literature.

Based on the simulation results, it can be confirmed that the performance of the proposed algorithm is better than other algorithms.

Furthermore, a bi-objective path planing optimization problem is also used to evaluate the performance of the proposed algorithm. Apart from the moving cost

of the machining tools C_M, the parameter T_{rd} which means reverse shifting times of machining tools is also considered. In machining process, low T_{rd} is desirable to decline accumulated error. Without loss of generality, every machining point is expressed by a x dimensional vector $Z = (z^1, z^2, ..., z^x)$. The position of N machining points can be expressed as N vectors $Z_i = z_i^1, z_i^2, ..., z_i^x, i = 1, 2, ..., N$. The optimization objective is to find a point machining sequence $Z_{i1}, Z_{i2}, ..., Z_{iN}$ that minimizes the following equation:

$$T_{rd} = \sum_{i=2}^{N-1} \sum_{j=1}^{x} A_i^j, \tag{2.9}$$

where

$$A_i^j = \begin{cases} 1, & when\ D_i^j < 0 \\ 0, & otherwise \end{cases}. \tag{2.10}$$
$$Here, D_i^j = (z_{i+1}^j - z_i^j)(z_i^j - z_{i-1}^j).$$

Clearly, this is a multi-objective machining path optimization problem which aims to minimize the moving cost of the cutting tools C_M and the reverse shifting times of machining tool T_{rd}. That is $\min[C_M, T_{rd}]^T$. To solve this bi-objective machining optimization problem, a fast bi-objective non-dominated sorting algorithm (BNSA) is adopted to recognize non-dominated solutions [70].

The data shown in Table 2.6 are used to verify the proposed method. Parameters of the proposed algorithm are set as follows: the number of initial solutions $m = 100$, the number of elite pools $n = 100$, the proliferation rate $M = 1$, and the maximum generation $G_{max} = 3$.

Table 2.7 shows the computational results of different algorithms. Fig. 2.11 illustrates the Pareto solutions of the bi-objective machining path planning problem. The result of CRECSA shown in Table 2.7 and Fig. 2.11 is one of the results from 10 runs. Different colors and different numbers in Fig. 2.11 represent different rank of Pareto solutions. The Pareto front, which consists of solutions with rank 1, is also shown in Fig. 2.11.

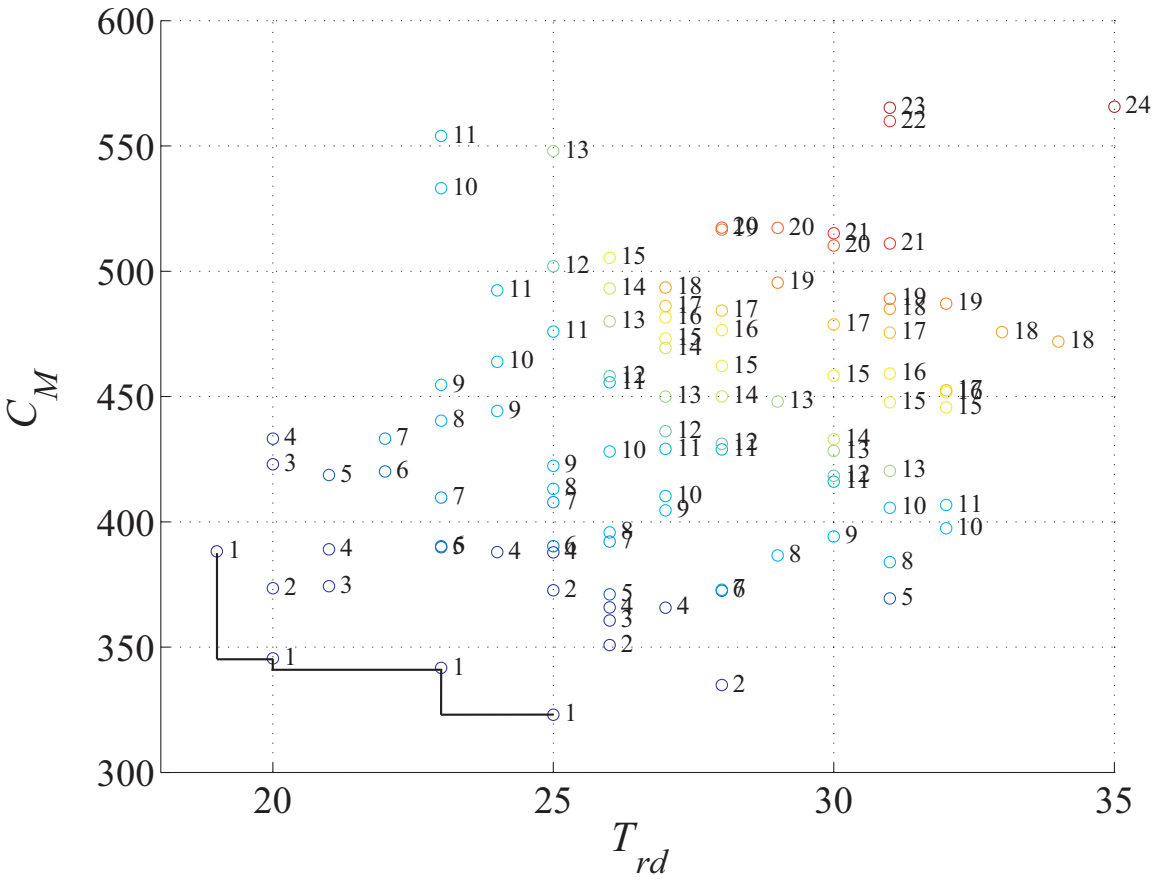


Figure 2.11: Pareto solution distribution of 10-hole part bi-objective machining path planning.

2.5 Conclusions

In this paper, a framework of complete receptor editing based on clonal selection algorithm CRECSA is proposed. The complete receptor editing operation imitates not only the reversion rearrangement, but also the deletion rearrangement of gene segments in natural immune system. The deletion operation further improves the local search ability. Moreover, in order to overcome the drawback of asexual proliferation during the immune maturation process, an improved quantum interference crossover is embedded to enhance the global search ability.

The proposed algorithm CRECSA is evaluated on two benchmark problems containing TSP and HMPP. Computational results reveal that CRECSA is very competitive with the other algorithms in terms of optimal result in single objective problems

and well-distributed solution set in bi-objective optimization problems. Our future work is to explore other complex multi-objective optimization problems, such as cloud computing resources manage problem, software test data generation problem, and radio resource management.

Table 2.4: Comparison of optimization results of RECSA, RECSA+IQC, RECSA+DEL and CRECSA for TSP instances.

TSP	RECSA			RECSA+IQC				RECSA+DEL				CRECSA			
Instance	*PDM*	*PDB*	*SD*	*PDM*	*PDB*	*SD*	t/(s)	*PDM*	*PDB*	*SD*	t/(s)	*PDM*	*PDB*	*SD*	t/(s)
eil51	2.63	1.41	0.0083	2.16	1.17	0.0061	2.99	1.27	0.70	0.0035	2.02	**0.85**	0.23	0.0058	3.70
st70	2.39	0.30	0.011	1.10	0.44	0.0045	6.19	1.56	1.04	0.0052	3.63	**0.39**	0.00	0.0045	6.80
eil76	4.54	3.35	0.0066	3.09	1.86	0.0074	7.36	2.84	1.67	0.0081	4.17	**1.90**	0.56	0.0075	7.94
rd100	5.85	4.55	0.012	3.30	2.40	0.0081	14.71	3.53	2.34	0.0100	7.48	**1.47**	0.49	0.0089	13.47
eil101	6.60	5.25	0.0065	4.90	3.82	0.0062	14.53	3.82	2.86	0.0057	7.56	**3.15**	2.23	0.0050	13.94
lin105	5.39	2.48	0.013	2.32	1.68	0.0061	16.56	2.85	1.35	0.0100	8.27	**0.89**	0.00	0.0052	14.96
pr107	3.40	2.11	0.011	2.77	2.44	0.0016	16.27	2.44	1.57	0.0061	8.74	**0.95**	0.36	0.0039	15.75
pr124	4.42	1.65	0.014	1.64	0.64	0.0072	24.19	2.97	0.94	0.0100	12.01	**0.19**	0.00	0.0022	22.16
bier127	4.41	1.55	0.013	2.00	1.51	0.0042	51.97	2.68	1.54	0.0058	24.38	**1.54**	0.41	0.0081	45.00
pr136	6.28	4.39	0.0092	6.80	5.13	0.0085	62.71	3.61	2.17	0.0091	28.74	**1.94**	0.86	0.0062	52.43
pr152	3.50	2.82	0.0051	1.87	1.21	0.0046	85.20	2.19	1.49	0.0043	37.22	**0.61**	0.00	0.0038	66.88
rat195	13.12	10.55	0.015	**3.15**	2.54	0.0030	171.19	9.74	8.39	0.0075	68.15	4.91	3.96	0.0057	120.46
kroA200	6.59	4.46	0.012	**1.77**	1.17	0.0039	447.68	4.75	3.47	0.0068	181.39	2.82	1.90	0.0052	314.67
lin318	8.51	7.00	0.0082	5.03	4.38	0.0046	3133.79	6.61	5.86	0.0051	1150.34	**3.85**	2.74	0.0059	1914.57
Average	5.55	3.71	0.0104	2.99	2.17	0.00543	289.67	3.63	2.53	0.00694	110.29	**1.82**	0.98	0.00556	186.62

Table 2.5: Best and mean optimization results obtained by different algorithms on 26-hole part.

Algorithm	D_b	D_m	Success rate/ %	Improvement/ %
Hopfield [1]	2923.6	3313.5	25	64.9
ARTIA [1]	2876.1	3091.3	78	67.3
EACS [1]	2834.7	2910.2	93	69.2
GA [2]	3570.34	-	-	38.52
AFSA	2696.38	2934.9	-	68.9
RECSA	2696.38	2815.05	27	69.6
RECSA+IQC	2696.38	2841.98	12	69.3
RECSA+DEL	2696.38	2879.73	2	68.9
CRECSA	2696.38	2703.93	65	70.8

Table 2.6: The coordinates of 10 holes in [1]*.

Point	$(x, y, z, \alpha, \beta, \gamma)$	Point	$(x, y, z, \alpha, \beta, \gamma)$
1	(0,0,0,0,0,0)	6	(50,36,14,16,11,8)
2	(10,30,20,8,7,9)	7	(46,27,16,13,7,2)
3	(20,17,16,12,15,5)	8	(90,40,3,5,9,10)
4	(11,20,21,7,2,0)	9	(36,70,40,6,12,9)
5	(70,2,11,6,12,15)	10	(100,50,4,4,5,11)

* x, y, z determine the position of the hole, and α, β, γ determine the orientation of the hole.

Table 2.7: Comparison of computational results of different algorithms on 10-hole part.

Algorithm	MEACS [1]*	MARTIA [1]*	RECSA**	RECSA+IQC**	RECSA+DEL**	CRECSA**
(T_{rd}, C_M)	(28, 310.9826)	(31, 310.9826)	(26, 319.80)	(26, 319.80)	(26, 319.80)	(25, 322.9896)
	(25, 312.9305)	(26, 315.7935)	(22, 333.95)	(21, 321.31)	(22, 331.40)	(23, 341.6924)
	(24, 323.0159)	(24, 329.7422)	-	(19, 337.88)	-	(20, 345.4935)
	(23, 326.8473)	(21, 352.6687)	-	(17, 377.73)	-	(19, 388.1514)
	(21, 352.6687)	(20, 367.8551)	-	-	-	-
	(20, 367.8551)	(19, 418.9564)	-	-	-	-
	(19, 416.1381)	-	-	-	-	-
CPU time/s	44.3740	33.8860	0.047	0.031	0.071	0.047

In [1]*, CPU: PIII667MHz, RAM:128M

Our PC**: CPU: Core i5-3210@2.5GHz, RAM:6.00GB, OS: WIN8 64-bit

Chapter 3

Improved Clonal Selection Algorithm

3.1 Introduction

Since the concept of cloud computing was proposed in 2006, the term cloud computing has become one of the most well-known buzzword in academic research and IT industries [71]. Many companies, such as Amazon, IBM, Google, Baidu, Alibaba and so on, accelerate their paces in constructing Cloud Computing systems and provide different types of services for institutions or individuals. Based on the different levels service provisioning, the services offered by cloud computing can be classified into three types: Infrastructure as a service (Iaas), Platform as a service (PaaS) and Software as a service (SaaS) . In Iaas, computing resources (such as servers, networking, storage and data-center space) are provided to the consumers. PaaS supplies an on-demand environment for developing, testing, delivering, and managing software applications. SaaS is a method for delivering software applications over the Internet, on demand and typically on a subscription basis [72] [73].

Different from traditional service model, cloud computing provides an on-demand service model with scalability, reliability, high performance and comparatively low cost feasible solution [74]. With the increasing demand for cloud computing, the energy consumption has become a focus problem. Meanwhile, cloud-based services have become very dynamic, resource scheduling problem plays a vital role in cloud

computing. Suppose that there are m available virtual machines (VMs) and n tasks in a workflow, there exist mn different allocation solutions in which the tasks can be mapped to the VM pool. Obviously, for a large value of m and n, making the appropriate decision to allocate cloud resource to end users to the available VM pool is a major concern, and finding an optimal allocation solution by brute force method is computationally very expensive [75].

In addition to exact approaches, lots of different heuristic algorithms have been proposed in cloud environment to solve workflow scheduling problems. In HEFT (Heterogeneous Earliest Finish Time) algorithm proposed by H. Topcuoglu [76] [77], tasks are scheduled according to their priority and each task is allocated to virtual machine that can complete the task at the earliest time. Min-max scheduling algorithm assigns the larger tasks to best resources at the starting to improve the makespan time, response time as well as resource utilization ratio of user request [78] [79]. Deadline based scheduling algorithm and its variants have also been proposed to dynamically provision the resource for scalability as per demand and optimize the key QoS parameters such as task meet with deadline, time etc [80] [81]. Other algorithms, such as FCFS (First come first serve), bin packing, agent and credit based algorithm, best fit algorithm, priority based scheduling and so on, have also been proposed and improved to deal with cloud resources scheduling problem [82].

However, these heuristic algorithms mention above are problem dependent and show different performance for different problems. Therefore, a meta-heuristic algorithm may be effective for solving cloud resources scheduling problem. Meta-heuristic algorithms are problem independent and can be used to solve large optimization problems with acceptable performance in short period of time. Scheduling of task is one such NP-complete problem due to large solution space and takes the long time to obtain the optimal solution. According to the latest literature surveys [82] [83], there are various meta-heuristic algorithms, such as Genetic Algorithm [84] [85] [86], Particle Swarm Optimization (PSO) [87] [88], Ant Colony Optimization (ACO) [89] [90], Simulated Annealing (SA) [91], Bacteria Foraging Optimization (BFO) [92] and so on , proposed to solve task scheduling problem. During the last few decades, an

innovative optimized meta-heuristic algorithm - Clonal Selection Algorithm (CSA) has been widely researched [93] [94], and also applied in different categories such as price forecasting [95], face recognition [96], wireless sensor network planning [97], vehicle routing optimization [98], carbon dioxide emissions predicting [99] and so on. Existing studies confirm that CSA is reliable and performs better than many current meta-heuristic algorithms in multi-discipline areas. However, this biology-inspired algorithm suffers from several problems, such as premature convergence and difficulties in obtaining high-quality solutions in reasonable time [100]. In this research article, an improved clonal selection algorithm is proposed for solving cloud computing resource scheduling problem. This novel algorithm enhances the diversity of solutions by adding vaccines to current solutions. On the other hand, Gauss mutation is used to improve performance of algorithm to escape from local optima. Computational results show that the algorithm works better than other method in terms of reducing execution time.

