Bio-inspired Computational Algorithm and Its Applications to Optimization Problems

by

Zhe Xu

A dissertation

submitted to the Graduate School of Innovative Life Science

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Engineering

University of Toyama

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

2016

(Submitted September 13, 2016)

Acknowledgements

I would like to deeply thank all various people who, during my study and research, gave me with useful and helpful assistance. Without their care and consideration, this thesis would likely not have finished.

To my supervisor Prof. Zheng Tang at University of Toyama, who introduced me to the significant and fascinating world of Intelligent Soft Computing, for his support and continuous encouragement. Without his kind guidance and encouragement, I would never have completed this degree. Furthermore, his help and support are not limited in my study career, but also extended to my living life in Japan. Numerous stimulating discussions and supports make me go ahead during the past times since I came to Japan. Thanks to him, I could accomplish this thesis within three years.

I would like to deeply make thanks to my thesis referees, Prof. Shangce Gao at University of Toyama. As the first beginning TSP research, the work has been accomplished under his kind guidance and support. It is said that the first step costs troublesome. Without his help and support, I had no courage to start and finish the brain research which was at the start-up periods.

To all members in the Intelligent Information Systems Research Lab in University of Toyama, for all their suggestions to my research and the friendship that made my Ph.D. time very enjoyable and a final success graduate.

I would like to thank all the members of my family, for their unconditional love, support, and encouragement through this process, through all my study process. In particular, I would like to offer thanks to my wife Yanting Liu, who endured my seemingly endless hours of absorption in this effort without complaint, who gave me her unwavering support, and who took care of many of those "nuisance" items usually referred to as "Real Life"

when I was off in my other world, that of completing this endeavor.

Abstract

Bio-inspired meta-heuristics are the study of investigating biological mechanisms and thereafter modeling such living mechanisms and theories and their live by computer simulations. A large number of papers have focused on the character is tics of the swarm behaviors, such as birds, particles, fishes, human brains, as well as other insects including mosquitoes, because of their incredible abilities to solve a lot of very practical engineering and optimization problems. These problems are very difficult to be solved and some of those have proved to be NP-hard or NP-complete. That is to say, no polynomial time algorithm can be designed and used to solve such problems.

Based on the above research background, in this thesis, I am devoting to studying a number of meta-heuristics and applying them to solve some important practical problems. Some typical meta-heuristics involving the genetic algorithm, genetic programming, ant colony optimization, particle swarm optimization, differential evolution algorithms, artificial immune algorithms, gravitational search algorithm, and some others which will not be introduced in this thesis in details. Those who are interested in these algorithms may refer to my cited literatures.

The thesis is organized as in the following.

In chapter 1, we first introduced some basic concepts and theories of the bio-inspired meta-heuristics. The mechanisms of the biologically inspirations and the framework of how to design such meta-heuristics are brief summarized and introduced. Moreover, it studies the field consist of all their social behavior and the connections among their behaviors. Briefly the utilization of simulation by computers to model the living mechanisms and for improve the usage effectiveness of such simulations is important. Bio-inspired meta-heuristics is an interdiscipline which is composed by a lot of different research fields,

including computer science, artificial intelligence, applied mathematics, biological theories, physical phenomena, genetics, and some others. The powerful learning abilities and evolutionary capacities of biological systems are motivating us to design more robust and strong computational intelligent systems to solve the practical problems. These problems are becoming much harder to be revolved due to its dynamic environment, complex variable dependent relationships and time-related factors. Thus, the fundamental concepts and related researches of the background, together with the research purpose are summarized in this chapter.

In chapter 2, we focus on bio-inspired meta-heuristics, especially its computational framework. We introduced bio-inspired computations from the following three parts: the most famous Evolutionary computation (EC), the recent developed Ant colony optimization (ACO) and the novel introduced Artificial immune system (AIS). In this chapter, the basic descriptions including their biological inspirations, algorithmic modelling, computational systems and typical applications of EC, ACO, and AIS are presented.

In chapter 3, a new hybrid method by incorporating EDA into IA is proposed in order to solve the TSPs. In this method, EDA is used to realize the information exchanging during different solutions generated by IA through the probabilistic model. Thus, EDA enables IA to be quickly convergent to promising search areas. To refine the solutions sampled by EDA, a heuristic local search operator is also proposed to repair the infeasible solutions, and further facilitate the search by making use of the problem-dependent knowledge of TSP.

In chapter 4, another computational algorithm using particle swarm optimization (PSO) and a probability model (PM was proposed and presented to solve the well-known graph planarization problem (GPP). GPP is a traditional combinatorial optimization problem which is deeply related with circuit board layout problem, VLSI desing, automatic graph drawing problem, and some other graph-based problems. Moreover, it has significant theoretical importance related with networks design and analysis, computational geometry, and some other topological problems. Generally speaking, GPP is required to carry out two tasks involving a maximum planar subgraph acquisition and a plane embedding problem. The former is to find a maximum planar subgraph with a minimum cardinality subset of

edges which can be removed from the original graph, and the later is to draw the remaining subgraph on a plane such that no two edges intersect with each other except a common endpoint. The proposed PMPSO has demonstrated to be effective to find near optimal solutions for GPP than the previously proposed methods.

In chapter 5, Ant colony optimization (ACO) is one of the best heuristic algorithms for combinatorial optimization problems. Due to its distinctive search mechanism, ACO can perform successfully on the static traveling salesman problem(TSP). Nevertheless, ACO has some trouble in solving the dynamic TSP (DTSP) since the pheromone of the previous optimal trail attracts ants to follow even if the environment changes. Therefore, the quality of the solution is much inferior to that of the static TSP's solution. In this paper, ant colony algorithm with neighborhood search called NS-ACO is proposed to handle the DTSP composed by random traffic factors. ACO utilizes the short-term memory to increase the diversity of solutions and three moving operations containing swap, insertion and 2-opt optimize the solutions found by ants. The experiments are carried out to evaluate the performance of NS-ACO comparing with the conventional ACS and the ACO with random immigrants (RIACO) on the DTSPs of different scales. The experimental results demonstrate our proposed algorithm outperforms the other algorithms and is a competitive and promising approach to DTSP.

In chapter 6, there are conclusions and future Works. We proposed IA-EDA, PM-PSO and NS-ACO. They can be concluded that these new models can produce better solutions than traditional model before. And in the future, I will go a step further on various kinds of algorithm of Bio-inspired computation. Then, improve the performance of the current existent Bio-inspired computation and apply them to solve engineering problems in new fields. Last but not least, Bio-inspired computation can combine with other computational intelligence algorithms for solving much complex problems.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Bio-inspired meta-heuristics are the study of investigating biological mechanisms and thereafter modeling such living mechanisms and theories and their live by computer simulations. A large number of papers have focused on the character is tics of the swarm behaviors, such as birds, particles, fishes, human brains, as well as other insects including mosquitoes, because of their incredible abilities to solve a lot of very practical engineering and optimization problems. These problems are very difficult to be solved and some of those have proved to be NP-hard or NP-complete. That is to say, no polynomial time algorithm can be designed and used to solve such problems.

The mechanisms of the biologically inspirations and the framework of how to design such meta-heuristics are brief summarized and introduced. Moreover, it studies the field consist of all their social behavior and the connections among their behaviors. Briefly the utilization of simulation by computers to model the living mechanisms and for improve the usage effectiveness of such simulations is important. Bio-inspired meta-heuristics is an interdiscipline which is composed by a lot of different research fields, including computer science, artificial intelligence, applied mathematics, biological theories, physical phenomena, genetics, and some others.