The remainder of this paper is organized as follows: Section 3.2 introduces the clonal selection algorithm and reviews the the related work on cloud computing resource scheduling problem. Section 3.3 presents problem description and formalization. Then, an improved CSA to solve the problem is proposed in Section 3.4. In Section 3.5, simulation comparisons are given to demonstrate the performance of the algorithm. Finally, we draw the conclusions in Section 3.6.

3.2 Related Works

Clonal selection algorithm is essentially an artificial immune method inspired by natural immune system. In this section, we will firstly introduce artificial immune system (AIS) and clonal selection algorithm in Section 3.2.1. Then, a short review of several meta-heuristic algorithms relevant to cloud resource scheduling is presented in Section 3.2.2.

3.2.1 Artificial Immune System and Clonal selection Algorithm

As mentioned above, many natural inspired methods were designed and developed during the last few decades. Genetic Algorithm (GA) was proposed based on the principle of the theory of the survival of the fittest and the theory of evolution [14]. Ant Colony Optimization (ACO) simulates the behavior of ant colonies [101]. Artificial Bee Colony (ABC) which depends on different intelligent behaviors of honey bee swarms [102]. Meanwhile, an artificial computational system named artificial immune system inspired by the characteristics of learning and memory of natural immune system has received a rapidly increasing interest [103].

As one of the most important artificial immune algorithms, Clonal Selection Algorithm (CSA) was designed based on the clonal selection theory of adaptive immunity. The clonal selection theory describes the basic features of immune response to an antigenic stimulus. It establishes the idea that only those immune cells that recognize the antigens proliferate [103]. By imitating the process of natural immune response, CSA iterates the procedures including initialization, affinity evaluation, clonal operator, affinity maturation, and clonal selection until a pre-specified termination criterion is satisfied.

3.2.2 Cloud Computing Resources Scheduling Problem and Algorithms

With the rapid increasing types of computing services and requirements, assigning tasks and services requests into available cloud resources has become a persistent problem in cloud computing environment. From the perspective of cloud service providers, a variety of virtualized resource need to be allocated to different end users dynamically, correctly and profitably. For cloud users, they are economically driven entities and always compare the cost between different cloud providers [71]. Therefore, many researchers have proposed different heuristics as well as meta-heuristics methods for solving resource scheduling by considering cost, energy, makespan, etc.

As mention above, heuristic algorithms, as a problem dependent method, normally give exact solution for particular problem in finite amount of time but cannot handle large scale hard optimization problems effectively. Therefore, meta-heuristic algorithms have gained much interesting during the last few decades due to their effectiveness in solving complex optimization problems.

Particle Swarm Optimization (PSO) algorithm is a swarm intelligence evolutionary algorithm derived from the physical phenomena of bird flocking [88]. Each particle in the swarm is distributed randomly in the search space with a attribute of velocity and position. During the iterations, each particle updates its own speed and position based on individual and global extremum so as to find the optimal solution. [104] [105] both used adaptive mechanism to change the inertia weight of particle position update, so as to accelerate the convergence speed. While [106] proposed RBPSO algorithm, which also considered how to minimize the degree of load imbalance, maximize the resource utilization and minimize the energy consumption.

Genetic algorithm (GA), a classical evolution method, is inspired by the principles of evolution and was proposed by Holland in 1975. GA and its variants have been proven to be very robust algorithms for solving optimization problems. Three GAs in [107] [108] [73] are all considering not only the makespan time but also the maximum customer satisfaction. Based on the concept of green development, the author proposed Non-dominated Sorting Genetic Algorithm (NSGA-II) control the energy consumption efficaciously [109]. However, all the algorithms mention above have to face the same question that the algorithms' performance will go down when the scheduling task complexity increases. The article [110] proposed an improved algorithm which uses K-means cluster method to classify the tasks at the beginning of scheduling and uses genetic algorithm to scheduling tasks of each class. To the best of our knowledge, clonal selection algorithm (CSA) has rarely been used to solve cloud resources scheduling problem. Even in the latest literature review for scheduling techniques in cloud computing [82] [83], there are no words about CSA.

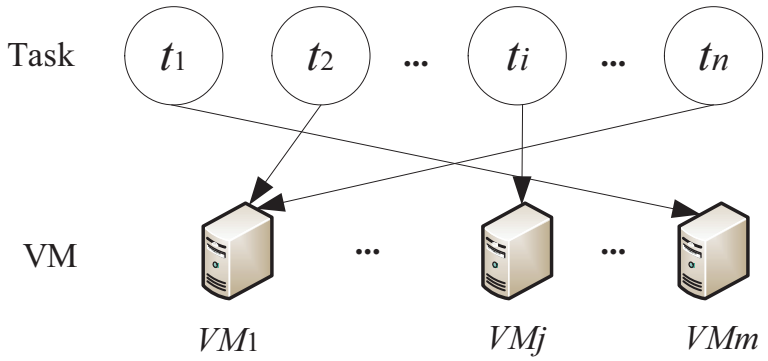


Figure 3.1: Diagrammatic presentation of cloud resources scheduling problem.

3.3 Problem Description and Formulation

This section presents the description and formulation of the cloud resource scheduling problem.

3.3.1 Problem Description

We assume that there are m available computational resources (virtual machine: VM) given as $VM = \{VM_1, VM_2, , VM_m\}$, and n tasks given as $T = \{t_1, t_2, , t_n\}$. The cloud resources scheduling problem can be described as Figure 3.1 shows.

The aim of the problem is to assign each task to available resources so that the total execution time (makespan) is minimized. Every scheduling is defined as a nonnegative matrix E of nm elements. For instance, $e_{ij} = 1$ represents the task t_i is assigned to the virtual machine VM_j.

3.3.2 Problem Formulation

Makespan is referred as the total execution time for the entire workflow T on the available resources R. If the scheduling solution is written as:

$$Sch = \{T, R, T_s, T_f\}, \tag{3.1}$$

where $T_s$and T_f are the start time and the finish time of task t on virtual machine r. Therefore, makespan can be mathematically formulated as follows:

Table 3.1: Relationship between immune system and cloud resource scheduling problem.

Immune system	Resource scheduling problem
Antigen	Scheduling problem
Antibody	Solution
Immune cell gene sequence	Mapping sequence of solution
Affinity	Spantime

$$Makespan = max\{Tf_i\} - min\{Ts_j\} \quad 1 \leq i, j \leq n, \tag{3.2}$$

where $max\{Tf_i\}$ and $min\{Ts_j\}$ denote the latest finish time and the earliest start time for all tasks.

3.4 Proposed Algorithm

In this section, an improved clonal selection algorithm (CSA) inspired by the natural immune system for resource scheduling is described. Before introducing the proposed algorithm, the corresponding relationship between immune system and resource scheduling problem is illustrated in Table 3.1.

Like other evolution algorithms, CSA is usually used to solve optimization problems. Initially, a number of candidate solutions called individuals are generated randomly. Then these solutions will be improved by clonal selection, affinity maturation, and clonal proliferation. During each iteration, the affinity of the current solutions is evaluated. Then a number of solutions are selected to proliferate. The proliferation rate is directly proportional to their affinity level. In order to enhance the diversity of solutions, some new antibodies will be added to current solutions by vaccine injecting.

The flowchart of the proposed algorithm for resource scheduling is illustrated in Figure 3.2 and the notations are explained as Table 3.2.

Initialization:

A prespecified maximal generation number G_{max} is given as the termination criterion. Then generate the initial solution matrix $Ass[M][taskN]$ randomly with the value between 1 and $vmNum$.

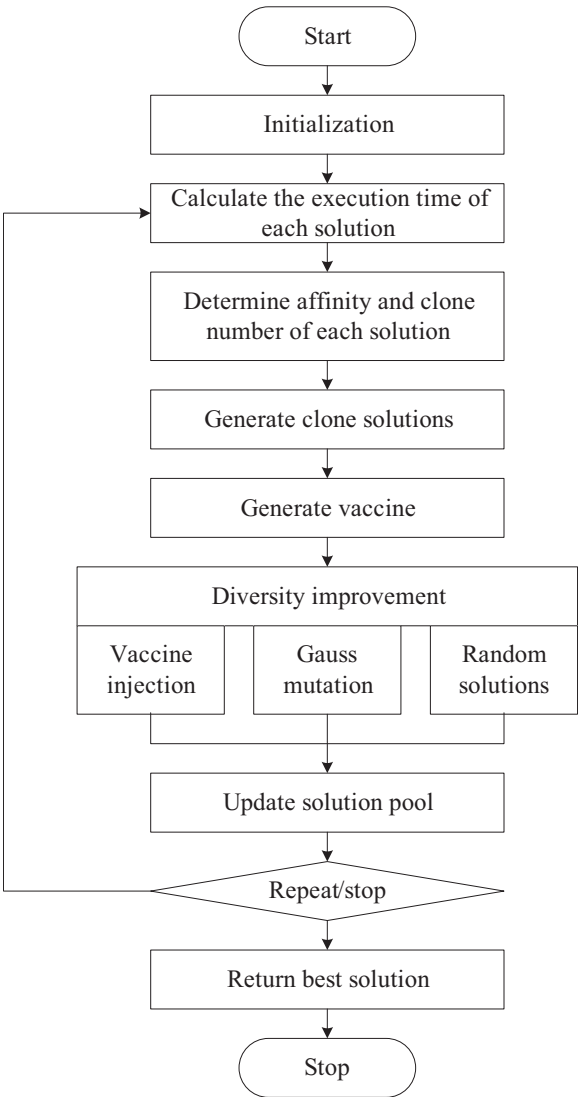


Figure 3.2: Flowchart of the proposed CSA for resource scheduling problem.

Evaluation:

Calculate the execution time of M solutions and then sort them in descending order.

$$timeSum[m] = \sum_{j=1}^{taskNum} time[j][Ass[m][j]]. \tag{3.3}$$

Clonal Proliferation Number Calculation:

Clone the M best solutions with a rate proportional to their affinities. The amount

Table 3.2: Notations and definitions.

Notation	Definition
$vmNum$	Number of virtual machine(VM)
$taskN$	Number of task
G_{max}	Maximal generation number
M	Number of initial solution
$clonalN$	Proliferation rate
$time[i][j]$	Processing time of tasks i assigned to virtual machine j
$Ass[M][taskN]$	Scheduling matrix
$timeSum[M]$	Total execution time of M solutions
$Vmtime[M][vmNum]$	Execution time of each VM
$assN[M]$	Clone number of each solution
$relate[M]$	Affinity of each solution
$Var[M][taskN]$	Gauss mutation solutions

of clone generated for these solutions is determined by Equation 3.4:

$$assN[i] = (int) * (clonalN * relate[i]) \quad i = 1, 2, ..., M, \tag{3.4}$$

where $relate[i]$ is the affinity of solution i calculated by Equation 3.5.

$$relate[i] = (maxT - timeSum[i])/(maxT - minT) \quad i = 1, 2, ..., M, \tag{3.5}$$

where $maxT$, $minT$ are the longest and shortest execution time of current solutions.

Vaccine Selection and Injection:

One antibody is selected as vaccine according to the following step:

(1) Calculate the selection probability of each antibody: $p_{select}[i] = relate[i]/relate_{sum}$, where $relate_{sum} = \sum_{i=1}^{M} relate[i]$. Then the interval of selection probability of antibody i can be described as $[\sum_{j=1}^{i-1} p_{select}[j], \sum_{j=1}^{i-1} p_{select}[j] + p_{select}[i]]$. It should be notice that $p_{select}[i] = 0$ when $i = 0$.

(2) Generate a random number p_{rand} between [0,1], if $\sum_{j=1}^{i-1} p_{select}[j] \leq p_{rand} \leq \sum_{j=1}^{i-1} p_{select}[j] + p_{select}[i]$, then antibody i is selected as vaccine Vac.

Then the vaccine will be injected like the following way:

(1) Generate an vector $V = v_1, v_2, ..., v_{vmNum}$ with the random number between

$[0, 2^{cloudletNum}]$. (2) Compare every bit of ith antibody with vector V, the jth bit of current antibody will be replaced by $Vac[j]$ if $v_j = 0$.

Local Gauss Mutation:

In this paper, CloudSim is used for simulation and experimentation. CloudSim is an open source toolkit and based on Java programming language. The method $nextGaussian()$ of class $Random$ is used to generate Gaussian pseudo-random values. Local Gauss mutation can be expressed as Equation 3.6 (where r is a random value and p_m is mutation probability):

$$Var[i][j] = (int)(Ass[i][j] + p_m * r.nextGaussian() * Math.pow(Math.E, -relate[i])). \tag{3.6}$$

Select M best solutions as elites and put them into $Ass[][]$. Repeat these operations until termination criteria is met.

3.5 Simulation and Comparison

This section presents the simulation results of the proposed algorithm and its comparison with first come first service (FCFS) strategy, greedy scheduling strategy, and classical clonal selection algorithm.

There are some simulation tools, such as Cloud Analyst, EMUSIM, GroudSim, and GreenCloud, which is used to analyze and evaluate the performance of new scheduling methods. However, CloudSim is the most popular simulation tool for resource scheduling. CloudSim is open source toolkit and implemented in university of Melbourne with java programming language [82]. All the computational results in this article are obtained by CloudSim.

In order to study the efficiency of the proposed algorithm,10 tasks will be scheduled on 5 virtual machines. Simulation parameters are shown in Table 3.3.

For illustration purposes, an example of the scheduling solution $AssBest[10] = 1, 1, 2, 2, 4, 5, 5, 5, 4, 3$ for 10 tasks with 5 virtual machines is shown in Table 3.4. The timing property of the scheduled task is shown in Table 3.5. Time means the execution

Table 3.3: Simulation parameters and values.

Parameter	Value
$taskN$	10
$vmNum$	5
$VM[vmNum]$	278,289,132,209,286
Host RAM	2048MB
Host storage	1000000
Host Bandwidth	10000
VM RAM	2048MB
VMM	Xen
VM OS	Linux
Number of CPU	1

Table 3.4: Example of one mapping solution.

Tasks	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
VMs	VM_1	VM_1	VM_2	VM_2	VM_4	VM_5	VM_5	VM_5	VM_4	VM_3

time of one task processed by the assigned virtual machine. In this solution, task t_6, t_7, t_8 are assigned to VM_5, and the processing time is $64.11s$, $70.08s$, and $110.11s$ respectively. These three tasks will be processed by VM_5 one by one. The Finish time of VM_5 is 0.1+64.11+70.08+110.11=$244.40s$. The execution time (makespan) of this scheduling solution is the maximal VM finish time, that is $257.45s$.

In order to further evaluate the efficiency of the algorithm, we measured the execution time of the algorithm for different values of number of tasks and virtual machines. Figure 3.3 shows the execution time in seconds versus the number of tasks for various value of number of virtual machines. Figure 3.4 presents the execution time in seconds versus the number of tasks and number of VMs. The results illustrated in Figure 3.3 and Figure 3.4 show linear or nearly linear increment versus each parameter.

In addition, in order to evaluate the effectiveness of the improved clonal selection algorithm, we compare the proposed algorithm with FCFS [111], greedy method [111], and improved greedy method [112]. Table 3.6 shows the experimental results of cloud computing resource scheduling problem with 10 tasks and 5 VMs. All simulation results are run 10 times on Windows 10 64-bit system with 32GB RAM, and Intel(R)

Table 3.5: Timing properties of scheduled task.

Task	VM	Time(s)	Start Time(s)	Finish Time(s)
6	5	64.11	0.1	64.21
1	1	69.66	0.1	69.76
5	4	80.16	0.1	80.26
3	2	104.56	0.1	104.66
7	5	70.08	64.21	134.29
9	4	147.02	80.26	227.28
10	3	234.97	0.1	235.07
8	5	110.11	134.29	244.40
2	1	179.17	69.76	248.92
4	2	152.79	104.66	257.45

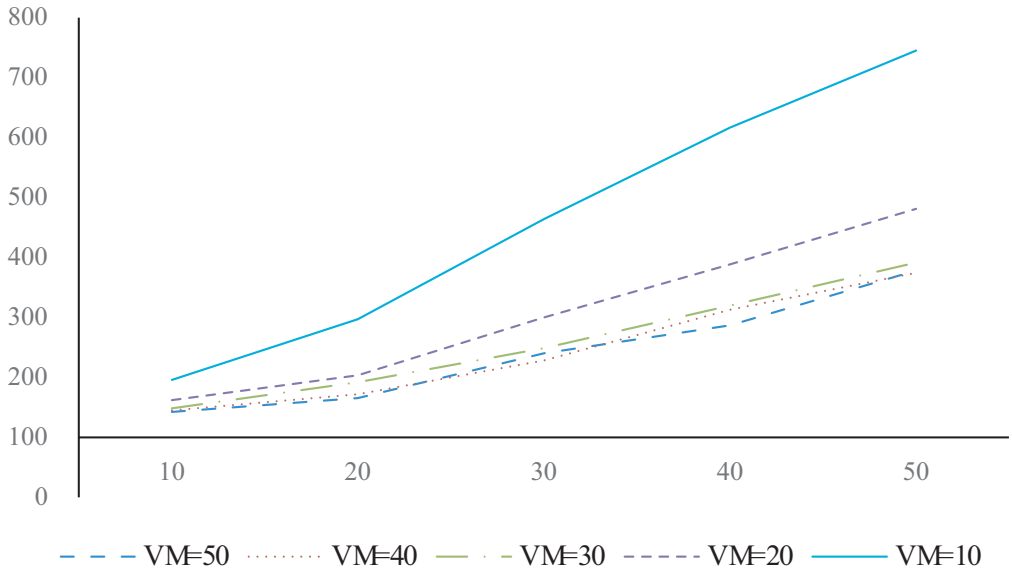


Figure 3.3: Execution time versus number of tasks for different VMs.

core(TM) i5-7200 CPU @ 2.50GHz. Computational results demonstrate that the proposed algorithm outperforms other methods in small size cloud computing resource scheduling problem.

3.6 Conclusions and Future Work

As a novel computing paradigm, cloud computing provides the software, storage, database, infrastructure, hardware, and security as service to end users based on their demands anywhere and anytime in pay-as-you-go model. Because of the special

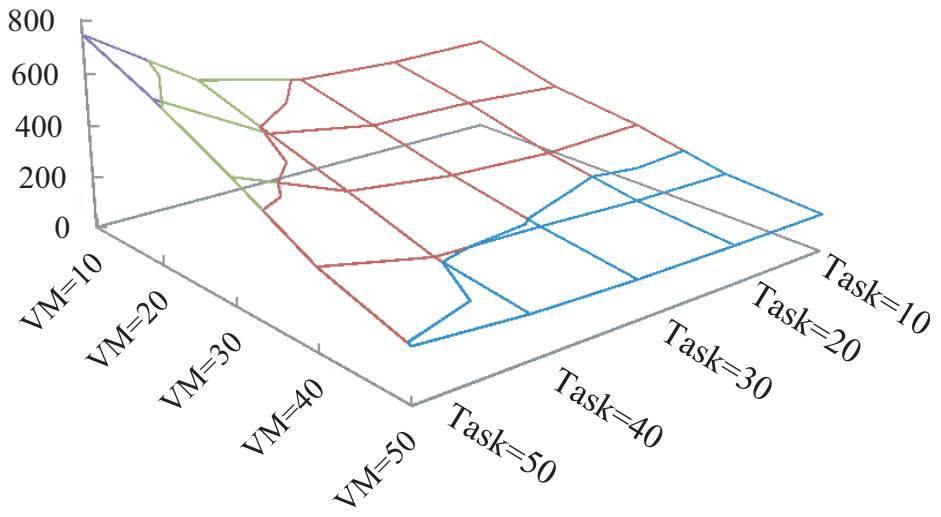


Figure 3.4: Execution time versus number of tasks and virtual machines.

characteristics of cloud computing including the agility, delegation of maintenance, disaster recovery, elasticity, reliability, scalability and security, it has attracted much attention during these years and has been applied to many categories. Meanwhile, how to manage the applications and computing resources in a more efficient way becomes a matter of great concern due to rapid growth of service demand. Therefore, in this work, an improved clonal selection algorithm is proposed to deal with the cloud computing resource scheduling problem. In this algorithm, a vaccine injection operation is designed to enhance the diversity of solutions. Furthermore, Gauss mutation is used to improve ability to escape from local optima. The efficiency and usefulness of the algorithm has been demonstrated by comparison with other resource scheduling methods. The computational results depict that the proposed algorithm performs better than FCFS by 42.4%, greedy method by 4.91%, and the improved greedy method by 5.75 in terms of execution time.

However, in the proposed approach, we just use the execution time as optimization objective. In reality, resource scheduling problem in cloud computing environment are mostly influenced by many other factors such as cost, energy, resource utilization, etc. The future work could focus on multi objective optimization algorithms so as to solve the actual, complex cloud computing resource scheduling problem.

Table 3.6: Comparison of simulation results of different algorithms.

	1	2	3	4	5	6	7	8	9	10	Mean
FCFS	467.60	467.60	467.60	467.60	467.60	467.60	467.60	467.60	467.60	467.60	
Greedy method	283.16	283.16	283.16	283.16	283.16	283.16	283.16	283.16	283.16	283.16	283.16
Improved Greedy	280.78	288.04	289.87	280.78	260.15	283.16	299.78	316.39	280.78	277.18	285.69
Proposed algorithm	269.47	261.93	272.22	260.22	261.86	269.46	267.63	274.56	277.01	278.32	269.27

Chapter 4

Novel Ant Lion-based Random Walk Differential Evolution Algorithm

4.1 Introduction

Recently, stochastic optimization algorithms with random operators have been applied in searching for global value in search space. Due to the existence of random operator, these algorithms outperform its rivals such as deterministic algorithms. Evolutionary algorithm (EAs) [113] is one of the stochastic optimizations algorithms. EAs utilize the approach of stochastic operator to generate new populations when searching for the global minimum of objective function in search space. EAs' great performance in getting rid of the local optima stagnation which most deterministic algorithms are suffering from makes EAs widely used to solve optimum problems in the area of industry.

Some wide-used algorithms in EAs' family are briefly descried as follows: particle swarm optimization algorithm (PSO) [114] simulates the social behavior of creatures, such as bird flock or fish schooling. The ant colony optimization algorithm (ACO) [115] is a stochastic search algorithm that simulates the process of natural ants seeking source of food. The artificial bee colony algorithm (ABC) [116–118] is an optimization algorithm that simulates the foraging behavior of honey bees. The genetic algorithm (GA) [119] is a metaheuristic algorithm inspired by the process of natural selection

strategy.

Differential evolution [26, 120–122] is another variant of EAs. In recent years, DE with a simple population structure has shown its ability of solving various optimization problem, arising from neural network learning [123] tasks, scheduling problems [124], etc. Inspired from genetic algorithm, DE generally has four main parts, including initialization, differential mutation, crossover, and selection. In initialization, a random NP D-dimensional parameter vector is assumed as the population for each generation G which is defined as:

$$x_{i,G} = [1, 2, 3, ...D], i = 1, 2, 3, ..., NP. \tag{4.1}$$

In the mutation procedure, the way of differential is considered as mutant vector which is defined as:

$$v_{i,G+1} = x_{r1,G} + F \cdot (x_{r2,G} - x_{r3,G}), \tag{4.2}$$

where $r_1, r_2, r_3 \in [1, 2, 3..., NP]$ are random indexes. F is the constant which represents the scaling factor. G denotes the generation.

The crossover operator is shown as follows:

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1}, & if\ rand \leq CR \\ x_{ji,G}, & if\ rand > CR \end{cases}, \tag{4.3}$$

where $rand$ is a vector of the numbers randomly generated from 0 to 1. CR is the user-defined crossover constant which is from 0 to 1.

In the selection procedure, the "one-to-one selection" rule is used to determine whether the mutated individual is going to be one of the $G + 1$ generation or not. Since the vector possessing better fitness should be guaranteed in the population, the trail vector $u_{ji,G+1}$ is compared with the former vector $x_{ji,G}$. If $u_{ji,G+1}$ has a better fitness, then it will survive in the population. Otherwise, $x_{ji,G}$ will maintain in the population.

Owing to its simple population structure, low computational complexity, and sol-

id theoretical foundations, DE has attracted many attentions and been successfully applied on many practical problems. Readers can refer [125,126] for more such applications. Although DE has achieved great successes which is mainly due to its great ability of exploring the search space, its performance is still limited, especially for large-scale or dynamic problems [39,127]. To further improve the search capability of DE, many attempts have been made by introducing new differential operators [127], ensemble strategies [128], novel population structures [129], orthogonal crossover [130] and external archive [131]. Moreover, hybridization of different algorithms is widely accepted as a general framework to merge useful characteristics of algorithms, and thus to improve its performance in terms of robustness and efficiency. For DE, many such hybrid DE-based algorithms have been proposed in the literature. For example, Gong et al. [132] proposed an algorithm combining DE with biogeography-based optimization. Lin et al. [133] proposed a hybrid evolutionary algorithm via combining DE with the real-valued genetic algorithm.

In this paper, we propose a new variant of DE by incorporating the random walk mechanism around the ant lion into the search of DE. The resultant hybrid algorithm is called random walk based differential evolutionary (RWDE). In it, the main search procedure is implemented by the original differential mutation and crossover operators in DE, while the random walk which is taken from the recently proposed new algorithm, ant lion optimization algorithm, is carried out to make a complementary role of balancing the exploration and exploitation of the search. The performance of RWDE is verified on both complex numerical optimization problems and clustering tasks. The experimental results based on thirty optimization instances and one clustering problem demonstrate its superiority in terms of the solution accuracy. Moreover, the statistical analysis shows that RWDE is significantly better than its priors.

The rest of this paper is organized as follows: Section 4.2 introduces the hybridization of random walk and DE, i.e., RWDE. Section 4.3, we represent the approach of using RWDE to solve clustering problems. In Section 4.4, the paper discusses the results of RWDE on the benchmark function suit CEC'17 [134] and clustering problems.

In Section 4.5, we conclude the work in the paper and give some future works.

4.2 Ant Lion based random walk DE

Ant lion optimizer [135] is inspired from the relationship between antlions and ants in the trap. Using the antlion hunting as a model, ants are required to walk over the searching space and the antlions need to catch them and get fitter. Thus, we are supposed to define the random walk in this model. Random walk is a stochastic process consisting of steps. It can be defined as:

$$X_n = \sum_{i=1}^{n} r_i, \tag{4.4}$$

where r_i is a random step defined as a stochastic function:

$$r_i = \begin{cases} 1, & if\ rand > 0.5 \\ -1 & if\ rand \leq 0.5 \end{cases}, \tag{4.5}$$

where i is the step of random walk. $rand$ is a uniformly generated random number and $rand \in [0, 1]$. The interaction between the two neighbour random walk can be defined as:

$$X_n = \sum_{i=1}^{n} r_i = \sum_{i=1}^{n-1} r_i + r_n = X_{n-1} + r_n. \tag{4.6}$$

Eq. (4.6) shows that the sequent step of the random walk only depends on the current step. The number of steps is a given constant. Ants utilize the random walk to update their positions in ever generation of the algorithm. Considering the search space in this study generally has boundaries, a normalizing process on Eq.(4.4) is implemented to ensure the ants are located within boundaries. Here, we use min-max normalization function defined as:

$$X_i^t = \frac{(X_i^t - a_i) \times (b_i - c_i^t)}{(d_i^t - a_i)} + c_i, \tag{4.7}$$

where a_i is the minimal value of the random walk of ith variable, b_i is the maximal value of the random walk of ith variable, c_i^t is the minimal value of ith variable at tth iteration, and d_i^t is the maximal value of ith variable at tth iteration.

In this study, we consider that every vector in the population after the mutation of DE is an ant lion in every iteration. We randomly generate the ants position around these ant lions and use the way of random walk mentioned above to update the position of ants. If the fitness of the ant position is better than that of the ant lion, then let the ant in the position be a new ant lion to update the old one. The pseudo-code of RWDE is given in Algorithm 1.

4.3 Evolutionary Clustering Analysis

As an important way of unsupervised learning, clustering is a technique of grouping a set of objects which can be patterns or vectors into clusters. The objects in the same cluster is similar and at the same time, objects in different clusters is dissimilar. One of the remaining questions is how to measure the similarity for assigning objects to the domain of a cluster center. One of the most popular ways of measuring the similarity is the Euclidean distance which measures the distance between objects x and y as:

$$d(x, y) = \sqrt{|x_1 - y_1|^2 + |x_2 - y_2|^2 + ... + |x_n - y_n|^2}. \tag{4.8}$$

If the distance is smaller, the two objects are treated as more similar.

In general, clustering analysis has two main procedures: one is the technique of measuring distance defined above and the other is the clustering algorithm. There are various clustering algorithms, e.g., K-means and hierarchical clustering [136], in the literature. In this paper, we use RWDE to deal with the clustering analysis problem. First of all, clustering in N-dimensional Euclidean space R^N is the process of partitioning a given set of n points into a number, say K, of groups (or, clusters) based on some similarity / dissimilarity metrics. Let the set of n points $x_1, x_2, ..., x_n$ be represented by the set S, and the K clusters be represented by $C_1, C_2, ..., C_k$. Then

we define:

$$\begin{cases} C_i \neq \emptyset \ \text{ for } \ \ i = 1, 2, ...K \\ C_i \bigcap C_j = \emptyset \text{ for } \ i = 1, 2, ...K, j = 1, 2...K, \ \text{ and } \ i \neq j \ \ . \\ \bigcup C_i = S \end{cases} \tag{4.9}$$

Euclidean distances of the points are calculated from their respective cluster centers. Mathematically, the clustering metric for the K clusters $C_1, C_2, ..., C_k$ is given by:

$$D = \sum_{i}^{k} \sum_{x_1 \in C_i} ||x_j - z_i| \,|. \tag{4.10}$$

The task of the RWDE is to search for the appropriate cluster centers $C_1, C_2, ..., C_k$ to ensure that the clustering metric is minimized. The pseudo-code of RWDE for clustering is given in Algorithm 2:

4.4 Experimental results

In this section, the proposed RWDE algorithm is tested on CEC'17 benchmark function suit. The population size is set as 100 and dimension is set to 30. The maximum iteration in the algorithm is 3000. Additionally, for getting rid of the random error, we run all the functions in the algorithm 30 times. Then we calculate their mean and standard deviation of every function results. In addition, we compare the proposed RWDE with some other evolutionary algorithms including differential evolution (DE), ant lion optimization (ALO), particle swarm optimization algorithm (PSO) to show its performance. The clustering problem is taken from [137]. The number of clusters is set to 3 and the iteration is set to 200.