The powerful learning abilities and evolutionary capacities of biological systems are motivating us to design more robust and strong computational intelligent systems to solve the practical problems. These problems are becoming much harder to be revolved due to its dynamic environment, complex variable dependent relationships and time-related factors. Thus, the fundamental concepts and related researches of the background, together with

the research purpose are summarized in this chapter.

1.1 Bio-inspired meta-heuristics

Bio-inspired meta-heuristics is inspired by biological mechanisms and theories. The objective of bio-inspired meta-heuristics is to design and develop computational intelligent systems by acquiring ideas from the biological phenomenons. It studies the field consist of all their social behavior and the connections among their behaviors. Briefly the use of computer simulation for modelling the living mechanisms and for improving the effectiveness of the designed computational tools. Bio-inspired computation is inspired from many kinds of natural metophors, including evolutionary theories, swarm intelligence, gravitational law, ants's food seeding mechanisms, particle corporation rules, and so on. The development of such meta-heuristics will help scientists and engineer design more powerful tools for those problem arisen from practical life. The emergence of such computational models and optimizers will give potential application on machine learning, big data analysis, Internet of Things (IoT), artificial intelligence, patter recognition, and other engineering applications.

The most characteristics of meta-heuristics is its wide applications and its self-adaption over various kinds of parameters in it. The properties of meta-heuristics is quite different from the common artificial intelligence (AI). In the traditional AI, programmers are the creators who design and write source code (i.e., programming) in advance to make the machine with some intelligence. The rules in AI is still-unknown. However, bio-inspired methodologies often include the method of specifying a set of simple rules which is used to control the following actions of the programming. Such a set of simple agents which obey those rules is available for generating intelligence. And usually there is a method which is designed in advance by the scientists to iteratively utilize these rules. That is to say, bio-inspired meta-heuristic is a specified bottom-up, decentralized approach, which can enable the constructed simple system to be a more complex one.

Complexity is often completely counter-intuitive from what the original rules would be expected to produce. Many optimization algorithms and models have been proposed

and developed in the literature. The models simulate the amazing characteristics of the nature, such as its collective behavior of decentralized, distributed population structure, self-organized mechanisms, and others. The resultant models are thus named as swarm intelligence or bio-inspired computational algorithms from the perspective of the application.

On the one hand, the swarm intelligence generally indicates a kind of capacity of solving problems that arisen from the interactions among agents of the population. These single agents are organized by following simple rules defined by the swarm intelligence. Moreover, the term swam, to be specific, indicates stochasticity, multiplicity, self-organization, randomness and messiness. The term intelligence represents that the emerged problem solving method is with intelligence to handle with problems on hand successfully.

On the other hand, the swarm behavior is regarded as in those natural phenomena like fish schools, bird flocks, mosquitoes insects, and others. For example, when a boock of birds try to change their direction of flight, they behave so neat and seem to be a single coherent entity instead of many individuals. Each agent in the population are trying to maintain a minimum and certain distance from other agents in the population during the evolution or moving periods, thus to make the entire population have sufficient diversity to keep elite genes to the next generation. The rule in such a population has the highest priority, and in nature it is usually confirmed by a frequently appeared behavior which is observed from animals, human beings, or insects. If an agent is not acting an avoidance behavior, it might trend to align themselves with their neighbors to avoid being isolated from others.

Furthermore, a swarm is generally considered as a collection of individuals which cooperate with each other, aiming to achieve the common goals such as the resolving of problems. This collective intelligence emerges as a large group, and such a group is composed by many relatively simple single individuals. These single individuals are controlled to or self-adaptively make use of simple local rules to control their own actions and movement. Via using these single actions and thus the reaction to the environment, the swarm achieves its objectives eventually. Most importantly, it is worth emphasizing that no central controller or rules exist in the swarm, which indicates that non central manipulation in the colony is performed. On the contrary, such high achievement of the whole swarm is real-

ized by taking advantage of all single individuals, which has a simple and random behavior of its perception in the nearby environment. It also should be noted that the local rules generated by single individuals are with no relations to the total global rules of the swarm. The above mentioned complex behavior of the swarm is thus realized by the interaction among these individuals and the emerged interaction network has a capacity of generating such powerful behavior for the swarm. As a result, the complex behavior of the whole swarm can deal with more complicated situations or problems to generate promising solutions.

Among all these characteristics, a so-called self-organization is the most attractive properties of the swarm. Self-organization is generally regarded as an essential feature for a swarm, and it usually help generate global (or macroscopic) response, rather than a local (or microscopic) one. In the previous research, Bonabeau et al. suggested that the self-organized ability of a swam have four kinds of characteristics. These are (1) positive feedback, (2) negative feedback, (3) fluctuations, and (4) multiple interactions. The distinct four feature of a swarm can be interpreted as in the following.

1. The positive feedback is the most important feature of a swarm. It is usually a simple behavior or generation rule which can promote the creation of the population structures via some feedback mechanisms. Lots of such feedback mechanisms are investigated by scientists. Among them, the recruitment and reinforcement have attracted much attentions. For example, the trail laying and trajectory tracking of some species such as ants, or the dance behaviors of bees. These behaviors are some typical examples of the positive feedback.

2. The negative feedback is also an essential feature for a swarm. In order to avoid the saturation which might occur in terms of available foragers, food source exhaustion, crowding, or competition at the food sources [1].

3. Not only the regular behaviors, but also the fluctuations are crucial for the innovation and creation of systems. These fluctuations include systemic errors, random walks, stochastic task switching rule among several agents.

4. It is widely accepted that the randomness if a key to the emergence of structure

because such mechanism enables the systems to generate new solutions or patterns. The randomness also come from the interaction among different individuals. Such interactions take place in a swarm and thus make the swam utilize the information of the whole interaction network. The data or information is spread in the whole network via the interactions.

In the literature, Millonas [2] investigated the swarm and conclude five basic principles which are essential to make a swarm be intelligent. The five principles include: (1) A proximity principle which means that a swarm must be capable of performing computations on not only the space but also the time; (2) A quality principle which shows that a swarm must has an ability of generating satisfying solutions for problems and responding to all quality factors arisen from the surrounding environment; (3) A diverse response principle which presents such a rule that a swarm should not distribute all its resources to a single place, such as along a excessively narrowed channel, but such a rule that it must allocate its resources to as many as places; (4) A stability principle which suggests that a swarm should keep stable during its implementation. It is not desired that a swarm change its common behavior once the environment changes; and (5) A adaptability principle which guarantees the effectiveness of the problem-solving performance of a swarm. It means that once the investment of energy or money, or something else is worth the computational burden, the swarm must be capable of making such a response of changing.

Moreover, the characteristics or principles of a swarm described above have been modelled from two aspects: one is a low viewpoint and the other is a higher one. These models also have attracted much attentions from many scientists to motivate them to propose more and more problem-solving method with the methodology of swarm intelligence. Such problem-solving methods have been applied on very difficult real world problems, including network security, pattern recognition, data mining, combinatorial optimization, scheduling, planning, engineering designs, network routing, etc.

1.2 Computational Intelligence

In recent years, many research has been done on applying the information processing performance of living organisms to computer systems, which leads to the foundation of a new academic subject called Artificial Intelligence. The artificial intelligence is named AI briefly.

The study of artificial intelligence can be divided into two groups. One group is the Good old fashioned AI which mainly consists of the expert system and the case-based reasoning. The other one in artificial intelligence is the Computational Intelligence which contains the artificial neural network, the fuzzy control, the evolutionary computation, the artificial immune system, and so on.