The results of the tested algorithms are summarized in Table 4.1 and the best results are highlighted by bold values. The mean value of 30 independent runs denotes the average performance of the algorithm, while the standard deviation (Std) reflects the robustness of the algorithm. Obviously, it can be found that RWDE performs the best for 27 out of 30 optimization functions in Table 4.1. Fig. 4.1 shows the typical

convergence results of four functions (F16, F23, F24, and F26) selected out of the 30 functions, and it suggests that RWDE can converge fast to the optimal solutions, especially in the latter search phases due to the incorporation of the random walk. From it, it is apparent that the random walk which is performed as a local exploitation operator enables DE to well balance its search exploration and exploitation. All in all, it is obvious that the proposed RWDE algorithm obtains the best value on most functions in comparison with DE, ALO and PSO, and also has competitive performance on the remaining functions.

Then, Wilcoxon matched-pairs signed-rank test is used to precisely analyze the performance among RWDE and its competitors. Wilcoxon's test is a nonparametric procedure, which is used for hypothesis testing involving two samples. Table 4.2 exhibits the result of the Wilcoxon's test, and the p-values which are all smaller than the significance level (i.e., 0.05) shows that RWDE is significantly superior to other compared algorithms.

Fig. 4.2 is the convergence graph of the clustering and Fig. 4.3 is the graph of clustering result by RWDE. In Table III, we compare the proposed RWDE with K-means clustering. The minimum cost calculated by Eq. (4.10) of RWDE is far smaller than that of K-means, which demonstrates that RWDE is more effective for clustering problems.

4.5 Conclusions

In this paper, we proposed an ant lion-based random walk differential evolution algorithm, namely RWDE, to improve its search ability. The random walk which is constructed based on the ant lion optimization algorithm is used as a local search operator in the differential evolution (DE). By doing so, it can be expected that the search exploitation and exploration of DE can be well balanced, and thus improving its search efficiency. Experiment results based on thirty numerical optimization functions and a clustering problem verified that the proposed RWDE can definitely improve the performance of DE and other related algorithms in terms of solution

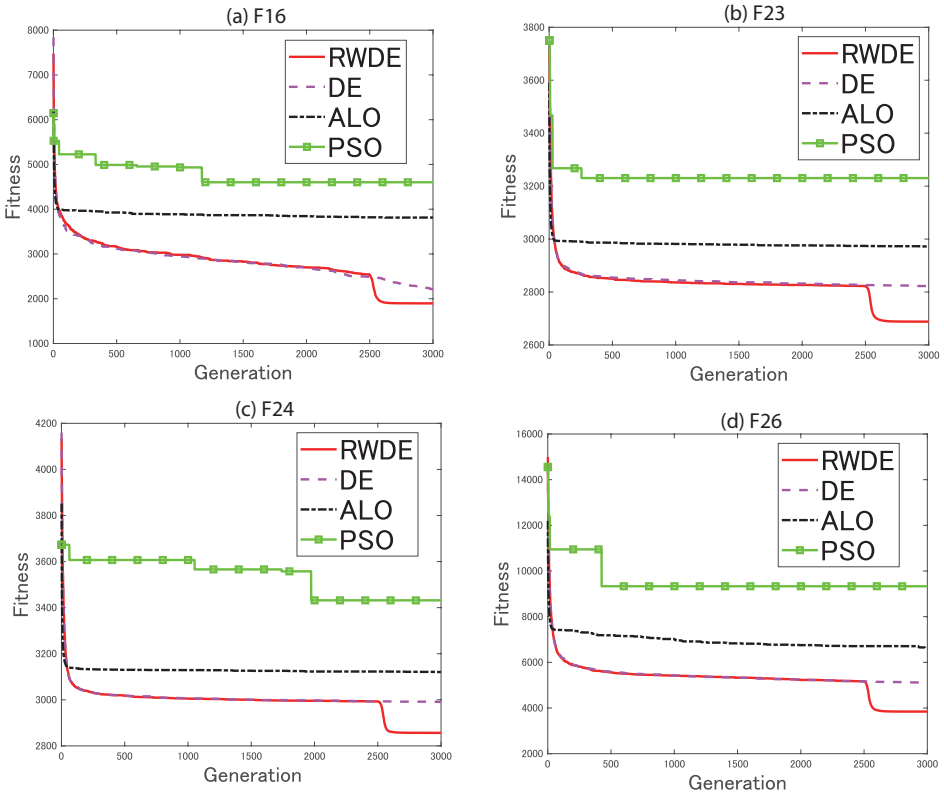


Figure 4.1: Convergence graph of the compared algorithms on (a) F16, (b) F23, (c) F24, and (d) F26.

quality and convergence speed. In the future, we plan to apply RWDE to multiple objective optimization [138–140], combinatorial optimization problems [141–143], vehicle routing problems [144, 145], and neural network leaning tasks [30, 146, 147].

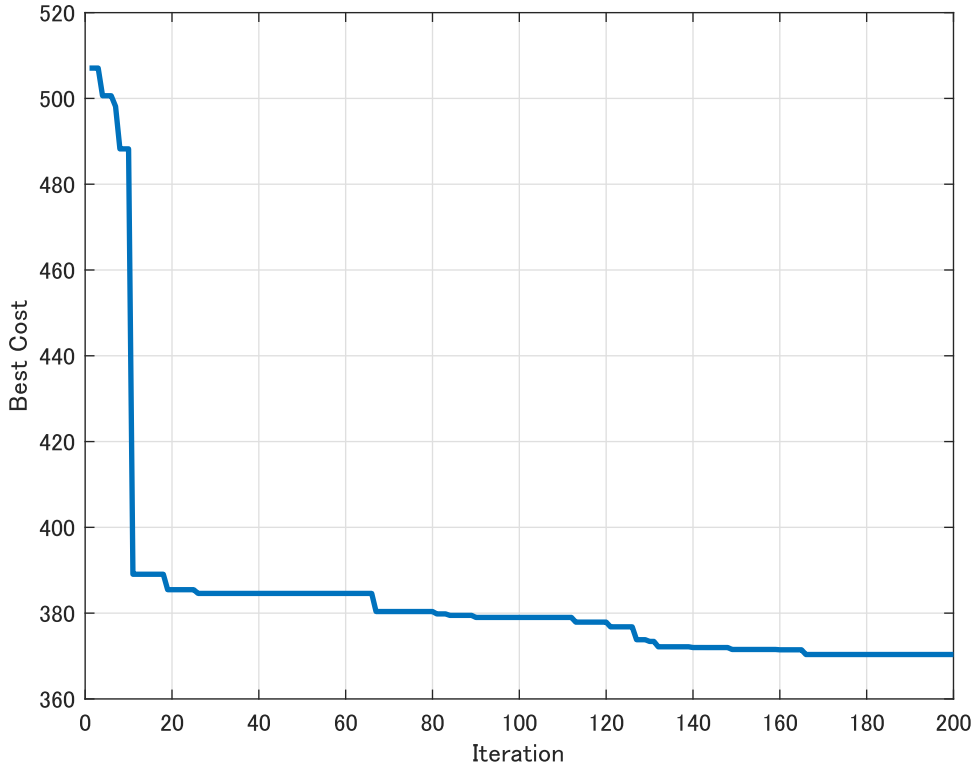


Figure 4.2: Convergence graph of clustering by RWDE.

begin
  Initialization
  Set $t = 0$; $pop = GeneratePopulation(popsize, dim, low, up)$
  $ant = GeneratePopulation(popsize, dim, low, up)$
  for *iteration* ***from*** *1* ***to*** *maxcycle* **do**
    fitnesspop = function(pop)
    for i ***from*** *1* ***to*** $popsize$ **do**
      // DE Mutation
      $pop_{differential} = pop_{r1} + F \cdot (pop_{r2} - pop_{r3})$
      // crossover
      $t = rand(1, dim) < CR$
      $pop_{differential} = t * pop_{differential} + (1 - t) * pop$
      // Random Walk Mutation
      $p_A = random\_walk(ant_{selectant})$
      $p_E = random\_walk(pop_{elite})$
      $ant_i = \frac{p_A + p_E}{2}$
    end
    for i ***from*** *1* ***to*** $popsize$ **do**
      if $fitness_{ant} < fitness(pop)_{differential}$ **then**
        Update u with ant_i
        Update u with $fitness_{ant}$
      end
      else
        Update u with $pop_{differential}$
        Update u with $fitness(pop)_{differential}$
      end
    end
    // Selection
    for i ***from*** *1* ***to*** *size* **do**
      if $fitness(u_i) < fitness(pop_i)$ **then**
        Update pop_i with u_i
        Update $fitness(pop)_i$ with $fitness(u_i)$
      end
    end
  end
end

Algorithm 3: Pseudo-Code of RWDE

```
begin
    // Initialization
    Set t = 0
    Set K = 3
    pop_i = GeneratePopulation(size, dim, low, up)
    for iteration from 1 to maxcycle do
        fitness(P_i) = Cost(pop_i)
        // Mutation and crossover
        u_i = RWDE(P_i)
        fitness(u_i) = Cost(u_i)
        // Selection
        for i from 1 to size do
            if fitness(u_i) < fitness(P_i) then
                Update P_i with u_i
                Update fitness(P_i) with fitness(u_i)
            end
        end
    end
end
```

Algorithm 4: Pseudo-Code of Clustering via RWDE

Table 4.1: Experiment results of RWDE and DE on CEC'17.

	RWDE		DE		ALO		PSO	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
F1	**1.00E+02**	7.92E-15	1.00E+02	6.98E-15	1.48E+10	3.78E+09	5.06E+10	5.45E+09
F2	**1.61E+06**	8.65E+06	3.41E+06	7.28E+06	1.43E+35	2.90E+35	1.64E+40	6.91E+40
F3	**3.09E+02**	7.10E+00	3.19E+02	1.63E+01	7.56E+04	2.09E+04	1.10E+05	1.50E+04
F4	**4.17E+02**	7.04E-01	4.59E+02	1.92E+00	1.63E+03	3.47E+02	9.69E+03	1.50E+03
F5	**5.37E+02**	1.18E+01	6.78E+02	7.11E+00	8.00E+02	2.89E+01	9.25E+02	2.17E+01
F6	**6.00E+02**	2.97E-08	6.00E+02	3.84E-08	6.58E+02	9.80E+00	6.86E+02	4.00E+00
F7	**8.59E+02**	5.15E+01	9.07E+02	1.10E+01	1.28E+03	6.16E+01	2.16E+03	1.50E+02
F8	**8.40E+02**	1.35E+01	9.77E+02	1.00E+01	1.08E+03	2.49E+01	1.19E+03	2.31E+01
F9	**9.00E+02**	0.00E+00	9.00E+02	0.00E+00	7.10E+03	1.98E+03	1.46E+04	1.41E+03
F10	**4.28E+03**	5.87E+02	7.80E+03	3.31E+02	8.51E+03	2.47E+02	8.26E+03	3.19E+02
F11	**1.12E+03**	1.88E+01	1.16E+03	3.18E+01	2.62E+03	4.77E+02	7.08E+03	1.07E+03
F12	7.83E+03	4.03E+03	7.43E+03	5.80E+03	1.37E+09	3.63E+08	5.98E+09	6.38E+08
F13	**1.38E+03**	7.30E+00	1.38E+03	7.99E+00	4.49E+08	1.78E+08	2.42E+09	5.54E+08
F14	**1.46E+03**	6.32E+00	1.46E+03	3.31E+00	2.89E+05	2.04E+05	6.05E+05	2.66E+05
F15	**1.53E+03**	9.28E+00	1.53E+03	8.59E+00	4.94E+07	3.14E+07	1.80E+08	8.67E+07
F16	**1.90E+03**	2.01E+02	2.21E+03	4.31E+02	3.81E+03	2.70E+02	4.41E+03	2.79E+02
F17	**1.75E+03**	1.67E+01	1.78E+03	8.02E+00	2.67E+03	2.21E+02	3.02E+03	1.78E+02
F18	**1.84E+03**	3.80E+00	1.84E+03	3.80E+00	3.48E+06	2.49E+06	8.58E+06	2.81E+06
F19	1.92E+03	6.38E+00	1.91E+03	5.99E+00	1.07E+08	4.82E+07	3.06E+08	1.15E+08
F20	**2.03E+03**	1.30E+01	2.03E+03	2.86E+01	2.84E+03	1.89E+02	2.80E+03	8.49E+01
F21	**2.34E+03**	1.31E+01	2.46E+03	9.83E+00	2.57E+03	2.44E+01	2.69E+03	1.90E+01
F22	**2.30E+03**	0.00E+00	2.30E+03	0.00E+00	6.78E+03	2.89E+03	7.74E+03	4.87E+02
F23	**2.69E+03**	1.34E+01	2.82E+03	8.89E+00	2.97E+03	2.90E+01	3.26E+03	4.18E+01
F24	**2.86E+03**	1.07E+01	2.99E+03	1.09E+01	3.12E+03	2.50E+01	3.46E+03	4.60E+01
F25	**2.89E+03**	2.33E-02	2.89E+03	2.72E-02	3.73E+03	2.20E+02	6.44E+03	5.84E+02
F26	**3.84E+03**	1.40E+02	5.11E+03	1.50E+02	6.62E+03	7.79E+02	9.72E+03	4.60E+02
F27	**3.19E+03**	8.89E+00	3.19E+03	1.02E+01	3.21E+03	1.68E+01	3.79E+03	8.61E+01
F28	**3.13E+03**	5.28E+01	3.14E+03	5.68E+01	3.75E+03	4.98E+02	6.34E+03	3.74E+02
F29	**3.35E+03**	3.27E+01	3.48E+03	1.30E+02	4.71E+03	2.48E+02	5.56E+03	2.92E+02
F30	5.01E+03	5.57E+01	5.00E+03	5.15E+01	8.16E+07	3.87E+07	2.86E+08	8.56E+07

Table 4.2: Results obtained by the Wilcoxon test.

RWDE VS.	R^+	R^-	p-value
DE	356.5	78.5	0.001718
ALO	465	0	0.000002
PSO	447.0	18	0.00001

Table 4.3: Clustering Results of RWDE and K-means.

	Minimum Cost
RWDE	370.36
K-means	593.391

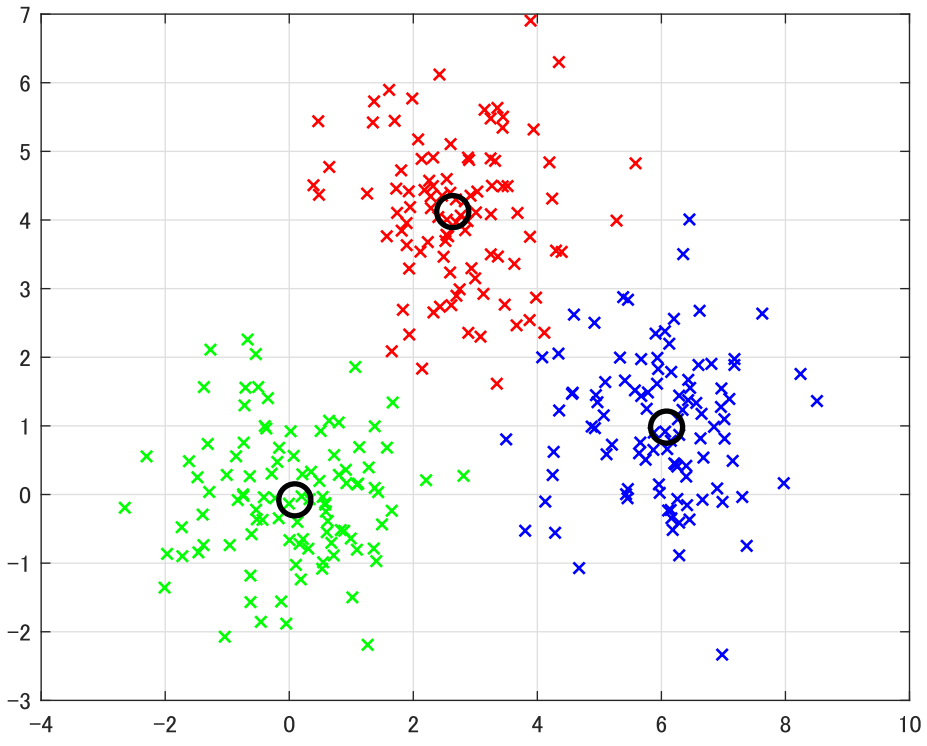


Figure 4.3: Cluster result graph of RWDE.

Chapter 5

Hybrid Hypercube and Spherical Evolution

5.1 Introduction

Nature-inspired meta-heuristic (NMH) algorithms are a class of optimizers that simulate the evolutionary process or social behavior of creatures for complex real-world optimization problems [148], e.g., traveling salesman problem [149,150], multi-objective optimization problem [94, 151], graph planarization problem [141, 152] and dynamic problem [39]. Genetic algorithm is one of the classic evolutionary algorithms based on Darwinian theory, where crossover and mutation operators are adopted to simulate the natural selection process [153]. As a representation of algorithms inspired by the social behavior of animals, practical swarm optimization imitates the bird flocking or fish schooling [154]. In recent two decades, more and more NMH algorithms were proposed, such as gray wolf optimization which is inspired by the leadership hierarchy and hunting mechanism [155], artificial immune system which simulates the learning and adaptive mechanisms of living organisms in protecting themselves against antigens [156–158], ant colony optimization [115, 159] and artificial bee colony algorithm which mimic the foraging behavior of an ant and bee colony [160, 161], respectively, and so on [26, 162, 163]. Although these new algorithms are inspired by different animals or behaviors, there is no quintessential difference among them. A study found that most of NMH algorithms support a mathematical formation named search

pattern which can be represented as Eq. (5.1) [164].

$$X_{i,d}^{new} = X_{i,d}^{old} + \sum_{k=1}^{n} S\left(X_{\alpha,d}^{k}, X_{\beta,d}^{k}\right), \tag{5.1}$$

where $X_{i,d}$ indicates the value of the dth dimension in the ith solution, and X_α and X_β are two certain solutions selected by a selection strategy. $S(\cdot)$ represents the updating units in the search operator named search style, and n denotes the number of updating units. Most of NMH algorithms adopt hypercube search style that is the first-order difference among solutions as the updating unit, and it can be represented as Eq. (5.2).

$$\mathrm{HS}\left(X_{\alpha,d}, X_{\beta,d}\right) = SF_1() \cdot \left(SF_2() \cdot X_{\alpha,d} - SF_3() \cdot X_{\beta,d}\right), \tag{5.2}$$

where $SF_1()$,$SF_2()$,and $SF_3()$ are three scaling factor functions used to adjust the scale of difference between X_α and X_β.

Different search style determines the different characteristic of the search operator and affects the performance of the algorithm. In this paper, we adopt a combination of two different search styles as a united search operator to propose a hybrid metaheuristic algorithm HHS which adopt both the hypercube and spherical search styles. Also, we design a sophisticated control rule to use these search styles, aiming to well balance the exploration and exploitation of the search. Experimental results based on numerical optimization problems demonstrate the effectiveness and outstanding performance of the proposed hybrid algorithm in comparison with other the-state-of-the-art algorithms.

The rest of this paper is organized as follows: a brief introduction of sphercial evolution is given in Section 5.2. The hybrid algorithm HHS is presented in Section 5.3. The experimental results and discussions are given in Section 5.4. Finally Section 5.5 gives conclusions and some future works.

5.2 Spherical Evolution

Spherical evolution (SE) was proposed by Tang in 2019. It was composed of a spherical search style instead of a hypercube search style, and use a selection strategy similar to differential evolution [164]. Different from the rectangular search space of the hypercube search style in two-dimension space, the spherical search style presents a circular search space, and the exploration ability of spherical search is stronger than the hypercube search, which means spherical evolution can widely search the whole search space and avoid local optimal more possibly. The spherical search style can be represented as Eq. (5.3)-(5.7) in high-dimension, two-dimension, and one-dimension, respectively [165].

$$SS_{\geq 3}\left(X_{\alpha,d}, X_{\beta,d}\right) = \mathrm{SF}() \cdot \left\|X_{\alpha,*} - X_{\beta,*}\right\|_2 \cdot \prod_{k=d}^{dim-1} \sin\left(\theta_j\right), d = 1 \tag{5.3}$$

,

$$SS_{\geq 3}\left(X_{\alpha,d}, X_{\beta,d}\right) = \mathrm{SF}() \cdot \left\|X_{\alpha,*} - X_{\beta,*}\right\|_2 \cdot \cos\left(\theta_{d-1}\right) \cdot \prod_{k=d}^{\dim -1} \sin\left(\theta_j\right), 1 < d \leq \dim -1, \tag{5.4}$$

$$SS_{\geq 3}\left(X_{\alpha,d}, X_{i,d}\right) = \mathrm{SF}() \cdot \left\|X_{\alpha,*} - X_{\beta,*}\right\|_2 \cdot \cos\left(\theta_{d-1}\right), d = \dim, \tag{5.5}$$

$$\begin{aligned} SS_2\left(X_{\alpha,d}, X_{\beta,d}\right) &= SF() \cdot \left\|X_{\alpha,*} - X_{\beta,*}\right\|_2 \cdot \sin(\theta) \\ SS_2\left(X_{\alpha,d}, X_{\beta,d}\right) &= SF() \cdot \left\|X_{\alpha,*} - X_{\beta,*}\right\|_2 \cdot \cos(\theta), \end{aligned} \tag{5.6}$$

$$SS_1\left(X_{\alpha,d}, X_{\beta,d}\right) = SF() \cdot \left|X_{\alpha,d} - X_{\beta,d}\right| \cdot \cos(\theta), \tag{5.7}$$

where $\left\|X_{\alpha,*} - X_{\beta,*}\right\|_2$ denotes the radius of sphere computed by Euclidean norm in high dimension, θ donetes the angle between $X_{\alpha,*}$ and $X_{\beta,*}$ and it is generated by a random number of uniform distribution between $[0, 2\pi]$.

The main characteristic of spherical search style is the larger search range and an undirected search trajectory which can provide more promising evolutionary paths in comparison with the hypercube search style. Besides, contrary to the unidirectional evolution that is easy to fall into local optimal, undirected evolution may help algorithms escape local optima. However, the disadvantages of undirected evolution are duplicated search and slow convergence speed, because the rotation angle is a random distribution between $[0, 2\pi]$, which a high risk of generating an opposite to the pervious evolutionary direction. Both unidirectional and undirected evolution are needed for an effective metaheuristic algorithm, though it is difficult to coordinate each other.

5.3 Hybrid Hypercube and Shperical Evolution

5.3.1 Motivation

The performance of evolutionary algorithms depends on two main aspects. One is the ability to search an effective evolutionary direction that can improve the current individual or population. Another is the ability to continue searching in the vicinity of that effective evolutionary direction until reach optima [21]. As mentioned above, the spherical search style has more opportunities to find an effective evolution direction which is benefit for powerful exploration. Meanwhile the possibility of searching in an ineffective direction that escaping from the global optima can also be increased. Contrary to the spherical search style, the exploitation that can search in neighbors of effective direction is more powerful than exploration in hypercube search style. This naturally motivate us to propose a hypothesis of whether the combination of these two search styles may make the search perform better.

5.3.2 The Principle of HHS

Based on the above motivation, we proposed a hybrid hypercube and spherical (HHS) evolution algorithm. The core of the proposed algorithm can be described as three

steps:

(1) Let initial individuals in the population search by the spherical style and propagate them to the next generation.

(2) Evaluate the offsprings to verify whether the algorithm has found an effective evolutionary direction or not.

(3) Determine the search style of individuals in the current generation based on the results of the previous step and the search style of their parent. Repeat step (2) and (3) until fulfilling the termination conditions.

Specifically, there are four situations should be considered.

(a) When the offspring is better than its parent by the spherical search, it should continue in the same direction by a hypercube search operator

(b) When the offspring is not better than its parent by the spherical search, it should find another direction by a spherical search operator

(c) When the offspring is better than its parent by the hypercube search, it should continue in the same direction by a hypercube search operator

(d) When the offspring is not better than its parent by the hypercube search, it should find another direction by a spherical search operator

The Pseudo-code of HHS is shown as in Algorithm 5. Parameter $F0$, $\sigma 0$, and $flag$ of an individual present the initial scale factor in the hypercube search, the initial variance of Gauss distribution in SF turning and the search styles of the parent of x_i which are setted to 2.5, 0.5 and 0 (means spherical the style), respectively. The turning methods of SF and DSF are same as these in SE. It should be noted that we use σ instead of 0.1 as variance in Gauss distribution. In HHS, k in the spherical mutation is dependent on the mutation dimensions d which are selected by DSF. EP_i and F_i in the spherical mutation represent the effective path that made x_i better than its parent and corresponding scale factor, respectively. It is worth noting that

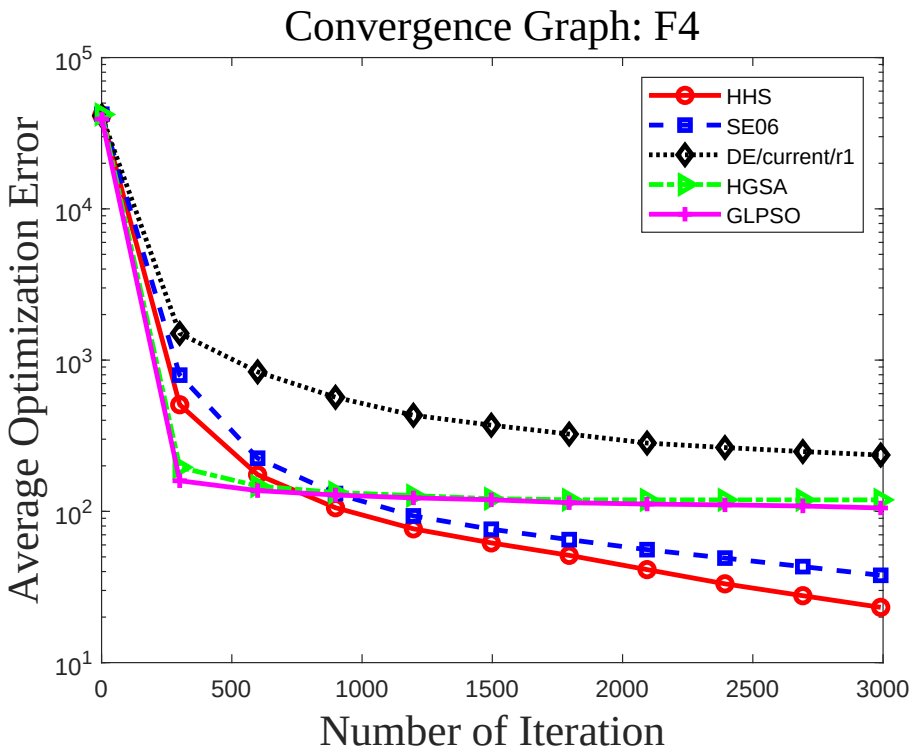


Figure 5.1: Convergence graph of F4.

an adaptive parameter control strategy is embedded into four situations mentioned above. In case (a), an effective path has been found, a tentative hypercube search should be performed as lines 23-25 in Algorithm 5. In case (b), expanding the search range may also increase the probability of finding an effective path as line 35. In case (c), the step size should be increased to obtain faster convergence as realized by line 21. In case (d), the optimal area may have been missed, therefore we need to adjust the direction rather than expanding the search range as implemented in line 31-33.

5.4 Experimental Results

To verify the performance of the proposed HHS algorithm, 30 benchmark functions of CEC2017 which composed of unimodal functions (F1-F3), simple multimodal functions (F4-F10), hybrid functions (F11-F20) and composition functions (F21-F30) are adopted to experiment. Besides SE, HHS was compared with three state-of-the-

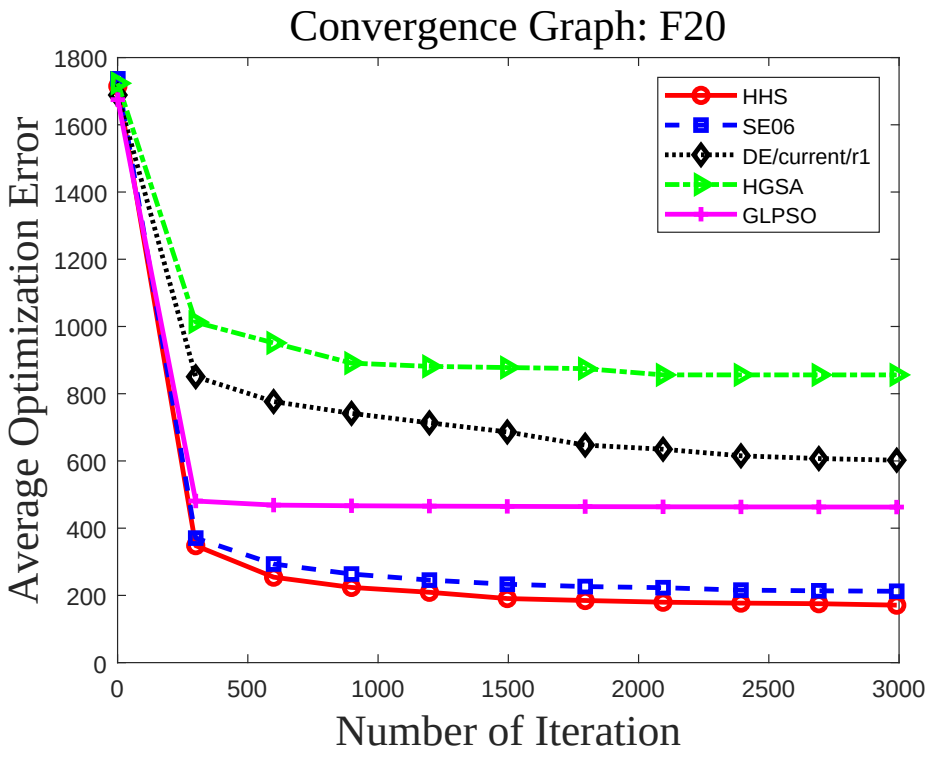


Figure 5.2: Convergence graph of F20.

art algorithms including differential evolution (DE) [120], hierarchical gravitational search algorithm (HGSA) [166] and genetic learning practical swarm optimization (GLPSO) [167]. Each algorithm was run 30 times with population size 100, and the maximum number of the function evaluations $D*1E+4$ where D is the dimension size. It is worth mentioning that the $DE/current/r1$ of DE and $SE06$ of SE are adopted to test. The parameter setting as shown in Talbe 5.3. The experiment results are listed in Talbe 5.1 where the highlighted ones are the best. In it, the obtained results are the errors between the returned the number of tested functions for which function values and the known best optimal value. Additionally, W/T/L denotes the proposed algorithm HHS performs significantly better/tied/worse than its competitor under a significance level 5%. The analysis results of Wilcoxon matched-pairs signed-rank test are presented in Table 5.2. The representative convergence and box-and-whisker plot results are exhibit in Figs. 5.1-5.4.