The expression of computational intelligence is usually defined as the capacity of a computer to learn a specific task or realize a particular objective from data and experimental observations. However, the computational intelligence is generally regarded as a synonym of machine learning in artificial intelligence community. They have its own distinct characteristics. Many definitions of computational intelligence have been introduced in the literature. The most widely accepted one is described by Bezdek [3], who interprets computational intelligence as follows.

"A system is called computationally intelligent if it deals with low-level data such as numerical data, if it has a pattern-recognition component and if it does not use knowledge as exact and complete as the Artificially Intelligent one [4]".

Usually, computational intelligence is considered to be a set of problem-solving algorithmic methods which are inspired by nature. These computational intelligent methods are especially useful for those hard problems such as NP-hard problems, which the traditional methods fail to solve. The reason seems to be two-fold: (1) The procedures seems to be too complicated for traditional mathematic methods that needs rigorous interpretations. The problem in hand may include uncertainties in itself, more worse, its implementation procedure may inherent be random [4]. Such a problem is hard to be solved using a traditional mathematic method. Moreover, as a matter of fact, most of real world problem is hard to be transformed into a binary encoded problem which the computer system is usually dealing

with. For those multiple-valued or continuous valued encoding problems, computational intelligence thus provide promising alternative methods for effectively solving them.

But generally, computational intelligence is a set of natural inspined computational methodologies and approaches to solve complex real-world problems which mathematical or traditional model can be useless for a few reasons: the processes might be too complex for mathematics, it might contain some uncertainties during the process, or the process might simply be stochastic in nature. In fact, many real-life problems cannot be translated into binary language (unique values of 0 and 1) for computers to deal with it [5]. Computational Intelligence therefore provides solutions for such problems [4].

There are many kinds of computational intelligent models which have been proposed in the literature, including evolutionary computation, artificial immune system, swarm intelligence, and other bio-inspired meta-heuristics. Nevertheless, the main principles and implementation procedures of these variants of computational intelligence are similar. For example, all of these methods are based on a population, and needs a number of evolutionary iterations (i.e. generations) to gradually improve the obtained solutions for a problem. And all these methods can solve many kinds of real world problems, such as image processing, natural language processing, optimization, computer security, IoT, and so on. In a word, it is hard to clearly distinct these different computational intelligent methods [6]. In addition, computational intelligence is expected to perform as natural animals including human beings. As a fact, the term of intelligence is generally related to human being. Recently, many sale products and items are also named as “intelligent”, which is directly linked to the decision making or decisions [7]. Moreover, a good news is that, according to the No-Free-Lunch theory, it is evident that more different computational intelligent methods are needed to be designed and further developed.

Chapter 2

Bio-inspired Computational Algorithms

Bio-inspired computational algorithms are usually shortly called as bio-inspired computation, and it generally include the evolutionary computation, ant colony optimization, particle swarm optimization, and artificial immune systems. As a result, the bio-inspired computation is generally considered as a total field of investigation which loosely knits the above mentioned subfields together. Moreover, the bio-inspired computational algorithms are tightly related with the community of artificial intelligence, especially the machine learning research field. Both of the bio-inspired computational algorithms and machine learning are based on some concepts and mechanisms of biology, mathematic theories and models, and computer science. To make it more specific, the bio-inspired computational algorithms are motivated from living phenomena, and thereafter modelled as computational methods. The bio-inspired computational algorithms are no doubt a major research subject in the natural computation [8].

2.1 Evolutionary computation

The most famous bio-inspired computational algorithm is the evolutionary computation (EC) which includes genetic algorithms, genetic programming, evolutionary strategies, and differential evolution. Without distinguishing the specific differences among those variants of EC, the general concepts and mechanisms of EC are introduced.

Generally, EC follows the main principle in nature which is called “survival of the fitness”. Such a principle demonstrates that the weak one in a population must be eliminated

to save the energy investment. The typical evolution can be described as in the following. First, a survival is realized via gene reproduction in the natural evolution process. Two parents can reproduce one or more offspring. The offspring are thus survived into the next generation by carrying their parents' promising features. On the other hand, those agents whose gene is not so good to be survive into the next generation will be delete from the population. Not only our human being, the animals in nature are also obeying such "survival of the fitness". For example, some bird species will kick all their siblings out from its nest to make all resources under manipulated by only one. The one thus is able to get more food and become more stronger, and eventually produce offspring to enter into the next generation [9].

From a computational perspective, EC makes use of a set of individuals, which is usually called a population. Thus, EC is one of the population based meta-heuristics. Each individual in the population is called a chromosome which displays all the features of the individuals during the evolution. The gene in the chromosome can represent the feature (or characteristic) of a chromosome which can be reserved by reproducing offspring. Mathematically, the values of these genes are defined as alleles. In each generation (or iteration, from a algorithmic viewpoint), the parents are scheduled to generate offspring. Such a competitive schedule is called selection in EC. The most famous selection scheme is elite selection [10]. The elite selection means that the individual with the best fitness in the environment will have a largest probability of being selected to produce offspring. In the traditional EC, two individuals (i.e. parents) are selected and exchange genes with each other. This procedure is called crossover. In addition, to simulate the learning capacity of human beings, each individual is also undergo a process called mutation. The mutation (or learning, from some machine learning techniques' viewpoint) can alter the gene of chromosome and thus change the corresponding allele. Such an allele is calculated using a fitness evaluation function. The fitness is used to evaluate the degree that a chromosome is fitting the problem in hand to be solved, and also it can be used to measure the survival strength of an individual. After a generation of evolution, individuals will be manipulated to undergo pruning, then some elite individuals are able to survive to enter into the next generation [11].

In addition, the behavioral features sealed in the phenotypes are also capable of affecting the evolutionary directions from two aspects. One is that phenotypes can affect the genetic modifications, and the other is that the behavioral features can grow up independently.

EC have some different kinds of developed models [12–14]:

1. Genetic algorithms are used to simulate genetic evolution mechanisms. The genetic algorithms is summarized by Holland in 1975, and they are the most famous ECs and have been used in a large number of applications. The generic procedures of a genetic algorithm include encoding, fitness evaluation, initialization, selection, and reproduction. It needs to be noted that there are various selection methods for genetic algorithm. Some typical ones are: random selection, proportional selection, tournament selection, rank-based selection, Boltzmann selection and $(\mu + \lambda)$- selection.

   Also several stopping conditions of the algorithms are also found in the literature, including: (1) The algorithm will be terminated when no improvement of the fitness can be obtained over a number of consecutive iterations. (2) The algorithm will stop when no changes occur in the entire population. (3) The algorithm will terminate if a previously defined solution can be obtained. (4) The algorithm is scheduled to stop once the global optimal solution is found.

   There are a number of updating methods of individuals for genetic algorithms, involving (1) the replacement of the worst individual in the population; (2) the replacement of a random individual in the population; (3) a kill tournament scheme; (4) the replacement of the oldest individual; (5) conservative selection; (6) the elitist selection; and (7) a parent and offspring competition strategy.

   Several variants of genetic algorithms have been proposed in the literature, including (1) a messy genetic algorithm; (2) an interactive evolution; (3) an island genetic algorithm; and (4) niching genetic algorithms.

2. Genetic programming makes use of the tree-based data structure to represent a chromosome and it is very similar to the genetic algorithm. Genetic programming was

designed to evolve executable computer programs. The tree representations have distinct characteristics including: (1) adaptive individuals which indicate that the size and shape of a chromosome can be changed if necessary, and (2) Domain-specific grammar which means that the grammar can be accurately defined.

3. Evolutionary programming comes from the simulation of phenotypic evolution to utilize its most promising feature, i.e., the adaptive behavior. It originated from the study result of L. Fogel in 1962. The basic evolution programming makes use of four components, i.e., the initialization, the mutation, the evaluation, and the selection.

   The most distinct feature of evolutionary programming is its mutation strategies, including (1) an uniform mutation, (2) a gaussian mutation, (3) a cauchy mutation, (4) a Levy mutation, (5) an exponential mutation, (6) chaotic mutations, and (7) their combined mutations.

   A number of variants of evolutionary programming are also proposed in the literature. Some of them can be listed in the following: (1) the classical evolutionary programming, (2) a fast evolutionary programming, (3) an exponential evolutionary programming, (4) an accelerated evolutionary programming, (5) a momentum evolutionary programming, (6) evolutionary programming with a local search, (7) evolutionary programming with extinction, (8) a hybridization with particle swarm optimization.

   It also has been applied on a number of fields, such as Bayesian network training, controller design, robotics, games, image processing, power systems, scheduling and routing, model selection, and design.

4. Evolution strategies are geared to generate various crossover, selection, mutation strategies in the evolutionary progress. The first (1+1) evolution strategy was developed for experimental optimization in hydrodynamical problems. A typical difference between evolution strategy and other EC is that the changes derived from the mutation are only accepted when a successful improvement of fitness is achieved. The components of initialization, recombination, mutation, evaluation, and selection

are the main procedures in evolution strategies.

Evolution strategies are also applied on various problems, including parameter optimization, controller design, induction motor design, neural network training, transformer design, computer security, and power systems.

5. Differential evolution is similar to genetic algorithm, and is a random, population based evolutionary strategy. It differs from the genetic algorithm in terms of the reproduction strategy. The differential evolution utilizes a "one-to-one" selection scheme. It has been applied on the following problems: clustering, controllers, filter design, image analysis, integer programming, model selection, neural network training, scheduling, and system design.

6. Cultural algorithms make use of the cumulative deposit of knowledge to model the evolution of culture. It was designed by Reynolds in the early 1990s. The belief space is a distinct characteristics of the cultural algorithms which means the knowledge repository. The collective patterns of the agents in the cultural algorithms are maintained and memorized in such a knowledge repository. The knowledge at least contains two main components, i.e., a situational and a normative knowledge component, respectively. More kinds of knowledge such as domain, history, topographical ones can be incorporated into the algorithm as well.

7. Coevolution refer to two different agents (or individuals) evolve together via competition or cooperation. Such competition and cooperation can help both agents acquire more diverse features to survive and eventually get better performance for solving problems.

   There are many types of coevolution, such as competition, amensalism, mutualism, commensalism, parasitism, etc. The competitive fitness plays important roles in the algorithm, where there are many kinds of fitness samplings, including (1) all versus all sampling strategy, (2) random sampling scheme, (3) tournament sampling method, (4) all versus best sampling strategy, and (5) shared sampling method. In addition, the fitness evaluation involves (1) a simple fitness evaluation method, (2)

a fitness sharing method, (3) a competitive fitness sharing method, and (4) a tournament fitness method.

Coevolution algorithm also has been applied on many problems, including game learning, iterated prisoners dilemma, military tactical planning, drug design, controller design, scheduling, neural network training, robot control, autonomous vehicles, evolving marketing strategies, constrained optimization, and rule generation for fuzzy logical controllers.

2.1.1 Genetic Algorithms (GA)

GA is a well-known intelligent meta-heuristics that originated from machine learning. Its evolutionary behavior is simulating those rules and theories in the natural evolutions and makes use that as a metaphor. The genetic algorithm's most important component is a population of chromosomes. It is realized by creating a colony of chromosome (also called individuals from an algorithmic perspective). The data structure of genetic algorithm is generally using a set of genes which is used to represent the characteristic or feature of each chromosome. Each chromosome will be manipulated to undergo a number of generations, where in each generation the chromosome will be selected, mutated, under crossover, and updated before entering into the next generation. Such a cycle is to simulate the natural evolution mechanism.

Till now, genetic algorithm has been successfully applied on an amount of different applications. It is not the objective of this thesis that making a comprehensive summary of the description and applications of introduced algorithms. Alternatively, some typical one will be emphasized to make readers clearly understanding an algorithm. Based on this consideration, the multi-modal optimization problems which are solved by genetic algorithms are used as a good example [15]. Such a multi-modal optimization problem contain more than one bits which are needed to be optimized. To successfully optimize all those bits, it is a usual method to encode each parameter in the problem via a string of the chromosome. Then the string of chromosome will be optimized generation by generation.

Although there are many investigation results which suggest that variable length of

chromosome might give some advantages over the problem solving, it is more general that the utilization of a fixed length of chromosome, especially for the optimization problems because the number of parameters in the problem is fixed and known in advanced. After determining the date structure for genetic algorithm, the algorithm will be carried out in a cyclic manner, and it will continue until a termination condition is fulfilled.

The overall framework of Algorithm of GA is shown in Fig. 2.1.

The general implementation of a genetic algorithm can be summarized as follows:

First, the fitness is calculated for all the chromosomes in the population, the fitter a chromosome, the better a solution. Second, the current chromosomes are evaluated and compared. According to the used selection strategy, parent chromosomes are selected to generate offspring. Meanwhile some of these chromosomes are undergone mutation operators. Then the offspring and the parent chromosomes are compared. Better ones are survived to enter into the next iteration. Such an iteration is called a generation for evolution. As a matter of fact, each single chromosome might fail to be an optimal solution for the problem needs to be solved. But the elite selection scheme of genetic algorithm makes sure that the average fitness of a following generation is better than that of the former generation. It is important for genetic algorithm to be able to find the global optimal solutions eventually [15].

2.1.2 Genetic Programming (GP)

Genetic programming was developed to be applied on automatic programming and program induction. Genetic programming has deeply relationship with the former introduced genetic algorithms, and it usually can be regarded as a specialized GA by the modifications of selection strategies, crossover and mutation operators, and updating schemes. However, the most biggest difference between a genetic programming and a genetic algorithm is the data representation method.

The genetic programming makes use of a tree-based representation for the problem and it does not make distinguishment for the solution search space and the chromosome representation space, which is quite different from other ECs. In other evolutionary com-

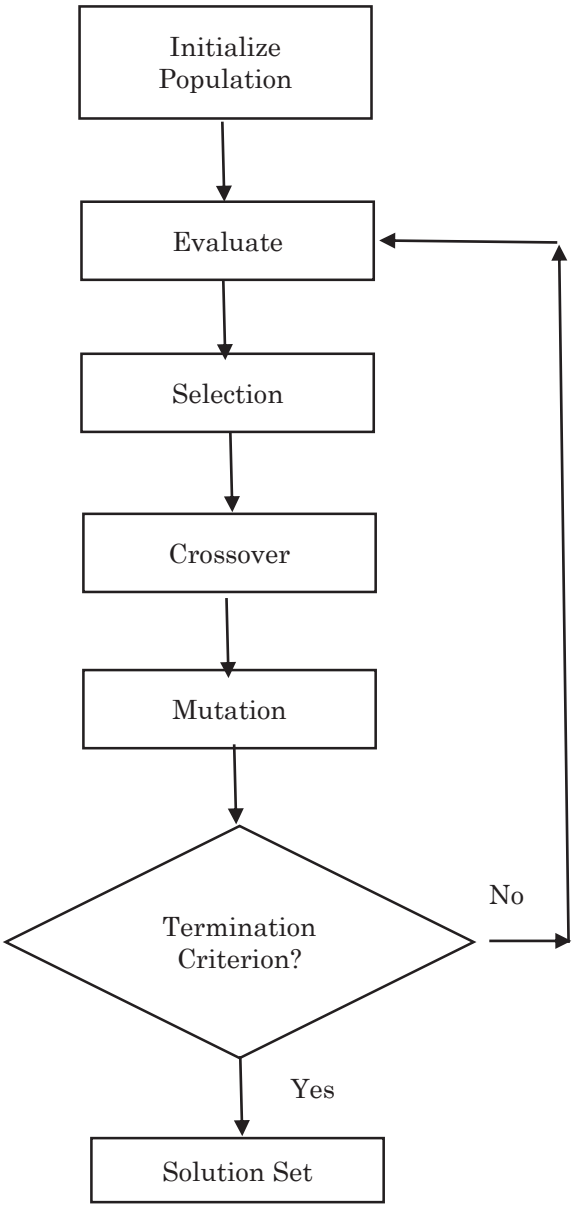

Figure 2.1: The overall framework of Algorithm of GA.

putational algorithms, the solution search space and the chromosome representation space are generally distinguished by using the concepts of genotype and phenotype, respectively. It is very easy in other evolutionary computational algorithms to make such a difference, e.g., by the utilization of a kind of simplest form, the one-to-one mapping scheme [16].