It is obvious that the proposed HHS algorithm performs superior on composition

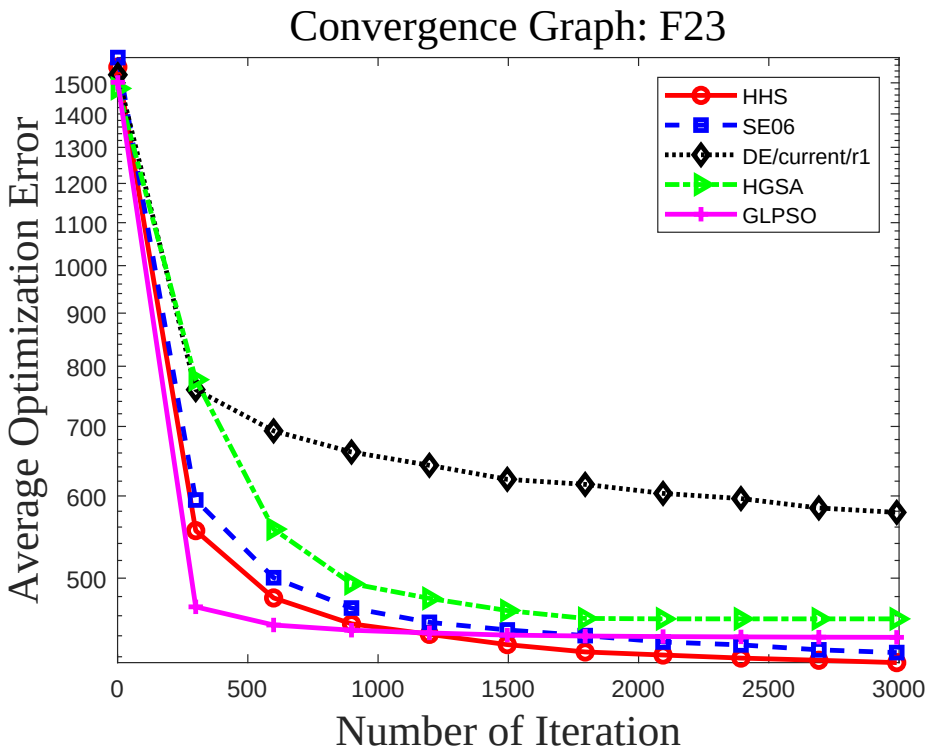


Figure 5.3: Convergence graph of F23.

functions, and competitive on other functions in comparison with those state-of-the-art algorithms according to Table 5.1 and Table 5.2. Moreover, it is worth noting that the overall slope of the curve belongs to HHS is larger than that of other algorithms in Figs. 5.1-5.3. This characteristic reveals that the algorithm is not easy to premature convergence and drop into local optimum, which demonstrates our assumption that combining the advantages of spherical search and hypercube search is reasonable. We can also observe that HHS is still convergent when the search is adjacent to the termination condition. We convince that HHS can perform better than other algorithms after increasing the number of function evaluations. As a result, we can conclude that the proposed hybrid hypercube and spherical evolution algorithm performs effective and robust on optimization problems.

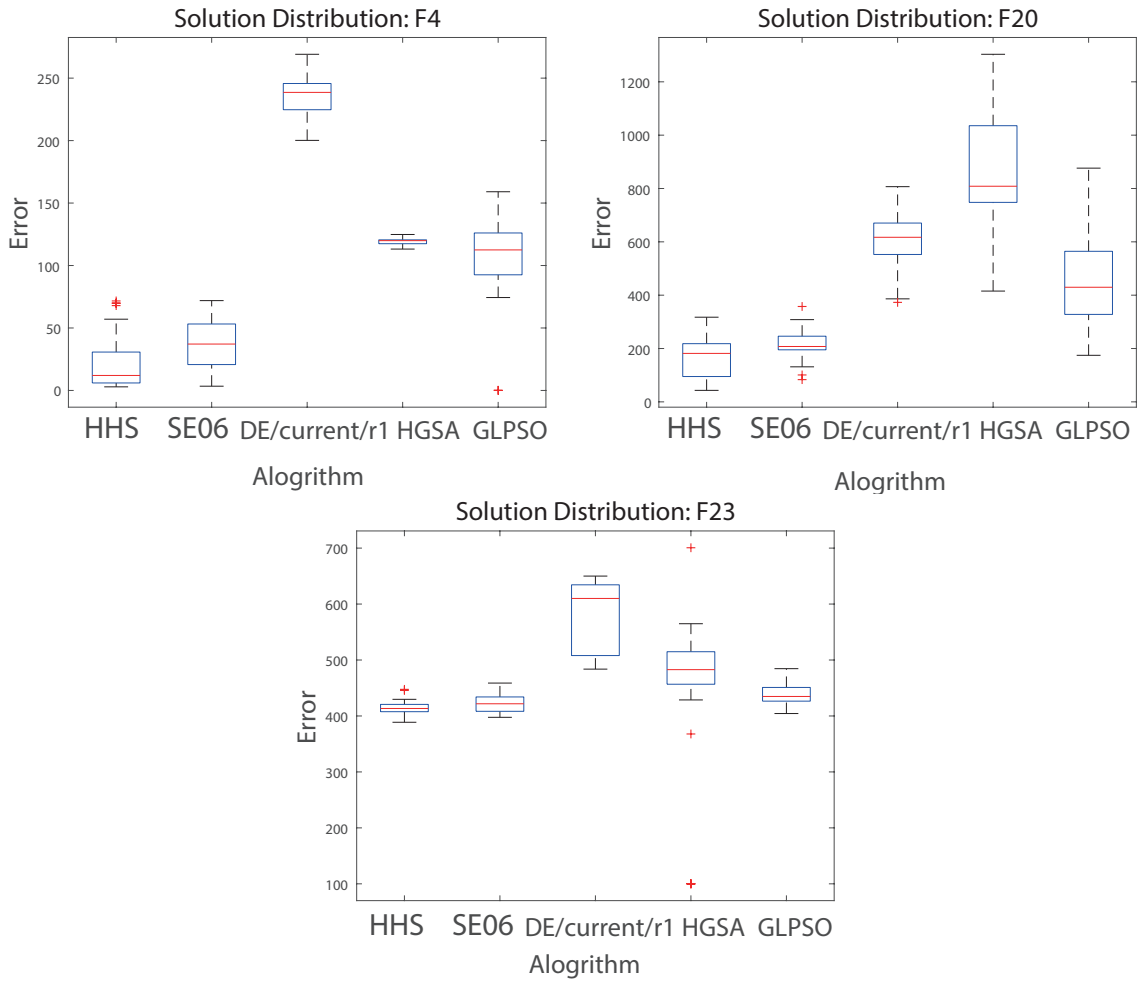


Figure 5.4: Box-and-whisker graphs of F4, F20 and F23.

5.5 Conclusions

In this paper, we study the characteristic of two different and interrelated search styles, i.e., the hypercube search and spherical search. We find that these two search styles affect two main abilities of the algorithm from the perspective of exploration and exploitation. Inspired by it, we propose a hybrid hypercube and spherical evolution (HHS) algorithm with an exquisite parameter control strategy. To assess the performance of the proposed HHS algorithm, we compared it with SE06, DE/current/r1, HGSA and GLPSO on benchmark functions in CEC2017. Experimental results prove the superior performance of HHS. In the future, we want to turn our attention to solve

some real-world problems, e.g., Internet of vehicles [145, 168], multiple-valued logic network synthesis [169, 170] and protein structure prediction [139, 140].

begin
  /* Initialization */
  for *i = 1 to M* **do**
    Randomly initialize x_i
    Evaluate $f(x_i)$
  end
  while *Termianal Condition* **do**
    Scale factor (SF) tuning with $N(0, \sigma)$
    Dimension selection factor (DSF) tuning
    for *i = 1 to M* **do**
      Select two distinct individuals x_{r1} and x_{r2} randomly, $i \neq r1$ and $i \neq r2$
      if $flag_i == 1$ **then**
        /* Spherical mutation */
        $x'_{i,d} = x_{i,d} + SS_k\left(x_{r1,d}, x_{r2,d}\right)$
      end
      else
        /* Hypercube mutation */
        $x'_i = x_i + F_i \cdot EP_i$
      end
      Evaluate $f(x'_i)$
      if $f(x'_i) < f(x_i)$ **then**
        if $flag_i == 1$ **then**
          $F_i = F_i + U(0, 0.1)$
        end
        $flag_i = 1$
        $F_i = F_i - U(0, 0.1)$
        $\sigma_(i) = \sigma 0$
        $EP_i = x'_i - x_i$
        $x_i = x'_i$
      end
      else
        if $flag_i == 1$ **then**
          $flag_(i) = 0$
          $F_(i) = F0$
          $\sigma_(i) = \sigma(i) - U(0, 0.1)$
        end
        else
          $\sigma_(i) = \sigma(i) + U(0, 0.1)$
        end
      end
    end
  end
end

Algorithm 5: Proce-dures of HHS

Table 5.1: Experiment results on CEC 2017.

	HHS		SE/current/r1 [164]		DE/current/r1 [120]		HGSA [166]		GLPSO [167]	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
F1	**1.8635E+03**	9.1792E+02	1.4913E+04	6.6144E+03	1.0624E+07	1.7674E+06	2.5825E+03	2.4958E+03	3.4644E+03	5.2567E+03
F2	6.4888E+12	1.3604E+13	2.0068E+13	2.6966E+13	4.0866E+24	5.0206E+24	**4.1639E+05**	2.0000E+06	6.87336E+11	1.6044E+12
F3	5.6464E+04	6.7760E+03	7.1855E+04	8.7703E+03	4.8272E+04	4.8272E+03	4.3258E+04	5.4921E+03	**2.6058E+04**	2.4546E+04
F4	**2.3036E+01**	2.3156E+01	3.7612E+01	2.0972E+01	2.3535E+02	1.6436E+01	1.1915E+02	2.6272E+00	1.0529E+02	3.5563E+01
F5	8.0380E+01	9.6564E+00	8.9863E+01	1.6600E+01	2.4717E+02	1.4400E+01	1.5299E+02	1.2839E+01	**6.4081E+01**	1.9227E+01
F6	3.0365E+00	4.2736E-01	4.5388E+00	7.5057E-01	3.6621E+01	2.4160E+00	8.1917E+00	4.5442E+00	**1.6567E-01**	8.9411E-02
F7	1.3249E+02	1.1059E+01	1.4494E+02	2.1131E+01	3.2612E+02	1.2082E+01	**4.0761E+01**	3.0080E+00	1.4292E+02	3.1953E+01
F8	8.3127E+01	1.0208E+01	8.8589E+01	8.9032E+00	2.4457E+02	1.1535E+01	9.9794E+01	9.0330E+00	**6.4210E+01**	1.3738E+01
F9	1.7413E+03	4.3021E+02	2.3486E+03	4.2888E+02	2.8307E+03	3.4700E+02	**7.9581E-14**	5.2988E-14	4.1587E+02	2.0831E+02
F10	**2.1375E+03**	2.1279E+02	2.2033E+03	3.1098E+02	7.1640E+03	2.2593E+02	3.2067E+03	2.9282E+02	2.6900E+03	6.0369E+02
F11	**7.9287E+01**	2.0868E+01	9.4038E+01	2.5082E+01	1.7590E+02	1.6468E+01	9.7308E+01	2.9810E+01	1.5874E+02	2.3347E+02
F12	2.6736E+05	1.3180E+05	2.9989E+05	1.1887E+05	1.8462E+06	3.0375E+05	**1.2777E+05**	8.1524E+04	1.7897E+06	2.2989E+06
F13	8.6092E+02	5.5899E+02	1.1783E+03	4.4436E+02	**7.4878E+02**	6.7608E+01	1.3349E+04	5.3217E+03	2.8120E+04	6.6856E+04
F14	1.6902E+04	1.2871E+04	1.5326E+04	1.2064E+04	**9.1439E+01**	5.1645E+00	5.3218E+03	3.0488E+03	2.5344E+04	3.3204E+04
F15	4.5481E+02	3.3126E+02	4.0246E+02	1.3926E+02	**1.3895E+02**	1.3121E+01	7.0068E+02	7.2068E+02	6.7004E+03	7.5244E+03
F16	4.7969E+02	1.3276E+02	**4.6570E+02**	1.3754E+02	1.6492E+03	1.5272E+02	1.2338E+03	2.3248E+02	1.0448E+03	3.1394E+02
F17	1.1478E+02	4.5853E+01	**1.0775E+02**	5.0417E+01	5.3564E+02	7.4791E+01	1.0655E+03	1.9931E+02	4.3446E+02	1.7931E+02
F18	1.0026E+05	4.4129E+04	9.4987E+04	3.5239E+04	**1.1820E+02**	8.8527E+00	5.9831E+04	1.4748E+04	1.9278E+05	4.1848E+05
F19	1.7392E+02	1.0199E+02	1.7206E+02	9.5874E+01	**4.5806E+01**	3.6850E+00	3.5238E+03	1.2483E+03	7.3666E+03	9.6109E+03
F20	**1.7098E+02**	7.8242E+01	2.1226E+02	5.9161E+01	6.0216E+02	1.0326E+02	8.5598E+02	2.2429E+02	4.6276E+02	1.7701E+02
F21	2.2647E+02	7.7721E+01	**2.1670E+02**	8.1303E+01	4.3440E+02	1.1808E+01	3.0985E+02	5.9021E+01	2.6360E+02	1.3127E+01
F22	1.1139E+02	1.5587E+00	1.1289E+02	1.2825E+00	2.4706E+02	3.8870E+01	**1.0000E+02**	3.9053E-09	9.2372E+02	1.4112E+03
F23	**4.1447E+02**	1.3447E+01	4.2387E+02	1.7311E+01	5.7713E+02	6.0703E+01	4.5683E+02	1.3328E+02	4.3849E+02	1.8589E+01
F24	**4.1772E+02**	1.7555E+02	4.8938E+02	1.6187E+02	6.8755E+02	1.4046E+01	5.2286E+02	3.5773E+01	5.4315E+02	3.8842E+01
F25	**3.8522E+02**	1.3514E+00	3.8667E+02	1.5149E+00	4.5099E+02	6.7690E+00	3.8995E+02	7.5869E+00	3.9981E+02	1.5786E+01
F26	**2.1980E+02**	4.7322E+00	2.3204E+02	3.3915E+01	1.7050E+03	1.2513E+02	2.5333E+02	5.0742E+01	1.8527E+03	1.0497E+03
F27	**5.1988E+02**	6.5450E+00	5.2294E+02	7.1198E+00	6.4745E+02	1.3231E+01	5.5121E+02	2.0816E+01	5.5785E+02	2.7177E+01
F28	4.1266E+02	5.1275E+00	4.1576E+02	5.4556E+00	5.5008E+02	1.6626E+01	**3.0725E+02**	2.8167E+01	4.2831E+02	5.6557E+01
F29	**6.0009E+02**	7.4391E+01	6.3692E+02	6.6334E+01	1.4098E+03	1.2188E+02	1.1500E+03	1.8828E+02	7.4341E+02	1.6579E+02
F30	**5.3005E+03**	1.0612E+03	8.3080E+03	1.9291E+03	7.4916E+04	2.0269E+04	7.9894E+03	2.6002E+03	2.7113E+04	1.7379E+04
W/T/L	-/-/-		20/10/0		24/1/5		17/4/9		18/6/6	

Table 5.2: Results obtained by the Wilcoxon test for algorithm HHS

VS	R^+	R^-	Exact P-value	Asymptotic P-value
SE06	359.0	106.0	0.008142	0.008997
DE/current/r1	363.0	102.0	0.006194	0.00705
HGSA	276.0	189.0	≥ 0.2	0.365462
GLPSO	377.0	88.0	0.002188	0.00286

Table 5.3: Parameter setting of algorithms.