As a result, the representation of the genetic programming include not only the solution search space, but also the the chromosome representation space. In other words, it is possible in the genetic programming to synchronously evolve both representations. By doing so, the computational effectiveness of genetic programming can be improved. To be specific, the theoretical search space in GP is composed by the all possible compositions of the solution search space and the primitive set of all symbols used in the programs. These symbols in the programs are expressed using either a tree or a linear array, according to the needs of the problem.

In a classic genetic programming, the crossover operator is the most important operator using implementation. Compared with the mutation operator which just select some subtrees to make random changes using another randomly generated subtree, the crossover is capable of exchange information among different individuals. Such interchanges of subtrees among different individuals have an advantage of no disruption of the programs' syntax [17].

Generally speaking, the genetic programming can evolve symbolic representation for a functional language, such as the LISP [18]. The structure behind the search of the genetic programming is crucial for the final performance of the algorithm. It can be described as follows:

1. An evolved program contains several code segments. These code segments will not effect the final performance generated by the whole program if they are deleted from the whole one. These codes are also called semantically redundant ones, and the related segments above can be considered as introns.

2. The growth of the evolution in genetic programming will not stop until a user-defined maximum tree depth is reached or the evaluated fitness for the problem remains unchanged for several iterations. In other words, such evolved process is able to in-

crease indiscriminately. This process is also called bloat.

3. In genetic programming, the biggest serious limitation is the bloat. As the bloat not only influence the fitness evaluation, but also case a reduction of effective search operators and meanwhile needs much time to be implemented. That is to say, the bloat is very time-consuming. The values of the fitness almost always stagnates once the bloat occurs.

4. Within the implementation of the programming language, i.e. LISP [19], the programs are constituted by using parse trees based data structure, rather than the linear array of codes. It thus makes the distinct characteristics for LISP. As a result, a number of genetic programming users tend to utilize the LISP.

5. Nevertheless, there is an implementation problem in details. As noticed by many researchers, it is more simple and straightforward to run genetic programming in a programming environment without using the LISP. The programs consist of many elements which are taken from the functional sets together with the terminal sets. The functional sets and terminal sets are those typically sets fixed for the symbols, which are selected to be an appropriate solution for the problem on hand (or, the problem under the area of interest).

In the following, more details of the crossover operations in genetic programming are going to be elaborated. In a genetic programming, the crossover operator is implemented via selecting randomly generated subtrees among other chromosomes (or, individuals) to be exchanged with each other. The selection probabilities are determined based on a fitness evaluation function. The fitter the subtree, the higher it will be selected. Moreover, it should be emphasizing that the genetic programming does not make use of any mutation operator in its evaluation, not like the genetic algorithms do. However, genetic programming still utilize various developed genetic operators, which are mostly problem-related, or based on some problem inherent knowledge. As an example, the automatically defined functions usually accept those definitions of the subtrees (i.e. part of the programs) that should be named as from the remainder of the program. Thereafter the genetic evolution's objective

is to get optimal solutions for the problem and their decompositions into the automatically defined functions together [20]. For a given environment, it can be easily determined the fitness function for the genetic programming, either by using a specific application evaluation method, or by utilizing a form of a symbolic regression strategy. No matter which method will be implemented, the final evolved program is scheduled to be implemented, for all cases, to find out the objective which it might be carried out. Then the outputs of such objective functions will be used to compared with a given desired results, determining the output errors which are used to evaluated the tree-based chromosome's fitness. Finally, a termination condition which is utilized to stop the implementation of the algorithm should be utilized, e.g., the condition that the maximum number of evolving iterations has been reached is widely used [21].

2.1.3 Evolutionary Programming (EP)

EP is very similar to genetic algorithms, and it comes from the simulation of phenotypic evolution to utilize its most promising feature, i.e., the adaptive behavior. It originated from the study result of L. Fogel in 1962. The basic evolution programming makes use of four components, i.e., the initialization, the mutation, the evaluation, and the selection. The most distinct feature of evolutionary programming is its mutation strategies, including seven different kinds of strategies as mentioned above. A number of variants of evolutionary programming are also proposed in the literature. Some of them can be listed as in the above introduced eight variants. It also has been applied on a number of fields, such as Bayesian network training, controller design, robotics, games, image processing, power systems, scheduling and routing, model selection, and design.

Genetic speaking, EP is a population-based random optimization method, which emphasizes the placement of the behavioral relationship between the parent individuals and the offspring individuals. Compared with genetic algorithm which generally trying to emulate some knowledge-domain specifically obtained genetic operators, evolutionary programming does not make use of such operator which can be observed from natural mechanisms.

Instead, it is quite similar to the evolution strategies even though both the evolution

strategies and the evolutionary programming are designed and developed by different scientists independently. The common features between evolution strategies and the evolutionary programming are that (1) they are useful optimization methodologies which can be applied on a lot of problems; and (2) they are population-based evolutionary methods, indicating that the analytical discovery for them are almost impossible, which is the distinct features of the traditional optimization mathematical methods. Like many other evolutionary computational algorithms, evolutionary programming also has an inherent problem that it is usually trapped into local optimal solutions. Finding the global optimal solutions is not a trivial task for it. As an alternate, real-valued function combinatorial optimization problems where the solution search space or the fitness search surface is very rugged [22].

To run an evolutionary programming, it should be firstly pointed out that an underlying assumption should be utilized. The assumption is that the solution landscape should be featured in terms of a number of variables, just like genetic algorithms do. As a result, optimal solutions, either local or global one, exist in the fitness space, and can be expressed in terms of these variable. As an example, for the famous traveling salesman problem, the objective is to find the shortest routing path for the salesman when the cities and their locations are given. Thus each solution for the traveling salesman problem should be an Hamilton routing, or path. The length of such an Hamilton routing can be regarded as the solutions' fitness. Consequently, the fitness landscape can be characterized via a hyper-surface whose Hamilton routing lengths are proportional to those under one-to-one mapping in the surface.

The overall framework of Algorithm of EP is shown in Fig. 2.2. Each procedure can be introduced in details as following [23].

1. An initialization procedure is carried out the initialize a population of candidate solutions for the problem. Generally, the solutions in the population will be generated based on a randomly distributed model. If some prior problem-related knowledge can be acquired, some other distribution models based initialization can also be designed and utilized. Most importantly, the number of the initial population is quite influencing the computational complexity of the algorithm. However, it is very hard

to determined a trade-off between the population size and the computational complexity. Usually, it is set according the problem.