Algorithm	parameters
HHS	$F0 = 2.5, \sigma = 0.5$
SE06	$\sigma = 0.1$
DE/current/r1	$F = 0.5 CR = 0.9$
HGSA	$G_0 = 100, L = 100, w_1(t) = 1 - w_2(t), w_2(t) = t^6/T^6, K \in [n, 2]$
GLPSO	$\omega = 0.7298, c = 1.49618, p_m = 0.01, s_g = 7$

Chapter 6

Conclusions

In this thesis, I studied two kinds of swarm intelligent algorithms. One is immunological algorithms which include clonal selection algorithm and quantum immunological algorithm. The other is evolutionary algorithm which involve ant lion algorithms, differential evolution algorithms, hypercube evolution algorithm, and spherical algorithms (especially the mechanisms used in particle swarm optimization). Several new mechanisms are proposed for the above mentioned algorithms to further improve their performance. Practical applications of optimization problems are carried out to verified their effectiveness. The contributions of this thesis can be summarized as follows.

(1) For one of the most popular immunological algorithm, i.e., clonal selection algorithm, I re-examine its basic theories and proposed two kinds of novel variants. The first one is a quantum clonal selection algorithm and its performance is confirmed by applying it on two typical combinatorial optimization problems. The main novelty of the proposed quantum clonal selection algorithm is the proposal of the combination of clonal selection theory and receptor editing, wherein a complete receptor editing operation based quantum clonal selection algorithm is proposed and a complete receptor editing operation based quantum interference crossover is introduced to alleviate the limitations of asexual proliferation during the immune maturation process. Extensive experimental results based on traveling salesman problems and holes machining path planning demonstrated the superiority of the proposed method in comparison with some other traditional immunological algorithms. On the other hand, I also

proposed another novel immunological algorithm based on clonal selection theory. The improved one is innovatively incorporated with a vaccine injection operator to maintain the diversity of B cells (i.e., solution population), and a Gaussian mutation operator which is aimed to enhance the algorithm to get out of the local optima once it is trapped. A more practical application is used, that is, to optimize the resource scheduling in cloud computing. The comparative results indicated that the proposed method is faster than other three typical algorithms.

(2) For evolutionary algorithms, I also studied two kinds of improved variants which is a hybridization of ant lion optimization with differential evolution, and another hybridization one by combing hypercube evolution with spherical evolution. Recently, hybridization of different algorithms is a common research framework for improving the search performance of algorithms according to the famous No Free Lunch Theorem. However, the questions of how to combine different algorithms and what kind of algorithms can be merged together to be an unified one, aiming to achieve better search performance, are still very challenging. In this thesis, I successfully give two examples of such hybridization which might give more insights into the above challenging issues.

In the future, I plan to apply the above improved algorithms to solve more kinds of combinatorial optimization problems, including set-union knapsack problem, generalized max-mean dispersion problem, equitable coloring problem, traveling repairman problem with profits, minimum differential dispersion problem, and so on.

Bibliography

[1] R. Xiao and Z. Tao, "Solution to holes machining path planning by evolutionary methods," *Computer Integrated Manufacturing Systems*, vol. 11, no. 5, p. 682, 2005.

[2] Z. Zhou and T. Ding, "Research on holes machining path planning optimization with tsp and ga," *Modular Machine Tool & Automatic Manufacturing Technique*, vol. 7, p. 008, 2007.

[3] J. B. Tenenbaum, C. Kemp, T. L. Griffiths, and N. D. Goodman, "How to grow a mind: Statistics, structure, and abstraction," *science*, vol. 331, no. 6022, pp. 1279–1285, 2011.

[4] E. Bonabeau, M. Dorigo, and G. Theraulaz, "Inspiration for optimization from social insect behaviour," *Nature*, vol. 406, no. 6791, p. 39, 2000.

[5] X. Hu and R. C. Eberhart, "Adaptive particle swarm optimization: detection and response to dynamic systems," in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, vol. 2. IEEE, 2002, pp. 1666–1670.

[6] P. B. Mullen, C. K. Monson, K. D. Seppi, and S. C. Warnick, "Particle swarm optimization in dynamic pricing," in *2006 IEEE International Conference on Evolutionary Computation*. IEEE, 2006, pp. 1232–1239.

[7] M. Mavrovouniotis and S. Yang, "Ant colony optimization with immigrants schemes in dynamic environments," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2010, pp. 371–380.

[8] W. Zhong, J. Xing, Y. Liang, and F. Qian, "Dynamic optimization with an improved θ-pso based on memory recall," in *2010 8th World Congress on Intelligent Control and Automation*. IEEE, 2010, pp. 3225–3229.

[9] T. Blackwell and J. Branke, "Multiswarms, exclusion, and anti-convergence in dynamic environments," *IEEE transactions on evolutionary computation*, vol. 10, no. 4, pp. 459–472, 2006.

[10] T. Smith, P. Husbands, and M. O'Shea, "Fitness landscapes and evolvability," *Evolutionary computation*, vol. 10, no. 1, pp. 1–34, 2002.

[11] P. A. Bosman and H. La Poutre, "Learning and anticipation in online dynamic optimization with evolutionary algorithms: the stochastic case," in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*. ACM, 2007, pp. 1165–1172.

[12] C. W. Lee, "Dynamic optimization of the grinding process in batch production," *Journal of Manufacturing Science and Engineering*, vol. 131, no. 2, p. 021006, 2009.

[13] H. Cheng and S. Yang, "Genetic algorithms with immigrants schemes for dynamic multicast problems in mobile ad hoc networks," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 5, pp. 806–819, 2010.

[14] J. H. Holland *et al.*, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[15] Y. Yu and Z.-H. Zhou, "A new approach to estimating the expected first hitting time of evolutionary algorithms," *Artificial Intelligence*, vol. 172, no. 15, pp. 1809–1832, 2008.

[16] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," in *Foundations of genetic algorithms*. Elsevier, 1991, vol. 1, pp. 69–93.

[17] A. Prügel-Bennett and J. L. Shapiro, "Analysis of genetic algorithms using statistical mechanics," *Physical Review Letters*, vol. 72, no. 9, p. 1305, 1994.

[18] D. H. Wolpert, W. G. Macready *et al.*, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67–82, 1997.

[19] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

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

[21] M. Črepinšek, S.-H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: A survey," *ACM Computing Surveys (CSUR)*, vol. 45, no. 3, p. 35, 2013.

[22] J. Liu, H. A. Abbass, D. G. Green, and W. Zhong, "Motif difficulty (md): a predictive measure of problem difficulty for evolutionary algorithms using network motifs," *Evolutionary computation*, vol. 20, no. 3, pp. 321–347, 2012.

[23] Y. Xu, L. Wang, S.-y. Wang, and M. Liu, "An effective teaching–learning-based optimization algorithm for the flexible job-shop scheduling problem with fuzzy processing time," *Neurocomputing*, vol. 148, pp. 260–268, 2015.

[24] G. Yang, S. Wu, Q. Jin, and J. Xu, "A hybrid approach based on stochastic competitive hopfield neural network and efficient genetic algorithm for frequency assignment problem," *Applied Soft Computing*, vol. 39, pp. 104–116, 2016.

[25] A. R. Yildiz, "A new hybrid artificial bee colony algorithm for robust optimal design and manufacturing," *Applied Soft Computing*, vol. 13, no. 5, pp. 2906–2912, 2013.

[26] 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.

[27] Z. Wu, L. Kang, and X. Zou, "A parallel global-local mixed evolutionary algorithm for multimodal function optimization," in *Fifth International Conference on Algorithms and Architectures for Parallel Processing, 2002. Proceedings.* IEEE, 2002, pp. 247–250.

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

[29] V. Hatzikos, J. Hätönen, and D. Owens, "Genetic algorithms in norm-optimal linear and non-linear iterative learning control," *International Journal of control*, vol. 77, no. 2, pp. 188–197, 2004.

[30] S. Gao, M. Zhou, Y. Wang, J. Cheng, H. Yachi, and J. Wang, "Dendritic neuron model with effective learning algorithms for classification, approximation, and prediction," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 2, pp. 601–614, 2018.

[31] J. Rayno, M. F. Iskander, and M. H. Kobayashi, "Hybrid genetic programming with accelerating genetic algorithm optimizer for 3-d metamaterial design," *IEEE Antennas and Wireless Propagation Letters*, vol. 15, pp. 1743–1746, 2016.

[32] S. Manaseer, W. Manasir, M. Alshraideh, N. A. Hashish, and O. Adwan, "Automatic test data generation for java card applications using genetic algorithm," *Journal of Software Engineering and Applications*, vol. 8, no. 12, p. 603, 2015.

[33] A. Pachauri and G. Srivastava, "Automated test data generation for branch testing using genetic algorithm: An improved approach using branch ordering,

memory and elitism," *Journal of Systems and Software*, vol. 86, no. 5, pp. 1191–1208, 2013.

[34] M. H. Moradi and M. Abedini, "A combination of genetic algorithm and particle swarm optimization for optimal dg location and sizing in distribution systems," *International Journal of Electrical Power & Energy Systems*, vol. 34, no. 1, pp. 66–74, 2012.

[35] J. F. Gonçalves and M. G. Resende, "A parallel multi-population biased random-key genetic algorithm for a container loading problem," *Computers & Operations Research*, vol. 39, no. 2, pp. 179–190, 2012.

[36] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins, "A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows," *Computers & Operations Research*, vol. 40, no. 1, pp. 475–489, 2013.

[37] S. Mallick, S. Ghoshal, P. Acharjee, and S. Thakur, "Optimal static state estimation using improved particle swarm optimization and gravitational search algorithm," *International Journal of Electrical Power & Energy Systems*, vol. 52, pp. 254–265, 2013.

[38] W.-f. Gao and S.-y. Liu, "A modified artificial bee colony algorithm," *Computers & Operations Research*, vol. 39, no. 3, pp. 687–697, 2012.

[39] 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.

[40] S. A. Hofmeyr and S. Forrest, "Architecture for an artificial immune system," *Evolutionary Computation*, vol. 8, no. 4, pp. 443–473, 2000.

[41] S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri, "Self-nonself discrimination in a computer," in *Research in Security and Privacy, 1994. Proceedings., 1994 IEEE Computer Society Symposium on.* IEEE, 1994, pp. 202–212.

[42] J. Timmis, "Artificial immune systems: A novel data analysis technique inspired by the immune network theory," Ph.D. dissertation, Department of Computer Science, 2000.

[43] Z. Tang, K. Tashima, and Q. Cao, "A pattern recognition system using clonal selection-based immune network," *IEICE Transactions on Information and Systems*, vol. 84, no. 12, pp. 2615–2622, 2001.

[44] C. Wu and C. Ku, "Stress-enhanced clonal selection algorithm for structural topology optimization," *International Journal for Numerical Methods in Engineering*, vol. 89, no. 8, pp. 957–974, 2012.

[45] X. Zuo and Y. Fan, "A chaos search immune algorithm with its application to neuro-fuzzy controller design," *Chaos, Solitons & Fractals*, vol. 30, no. 1, pp. 94–109, 2006.

[46] L. N. De Castro and F. J. Von Zuben, "Learning and optimization using the clonal selection principle," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 3, pp. 239–251, 2002.

[47] S. F. M. Burnet, *The clonal selection theory of acquired immunity.* Vanderbilt University Press Nashville, 1959.

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

[49] B. S. Rao and K. Vaisakh, "Multi-objective adaptive clonal selection algorithm for solving environmental/economic dispatch and opf problems with load uncertainty," *International Journal of Electrical Power & Energy Systems*, vol. 53, pp. 390–408, 2013.

[50] Y. Cai, J. Wang, J. Yin, and Y. Zhou, "Memetic clonal selection algorithm with eda vaccination for unconstrained binary quadratic programming problems," *Expert Systems with Applications*, vol. 38, no. 6, pp. 7817–7827, 2011.

[51] L. Hong and Z. Ji, "An adaptive multi-objective clonal selection algorithm," in *Computer Communication Control and Automation (3CA), 2010 International Symposium on*, vol. 1. IEEE, 2010, pp. 233–236.

[52] Y. Zhong, L. Zhang, and P. Li, "Multispectral remote sensing image classification based on simulated annealing clonal selection algorithm," in *Geoscience and Remote Sensing Symposium, 2005. IGARSS'05. Proceedings. 2005 IEEE International*, vol. 6. IEEE, 2005, pp. 3745–3748.

[53] 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.

[54] 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.

[55] A. R. Yıldız, "A novel hybrid immune algorithm for global optimization in design and manufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 25, no. 2, pp. 261–270, 2009.

[56] A. R. Yildiz, "A comparative study of population-based optimization algorithms for turning operations," *Information Sciences*, vol. 210, pp. 81–88, 2012.

[57] V. Kouskoff and D. Nemazee, "Role of receptor editing and revision in shaping the b and t lymphocyte repertoire," *Life Sciences*, vol. 69, no. 10, pp. 1105–1113, 2001.

[58] R. A. Goldsby, T. J. Kindt, B. A. Osborne, and J. Kuby, *Immunology*. WH Freeman, 2003.

[59] R. Pelanda and R. M. Torres, "Receptor editing for better or for worse," *Current Opinion in Immunology*, vol. 18, no. 2, pp. 184–190, 2006.

[60] L. K. Verkoczy, A. S. Mårtensson, and D. Nemazee, "The scope of receptor editing and its association with autoimmunity," *Current Opinion in Immunology*, vol. 16, no. 6, pp. 808–814, 2004.