2. The initialized population will be manipulated under a number of iterations (i.e., generations). Each solution in the population will be selected and mutated according to some mutation operators. If there are search boundaries for these mutated variables, once the mutated one located out of such scope, it will be re-initialized into the feasible scope.

3. Selection and mutation operators strongly influence the search performance of evolutionary programming. In the literature, many variants of selection and mutation operators have been proposed. After mutation, the offspring will be evaluated by using a fitness evaluation procedure. It should be noted that there is no crossover operator in the evolutionary programming.

4. The algorithm continues until a termination condition will be fulfilled.

2.2 Ant Colony Optimization (ACO)

2.2.1 Introduction

The Ant Colony Optimization (ACO) is inspired by the real ant behaviour (Fig. 2.3) in finding the shortest path between the nest and the food [24]. This is achieved by a substance called pheromone that makes ants can keep the trace of the others. In its search the ant uses heuristic information which itself knows where the smell of the food comes from and the other ants pheromone information of the path toward the food.

In fact the algorithm uses a set of artificial ants as individuals to solve a problem by exchanging information via pheromone deposited on graph edges. The overall framework of algorithm of ant colony optimization is shown in Fig. 2.4.

2.2.2 State transition

The conventional TSP is one of the most ordinary and popular NP-hard problems. A large number of algorithms mentioned in references resolving TSP and enhacing the solution's quality are divided two categories: the exact and the approximation algorithms [25–27]. But this kind of TSP that is static and ideal does not accord with the realistic applications. Nowadays, DTSP, the variant of TSP, possesses the dynamic environments such as replacing the cities with time [28, 29], changing the cost of the cities' arcs [30] and so on. Addressing the DTSP is not to obtain the global optimal solution completely anymore, but to track the new ones with the changing environments. Therefore, the efficiency and effect of the algorithms are the key factors of solving DTSP.

In this paper, we utilize the DTSP comprised by random traffic factors. This model introduces the cost of the edge between cities i and j. The cost is $D_{ij} \times T_{ij}$, where D_{ij} is the travelled distance between cities i and j, and T_{ij} is the traffic factor indicating the traffic jam between cities i and j. After every f iterations, T_{ij} changes randomly according to a probability value m whose changing range is limited in $[T_L, T_H]$, where T_L and T_H denote the lower and upper bounds of traffic factor respectively. The value of T_{ij} determines the degree of the traffic jam. The larger the T_{ij} is, the severer traffic jam the corresponding edge indicates. Furthermore, $T_{ij} = 1$ represents no traffic jam. In this way, dynamic environments are generated randomly by the frequency f and the magnitude of the change m.

2.2.3 Solving the TSP using ACO

ACO is a typical swarm intelligence and it has been used in a wide areas of applications. In this section, a typical and initial application of the ant search will be illustrated. The ant optimization search algorithm has good potential for problem solving NP-Hard problems and recently has attracted a lot of attentions. The Ant Colony System (ACS) is the best method of the earliest ones on solving the TSP problem. It is claimed that the ACS outperforms other nature-inspired algorithms such as simulated annealing and evolutionary computation. Moreover, they compared with a hybridization method which combined the

ant search algorithm with the famous local search method 3-opt. As a result, a version of the ACS improved with a local search procedure is proposed with the advantages of some of the best performing algorithms for symmetric and asymmetric TSPs.

In recent years, many variants of ACO have been proposed for solve TSP, aiming to improve its search performance [31]. In [31], the authors used two kinds of pheromone to guide the search. The two kinds of pheromone are combined using a new evaluation during the process of the ant tacking the routing of the TSP for the next visiting city. It is realized as by changing the sizes of ants' population during the implementation of the algorithm. All variants of ACOs are investigated by using a number of TSP benchmark instances.

Another promising research was carried out by M. Dorigo in [32], where three kinds of novel models including ant-cycle, ant-quantity and ant-density models were proposed. The difference among three models is the updating method of the pheromone, either ant cycled based, or ant quantity based, or ant density based. The three models are also derived from the research [31]. The important three kinds of variants of ant models can be introduced in details as shown in the following.

1. In the ant cycle based ACO model, the pheromone updating procedure is implemented after all the ants have finished their tracking. In other words, these pheromone are updated at the same time which is quite different in the other two models. On the contrary, ant will lay out its pheromone once it visit a city in the routing, and do not wait for any other ants in the ant quantity based and ant density based models.

2. Moreover, M. Dorigo claimed that the solution convergence is much faster by using the ant density based model by comparing it with the other two models, especially in the early stage of the iteration. As a result, it is strongly suggested that the ant density based model should be applied at the earlier search stage of the algorithm.

3. Due to distinct characteristics of ant cycle based model, it is recommended by M. Dorigo that the utilization of the ant cycle based model should be used in the rest of the search stage of the algorithm. It is because the ant cycle based model has utilized the global information of the pheromone.

4. Last but not least, a local mutation method which is almost the same as those proposed in evolutionary computational algorithms is also utilized in improved ant colony optimization models. The local mutation method is used to avoid the search algorithm not to be trapped into the local solution optima.

2.3 Artificial Immune System (AIS)

This part covers a term, AIS and its whole expression is Artificial Immune Systems. This system is a good system used as metaphor for solving engineering problems. As for AIS, it just comes into being, but it still draws a lot of experts. There are experts like Jerne [33], [34] and Bersini and Varela [35]. Why so many people are interested in this system. The reason is to get some useful technology and knowledge and thus people can solve many problems. It is quite common to find out gross simplifications of the way the immune system works. However, this is not a particular question. In order to update the system, some of the experts try to turn to nature rather than computer or other manmade machines. As a result, this system will not become a hard question for people to answer.

Attempts have been made to apply AIS to many different problems. For example, there are some people who are trying their best to make the system applicable [36]. Here the authors attempted to apply simple immune metaphors to the domain of fault diagnosis by using a sensor network. Afterward, people focus on the immune system and try to solve make use it to solve some problems like the safety of computer and the check of some unknown virus [37]. More people began to discover earlier work by theoretical immunologists such as work by Perelson [34] and work by Bersini and Varela [35] who attempted to create models of the immune system that could provided inspiration for biologically motivated computing. The range of application of AIS is growing, but in Computer Science terms, the discipline is still young.

2.3.1 Introduction

There is another system IS, whose whole expression is immune system. As for this system, it is very strange and it draws a lot of people's interest because it is as complicated as our brain. With the science and technology developing by leaps and bounds, the development of immune system is very fast, too. What's more, it has drawn great interest of experts. According to our knowledge about the IS system, we come to know it is the most important in the immune system family. It has a lot of useful features, and they are useful not only in the biology field but also in the protection of the computer. There is another system that has something in common with the IS system. It is ANN whose whole expression is artificial neural networks. Owing to this network, people begin to generate the AIS. Besides, it appears very quickly and has already become a new intelligence system in the computer field.

As for AIS, people usually think it has something to do with new ideas, some novel concepts and even the parts of the computer itself. What's more, this system cannot be seen and felt, so it is very abstract. Most AIS can solve many problems that are very complicated. And these problems can be about the computer and the engineer like the recognition of the pattern and the other problems. According to this problem, we can see some clear difference between AIS and other systems. As for AIS, it mainly focuses on the computer field and other immune systems like IS and other systems, they will pay attention to the principle of the IS in order to know the behavior of the systems. As a result, people can understand the science of biology. But, this system also has some relation with other systems. For example, people usually use one method to achieve other results. For instance, people use the model of the IS to stimulate the appearance of AIS. As a result, we come to know that there is a relationship among different systems.

This part provides a comprehensive overview of the field of Artificial Immune System (AIS) which is the application of immune system metaphors to computing problems. This paper will at first pay attention to the usage of the immune system. That is because this is most relevant to this thesis, examining in-depth how other works have extracted the metaphor and applied it to their particular domain. No real attempt is made at explaining

the domain application in depth, as this would require too much additional background knowledge rather, this work focuses on the metaphors employed. A summary is then provided that highlights the strengths and weaknesses of related research, and indicates the direction for research contained within this thesis.

2.3.2 Artificial Immune System (AIS)

The field of AIS is new and rapidly expanding. In the last few years, the Computer Science and Engineering communities have started to use the immune system as a metaphor to solve a myriad of different problems. As for this part, we will try to understand the survey that is made in applying the immune system metaphor to various applications. There are a number of reasons why the artificial immune system is of interest to many researches, based on the basic principles given in the previous chapter. These can be summarized and we can see the summery from the following part:

- Recognition: The first function of the immune system is to recognize the difference of many things. And thus, it can recognize a lot of different things and make response according to the difference. What's more, the immune system can know whether the cells belong to itself or not. As a result, it can help protect its own cells and keep other cells away from the body.

- Feature Extraction: If we want to extract the characteristics of the system, we will use APC, which stands for Antigen Presenting Cells. By using this method, we can not only extract the features of the system but also try to understand its characteristics.

- Diversity: If we want to have a huge variety of elements, we will have to use the hyper mutation of the cells. And thus, there will be other forms of cells in the body. In order to make it safer, we have to make sure that the good cells can be protected and some bad cells will be kept away from the body itself. As a result, we will reach the purpose of diversity.

- Learning: The immune system can be updated very quickly. As a result, we can say that the immune system is a studying system because it can improve itself according

to that knowledge.

- Memory: As for the immune system, it can remember what it has done.
- Distributed detection: We usually can see the distribution function of the immune system. Generally speaking, one will take control of other element.
- Self-regulation: owing to the diversity of the immune systems, there are a lot of regulations about the control of the population under this system. As a result, this system can reach self-regulation.

Thus, we can see the advantages of the immune system from the above knowledge. Thus, people want to use this system in fields like computer and science. What's more, people show great interest in those applications. So, we will give some examples about these applications.

2.3.3 Use of the Immune Network Metaphor

This section gives an overview of the current techniques that use the immune network as a metaphor for different domains. The purpose of this section is to highlight different approaches taken and to show the way in which they have applied the metaphor to particular problems.

2.3.3.1 Learning with Artificial Immune System

Since we can gain some knowledge about the immune system, we will think that the immune system can be seen as a very important method of learning. What's more, we will accept that the AIS can understand the different elements and make response according to the findings. Many experts once said there were a lot of factors or elements that can be used [34, 38]. This thesis can be taken as inspiration previous work this will be examined in detail. We know that the first work that has been done was made by Cooke and Hunt [39–41]. There are three passages talking about the learning method of the system. According to the clarification, we know that the DNA can be updated or not. What's more,

in order to achieve this goal, we will obtain the learning method of the system. By using some important methods like C4.5 [42] and other methods, we can make our work done [43]. As a result, we can use some good network or just obtain a better system to fulfill our purpose. According to the author's words, we can know that the error rate of the classification. The rate is only 3%. This is quite different from that of other technologies and they have better behavior than this system. In order to have a better understanding of the system, we will create a lot of elements that can be used in the clarification. These elements may include B cells and the concept of the immune network. What's more, we can know that this assortment can be very complicated.

The core of the work was how to use the Immune Network theory [44]. As a result, people can have a better understanding of this system. As for the B-cell, we usually think that this has something to do with the body and can be seen as the DNA elements. What's more, there will be some other factors that can change the genes. Besides, we can learn a lot from the DNA sequence. What's more, we can know that they usually symbolize the components of B-cells. Generally speaking, people will choose the data of the survey without purpose to create a B-cell network. Therefore, the rest will be left to make other things. As for Antigen, it is made out of the B-cell network and it is chosen without clear purpose. As a result, the surrounding B cells are up to a certain distance away in the network. To be specific, if the bind was successful, then the B cell was cloned and mutated. They may include choosing the gene and making the mutation out of the genes. This was an attempt by the system to generate a series of virus that can help to simplify the process of the assortment. Thus, so long as people generate a new kind of cell, they will add it to the network of the family of B-cells. If the new B-cell could not be integrated, it would not be included in the B-cell family. If people cannot succeed in getting a good relationship between the new virus and the B-cell family, they will get rid of the new gene.

This work was a very encouraging first step however, and attempts were made to make it applicable to some areas that are about ration and reason [43], and thus people can easily tell whether it is fake or not. This kind of work is more important [40] because it attempts to gain more from the B-cell family. As a result, people who write this paper will try to find the relationship between the new cell and other immune system. In the system, each

case is represented by a B-cell object and the case memory was created by using a network of B cells. They will have a link with other similar cases. The memory was self-organizing was involved, as it was stated by the authors that case base reasoning does not support the concept of mutation. The idea behind creating a case base is that, given a new case or situation, it is possible to query the case base for similar past situations. With the immune based CBR system, a new case could be presented to the immune network and the best cases that matched the new cases would be retrieved. The authors argued that one benefit of using this approach is that the case memory is able to create its own organization without the need for input from the user. This can be a great benefit since it is difficult for the user to identify and create possible structures for the case base. However, the authors do point out difficulties with their approach, the principal one being that of determinism. With the immune base approach, there was no guarantee that the same result would be returned twice from my particular search. This is to do with the stochastic nature of the search mechanism in the case base; as a consequence, there was never a guarantee that the same match would be found twice.

Later, people begin to use this learning method to avoid some fake works and so on. According to the works in [41], we can get the idea that the AIS system can help with the loan and mortgage system because it can watch for the real information about the two behaviors. Owing to this kind of pursuit, people will try their best to advocate the use of B-cell objects. Thus, they will try to find out the loan application. In order to have a better result, people apply a lot of good elements to this system and try to get the desirable result. And they even have set the good example for the later research (defining how good a match had to be before a bind between B-cell and antigen could occur). Whatfs more, people can have a good relationship with the immune system because they will try their best to find out it.

This work has not been the only work addressing the problem of creating a learning system. Research made by [45] the experts aims to get some knowledge about computation from the immune system. According to their survey, they usually reach to their results based on the immune system. These experts have already got some good results by using the behavior of the system and thus they can generate something that is about diversity

and efficiency. In order to get the desirable results, they will obtain a lot of solutions. When people have to make a comparison between their research and Gas which is more traditional, they will apply many useful elements.

2.3.3.2 Controlling Robot Behaviour

People once made some attempts to use the immune system to fight with those robots and tried to narrow down the number of them. As a result, they have some form of self-organizing group behavior, work by [46] attempts to create a group of robots which behave in a self-organizing manner, to search for food with no help of those international elements. The core of their research is how to improve the link between people and the robots. In order to reach their goals, these experts use three major features of the immune system. The first one is B-cells. In this feature, they use one cell to stand for a robot and every robot has totally different characteristics. The next one is immune web. Owing to it, the robots can have a conversation with each other. The last one is the data of the B-cells. More robots there are better performance they will have. To gain the data of the number of the robots, people will use some formula which is presented in the [47]. As a result, people come to know that these robots are insulted by the near robots and they are affected by the environment, too. Every robot has his ability to calculate food and when robots compare their capacities together, they will find the difference. We can say that if a robot cannot calculate the food accurately, it will be seen as very weak. Thus, it is likely to be replaced by others. However, if a robot can calculate the data very accurately, it will be seen as a good robot. Later, it will become a very good calculating tool. According to their data, we can see that they have very good, they also point out that if they want to have better results, they will need more survey and tests.