[61] M. C. Nussenzweig, "Immune receptor editing: revise and select," *Cell*, vol. 95, no. 7, pp. 875–878, 1998.

[62] S. Tonegawa, "Somatic generation of antibody diversity," *Nature*, vol. 302, no. 5909, pp. 575–581, 1983.

[63] L. N. De Castro and F. J. Von Zuben, "The clonal selection algorithm with engineering applications," in *Proceedings of GECCO*, vol. 2000, 2000, pp. 36–39.

[64] A. S. Perelson, "Immune network theory," *Immunological Reviews*, vol. 110, no. 1, pp. 5–36, 1989.

[65] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys, "A guided tour of combinatorial optimization," *Estimation, Simulation, and Control–Wiley-Interscience Series*, 1985.

[66] G. Reinelt, "Tspliba traveling salesman problem library," *ORSA journal on computing*, vol. 3, no. 4, pp. 376–384, 1991.

[67] K. Maekawa, N. Mori, H. Tamaki, H. Kita, and Y. Nishikawa, "A genetic solution for the traveling salesman problem by means of a thermodynamical selection rule," *Transactions of the Society of Instrument and Control Engineers*, vol. 33, no. 9, pp. 939–946, 1997.

[68] P.-C. Chang, W.-H. Huang, and C.-J. Ting, "Dynamic diversity control in genetic algorithm for mining unsearched solution space in tsp problems," *Expert Systems with Applications*, vol. 37, no. 3, pp. 1863–1878, 2010.

[69] R. Xiao, Z. Tao, and T. Chen, "An analytical approach to the similarities between swarm intelligence and artificial neural network," *Transactions of the Institute of Measurement and Control*, vol. 34, no. 6, pp. 736–745, 2012.

[70] M. Liu, W. Zeng, and J. Zhao, "A fast bi-objective non-dominated sorting algorithm," *Pattern Recognition and Artificial Intelligence*, vol. 24, no. 4, pp. 538–547, 2011.

[71] S. H. H. Madni, M. S. A. Latiff, J. Ali *et al.*, "Multi-objective-oriented cuckoo search optimization-based resource scheduling algorithm for clouds," *Arabian Journal for Science and Engineering*, vol. 44, no. 4, pp. 3585–3602, 2019.

[72] Https://www.ibm.com/cloud/learn/cloud-computing.

[73] A. Gawanmeh, S. Parvin, and A. Alwadi, "A genetic algorithmic method for scheduling optimization in cloud computing services," *Arabian Journal for Science and Engineering*, vol. 43, no. 12, pp. 6709–6718, 2018.

[74] S. Parthasarathy and C. J. Venkateswaran, "Scheduling jobs using oppositional-gso algorithm in cloud computing environment," *Wireless Networks*, vol. 23, no. 8, pp. 2335–2345, 2017.

[75] A. Choudhary, I. Gupta, V. Singh, and P. K. Jana, "A GSA based hybrid algorithm for bi-objective workflow scheduling in cloud computing," *Future Generation Computer Systems*, vol. 83, pp. 14–26, 2018.

[76] K. Dubey, M. Kumar, and S. Sharma, "Modified heft algorithm for task scheduling in cloud environment," *Procedia Computer Science*, vol. 125, pp. 725–732, 2018.

[77] R. Bala and G. Singh, "An improved heft algorithm using multi-criterian resource factors," *Int. J. Comput. Sci. Inf. Technol.(IJCSIT)*, vol. 5, no. 6, pp. 6958–6963, 2014.

[78] B. Kanani and B. Maniyar, "Review on max-min task scheduling algorithm for cloud computing," *Journal of Emerging Technologies and Innovative Research*, vol. 2, no. 3, pp. 781–784, 2015.

[79] X. Li, Y. Mao, X. Xiao, and Y. Zhuang, "An improved max-min task-scheduling algorithm for elastic cloud," in *2014 International Symposium on Computer, Consumer and Control*. IEEE, 2014, pp. 340–343.

[80] E. De Coninck, T. Verbelen, B. Vankeirsbilck, S. Bohez, P. Simoens, and B. Dhoedt, "Dynamic auto-scaling and scheduling of deadline constrained service workloads on iaas clouds," *Journal of Systems and Software*, vol. 118, pp. 101–114, 2016.

[81] M. Kumar, K. Dubey, and S. Sharma, "Elastic and flexible deadline constraint load balancing algorithm for cloud computing," *Procedia Computer Science*, vol. 125, pp. 717–724, 2018.

[82] M. Kumar, S. Sharma, A. Goel, and S. Singh, "A comprehensive survey for scheduling techniques in cloud computing," *Journal of Network and Computer Applications*, vol. 143, pp. 1–33, 2019.

[83] A. Arunarani, D. Manjula, and V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey," *Future Generation Computer Systems*, vol. 91, pp. 407–415, 2019.

[84] S. Kaur and A. Verma, "An efficient approach to genetic algorithm for task scheduling in cloud computing environment," *International Journal of Information Technology and Computer Science (IJITCS)*, vol. 4, no. 10, pp. 74–79, 2012.

[85] K. Dasgupta, B. Mandal, P. Dutta, J. K. Mandal, and S. Dam, "A genetic algorithm (ga) based load balancing strategy for cloud computing," *Procedia Technology*, vol. 10, pp. 340–347, 2013.

[86] B. Keshanchi, A. Souri, and N. J. Navimipour, "An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: formal verification, simulation, and statistical testing," *Journal of Systems and Software*, vol. 124, pp. 1–21, 2017.

[87] K. KRISHNASAMY, "Task scheduling algorithm based on hybrid particle swarm optimization in cloud computing environment." *Journal of Theoretical & Applied Information Technology*, vol. 55, no. 1, pp. 33–38, 2013.

[88] A. Verma and S. Kaushal, "A hybrid multi-objective particle swarm optimization for scientific workflow scheduling," *Parallel Computing*, vol. 62, pp. 1–19, 2017.

[89] H. Duan, C. Chen, G. Min, and Y. Wu, "Energy-aware scheduling of virtual machines in heterogeneous cloud computing systems," *Future Generation Computer Systems*, vol. 74, pp. 142–150, 2017.

[90] D. P. Mahato, R. S. Singh, A. K. Tripathi, and A. K. Maurya, "On scheduling transactions in a grid processing system considering load through ant colony optimization," *Applied Soft Computing*, vol. 61, pp. 875–891, 2017.

[91] S. K. Addya, A. K. Turuk, B. Sahoo, M. Sarkar, and S. K. Biswash, "Simulated annealing based vm placement strategy to maximize the profit for cloud service providers," *Engineering science and technology, an international journal*, vol. 20, no. 4, pp. 1249–1259, 2017.

[92] L. Tang, Z. Li, P. Ren, J. Pan, Z. Lu, J. Su, and Z. Meng, "Online and offline based load balance algorithm in cloud computing," *Knowledge-Based Systems*, vol. 138, pp. 91–104, 2017.

[93] H. Dai, Y. Yang, H. Li, and C. Li, "Bi-direction quantum crossover-based clonal selection algorithm and its applications," *Expert Systems with Applications*, vol. 41, no. 16, pp. 7248–7258, 2014.

[94] 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.

[95] M. Rafiei, T. Niknam, and M. H. Khooban, "Probabilistic electricity price forecasting by improved clonal selection algorithm and wavelet preprocessing," *Neural Computing and Applications*, vol. 28, no. 12, pp. 3889–3901, 2017.

[96] T. Gong, "Improved immune computation for high-precision face recognition," *Soft Computing*, vol. 21, no. 20, pp. 5989–5999, 2017.

[97] Y.-C. Chou and M. Nakajima, "A clonal selection algorithm for energy-efficient mobile agent itinerary planning in wireless sensor networks," *Mobile Networks and Applications*, vol. 23, no. 5, pp. 1233–1246, 2018.

[98] X.-H. Liu, M.-Y. Shan, R.-L. Zhang, and L.-H. Zhang, "Green vehicle routing optimization based on carbon emission and multiobjective hybrid quantum immune algorithm," *Mathematical Problems in Engineering*, vol. 2018, 2018.

[99] A. Lasisi, R. Ghazali, and H. Chiroma, "Utilizing clonal selection theory inspired algorithms and k-means clustering for predicting opec carbon dioxide emissions from petroleum consumption," in *International Conference on Soft Computing and Data Mining*. Springer, 2016, pp. 101–110.

[100] Y. Zhong, L. Zhang, and P. Li, "Multispectral remote sensing image classification based on simulated annealing clonal selection algorithm," in *Proceedings. 2005 IEEE International Geoscience and Remote Sensing Symposium, 2005. IGARSS'05.*, vol. 6. IEEE, 2005, pp. 3745–3748.

[101] G. Chen and J. Liu, "Mobile robot path planning using ant colony algorithm and improved potential field method," *Computational intelligence and neuroscience*, vol. 2019, 2019.

[102] B. Akay and D. Karaboga, "A modified artificial bee colony algorithm for real-parameter optimization," *Information sciences*, vol. 192, pp. 120–142, 2012.

[103] L. N. De Castro and F. J. Von Zuben, "Learning and optimization using the clonal selection principle," *IEEE transactions on evolutionary computation*, vol. 6, no. 3, pp. 239–251, 2002.

[104] W. Zhou and J. Cao, "Cloud computing resource scheduling strategy based on prediction and aco algorithm," *Computer simulation*, vol. 29, no. 9, pp. 239–242, 2012.

[105] H. Yuan, C. Li, and M. Du, "Optimal virtual machine resources scheduling based on improved particle swarm optimization in cloud computing." *JSW*, vol. 9, no. 3, pp. 705–708, 2014.

[106] X. Wang, H. Gu, and Y. Yue, "The optimization of virtual resource allocation in cloud computing based on rbpso," *Concurrency and Computation: Practice and Experience*, p. e5113, 2018.

[107] Y.-K. Lin and C. S. Chong, "Fast ga-based project scheduling for computing resources allocation in a cloud manufacturing system," *Journal of Intelligent Manufacturing*, vol. 28, no. 5, pp. 1189–1201, 2017.

[108] T. Jena and J. Mohanty, "Ga-based customer-conscious resource allocation and task scheduling in multi-cloud computing," *Arabian Journal for Science and Engineering*, vol. 43, no. 8, pp. 4115–4130, 2018.

[109] A. S. Sofia and P. GaneshKumar, "Multi-objective task scheduling to minimize energy consumption and makespan of cloud computing using nsga-ii," *Journal of Network and Systems Management*, vol. 26, no. 2, pp. 463–485, 2018.

[110] Y. Dai, Y. Lou, and X. Lu, "A task scheduling algorithm based on genetic algorithm and ant colony optimization algorithm with multi-qos constraints in cloud computing," in *2015 7th International Conference on Intelligent Human-Machine Systems and Cybernetics*, vol. 2. IEEE, 2015, pp. 428–431.

[111] B. Hayes, "Cloud computing," *Communications of the ACM*, vol. 51, no. 7, pp. 9–11, 2008.

[112] R. N. Calheiros, R. Ranjan, C. A. De Rose, and R. Buyya, "Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services," *arXiv preprint arXiv:0903.2525*, 2009.

[113] M. M. Drugan, "Reinforcement learning versus evolutionary computation: A survey on hybrid algorithms," *Swarm and Evolutionary Computation*, vol. 44, pp. 228–246, 2019.

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

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

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

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

[118] 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.

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

[120] 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.

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

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

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

[124] J. Wang, L. Yuan, Z. Zhang, S. Gao, Y. Sun, and Y. Zhou, "Multiobjective multiple neighborhood search algorithms for multiobjective fleet size and mix location-routing problem with time windows," *IEEE Transactions on Systems, Man and Cybernetics: Systems*, 2019, doi: 10.1109/TSMC.2019.2912194.

[125] K. R. Opara and J. Arabas, "Differential evolution: A survey of theoretical analyses," *Swarm and Evolutionary Computation*, vol. 44, pp. 546–558, 2019.

[126] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2010.

[127] Y. Yu, S. Gao, Y. Wang, and Y. Todo, "Global optimum-based search differential evolution," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 2, pp. 379–394, 2018.

[128] G. Wu, X. Shen, H. Li, H. Chen, A. Lin, and P. N. Suganthan, "Ensemble of differential evolution variants," *Information Sciences*, vol. 423, pp. 172–186, 2018.

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

[130] Y. Wang, Z. Cai, and Q. Zhang, "Enhancing the search ability of differential evolution through orthogonal crossover," *Information Sciences*, vol. 185, no. 1, pp. 153–177, 2012.

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

[132] W. Gong, Z. Cai, and C. X. Ling, "DE/BBO: a hybrid differential evolution with biogeography-based optimization for global numerical optimization," *Soft Computing*, vol. 15, no. 4, pp. 645–665, 2010.

[133] W.-Y. Lin, "A GA–DE hybrid evolutionary algorithm for path synthesis of four-bar linkage," *Mechanism and Machine Theory*, vol. 45, no. 8, pp. 1096–1107, 2010.

[134] N. Awad, M. Ali, J. Liang, B. Qu, and P. Suganthan, "Problem definitions and evaluation criteria for the cec 2017 special session and competition on single objective bound constrained real-parameter numerical optimization," in *Technical Report*. NTU, Singapore, 2016.

[135] S. Mirjalili, "The ant lion optimizer," *Advances in Engineering Software*, vol. 83, pp. 80–98, 2015.

[136] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long, "A survey of clustering with deep learning: From the perspective of network architecture," *IEEE Access*, vol. 6, pp. 39 501–39 514, 2018.

[137] Mydata, http://yarpiz.com/64/ypml101-evolutionary-clustering.

[138] 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.

[139] 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.

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

[141] 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.

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

[143] 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.

[144] 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.

[145] 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.

[146] 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.

[147] 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.

[148] X. Yang, *Nature-inspired metaheuristic algorithms*. Luniver Press, 2010.

[149] 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.

[150] 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.

[151] 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.

[152] 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.

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

[154] J. Kennedy, "Particle swarm optimization," *Encyclopedia of machine learning*, pp. 760–766, 2010.

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

[156] S. A. Hofmeyr and S. Forrest, "Architecture for an artificial immune system," *Evolutionary Computation*, vol. 8, no. 4, pp. 443–473, 2000.

[157] S. Gao, Z. Tang, H. Dai, and J. Zhang, "A hybrid clonal selection algorithm," *International Journal of Innovative Computing, Information and Control*, vol. 4, no. 4, pp. 995–1008, 2008.

[158] 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.

[159] 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.

[160] D. Karaboga, "Artificial bee colony algorithm," *Scholarpedia*, vol. 5, no. 3, p. 6915, 2010.

[161] J. Ji, S. Song, C. Tang, S. Gao, Z. Tang, and Y. Todo, "An artificial bee colony algorithm search guided by scale-free networks," *Information Sciences*, vol. 473, pp. 142–165, 2019.

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

[163] 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.

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

[165] D. Mustard, "Numerical integration over the n-dimensional spherical shell," *Mathematics of Computation*, vol. 18, no. 88, pp. 578–589, 1964.

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

[167] Y.-J. Gong, J.-J. Li, Y. Zhou, Y. Li, H. S.-H. Chung, Y.-H. Shi, and J. Zhang, "Genetic learning particle swarm optimization," *IEEE Transactions on Cybernetics*, vol. 46, no. 10, pp. 2277–2290, 2015.

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

[169] 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.

[170] 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.