In very similar work by [48], the immune network metaphor is applied to generating the wasp method for robots that belong to mobiles. Yet, this is quite similar with those things mentioned above. The authors do extend the concept in Lee [49] who introduce the metaphor of the T-cell to the system. Besides, they use some good formula to perfect the system [47]. However, the authors fail to get good results from the formula. Besides, people will try to include the desirable results and some other outcome with the help of the

data. Without the help of T-cell conversation, people won't have good results.

2.3.3.3 Robot Control

Moving away from the field do generation behavior for robots, work has been done on creation a simple system that can allow for a decentralized adaptive control mechanism. The authors propose this in the context of controlling the walking behavior of a 6 legged robot [50]. When we make this illustration, we usually think that the b-cell system is the leg of the robot and the immune system is the heart of the robot. Thus, the B-cell and the immune system have very close relationship. Due to the two systems, the robot can walk very well. Without one of the two elements, the robot cannot walk, let alone develop fast. As for the B-cells, we can say every B-cell is an antibody. And again it is calculated based on the equation given in [47]. As we all know that this antibody is the leg of the robot, and so the robot can move forward. This is only a proposal paper and therefore no simulation results were available.

2.3.3.4 Associative Memory

Robotics is not the only research field that has been attempting to use the immune network idea and has been applied to create an associative memory model [51]. Associative memory is used to remember patterns and enable fast and effective recall of those patterns. Rather than strictly using an immune network as a basis the author's abstract two mechanisms defined [35] that enable them to create a system that can create an associative memory. These two mechanisms are the immune system meta-dynamics and the immune recruitment mechanism. A population of points in space is defined. Thus, we can have some knowledge about the competition and try to make it better. As a result, we will have a better pattern of the robot. These patterns can then be recalled, and indeed the authors report significant improvements in both time and effectiveness of their algorithm when compared to other associative memory techniques.

2.3.3.5 Fault Diagnosis

As for the field of diagnosis, we usually say that it is the prediction and the recovery of the plant. One approach to detect abnormal sensors within a system [52], has been to use the combination of Learning Vector Quantization (LVQ) [53] and the immune network metaphor. The core idea about the system is using LVQ to have detection in the system and try to find the relationship between the two sensors. When this sensor is used, people will try to find the other sensor. Different sensors will work together. That means every B-cell will watch out for a sensor, and when it is not good, we can say it is not normal. Every sensor has a value and we can see the formula in [47]. According to the formula, we know that the characteristics of this sensor are similar to the other sensor rather than its neighbor. When we see a sensor that has a low value, we usually say that it is a fake sensor and they can generate their function in a fault situation. What's more, we have seen some cases where the results of the test are not the real purpose of the plant.

There are many kinds of diagnosis and a very normal one is active diagnosis. In this situation, people will try to find out the relationship between the current state and the ordinary sate. As for those experts, they think that this immune system can be seen as a good idea because it can generate a desirable diagnostic system. What's more, the immune system is also important to the immune network. Besides, we can see the relationship very easily [52]. These different sensors can be linked together through a web that is the immune system. Each sensor can keep its information and data although they can change very fast. Though science and technology can change by leaps and bounds, each sensor can have its own information. However, the work differs from the above is the away in which the reliability if each sensor is calculated. This will not be explored here. The key features of the immune systems are about the agents which can be seen as different from each other. They can have a parallel interaction with each other too. That means they can only rely on their knowledge and information to work. And there is not a central controller. Besides, people will use a network to create better memory of them.

2.3.3.6 Augmenting Genetic Algorithms

In order to address the issue of designing a Genetic Algorithm (GA) with improved convergence characteristics, especially in the design field, we will give some examples about the immune system based on the GA system [54]. The motivation for their work focuses on the fact that if we want to make a good design element, we will have to make some choice and never try to neglect the relationship among the designs. In order to have a better solution, people will try their best to use some antibodies like DNA to make the definition of the similarity among those complicated solutions. We can see this from work found in [47] and is simply a bit by bit match for continuous regions. As for this model, people can generate and remove some novel solutions. People can use this model to clarify a lot of questions. Thus, we can get a result that some ways are very good and specific but some ways are very abstract.

Yet, there are many experts who just point out that there are a lot of structural designs. They are very important because all of them can introduce a system of control into this design. Besides, these people can make a contextual solution to this problem. What's more, by using these solutions, people can understand that all the experts and other good people can have a better understanding of the problem.

The main difference of their approach to more traditional GA solutions is the way in which the fitness of a solution is computed. Using this approach, people can find a very suitable way to solve this problem. As a result, this solution is not only about the function of the object but also is about the design as it would be in a traditional GA, but also on how well the solution matches the best solution. Then they will choose some ways that may help them with the problem. For example, they may come to a conclusion that these ways are very useful. Whatfs more, when we use a method that is different from the traditional one, we will have to adjust to a better situation. Yet, although many experts think that this has something to do with the immune system, we cannot deny that other systems also contribute to it.

We can see the above researches that pay much attention to the problem. We can see this specific problem from the work in [55]. Besides, we can adjust to the situation where

people will use the immune network to solve many problems. And experts hold the view that these people represent the use of an immune system and can do a lot to solve the problem. As for the immune system, we usually think that it include B-cells, T-cells and some other important genes. As for the agents of the paper, we will try our best to figure out the ways to answer the questions. Later, when we want to know more about the problem, we will try to understand the problem. Besides, we can get better results if we can have a better understanding of the problem. What's more, we can say that if we want to find out the reason why the immune system has so many elements. I can tell you that these problems are everywhere. Besides, they may include B-cells, T-cells and other important genes. Besides, we can have similar researching experiences too. Also, these systems can have very different functions. And we can find those functions from work in [56–58]. At this time, the experts can use methods like calculation, clarification and other possible solutions to solve the problem. What's more, if we want to have more knowledge about the immune system, we can adapt some ways to solve this problem. There are two features about this. The first one is the capability to generate a series of solutions. The second one is about the efficient survey. When we include the mutation, we can say that this is very important and can stand for some standard situation. In order to solve the problem, people will try their best to gain more about the mutation. In doing so, we can apply a lot of good ways to solve those problems. Besides, they even argue that the functions of this system are better than that of the traditional functions. What's more, they even think that this system can maintain a lot of good features that can solve this problem. As a result, people can get better solution.

2.3.3.7 Scheduling

It is quite difficult to generate some important ways to deal with a fast changing problem. There are many authors who have written some good books like Mori [59], Chun [60], and Mori and Makotot [61]. They have generated some important ways to adapt to the metaphors of somatic hyper mutation and the immune network theory. There are also works in [59–61] that stress the importance of the building work. What's more, we can understand the sequence of the problem and the functions of the object. As a result, we

can say that these problems can be solved easily if we can understand each other quickly. Thus, we can be optimistic about the work. In order to control the immune network, we have to stimulate and perfect the system. Owing to this, we can say that this can help to take control of the production of many antibodies. If we do so, we will ignore the effect of T-cell. However, those experts still hold the view that their ways are efficient and thus can solve the problem very quickly. As a result, we can say these scheduling is very optimistic. In order to have a better result, we have to obtain more work to adapt to the changing surroundings.

2.3.3.8 The Use of Other Immune System Metaphors

The use of other ISM is attracting growing interest with Computer Scientists. This section will briefly outline current areas of research where immune system metaphors, other than the immune network idea, are being employed.

The first area to examine is the safety of computers. Thus, we may consider it as the safety system of computers [62].Although there are a lot of different aspects between the living organism and the computer system, we can say they still have many things in common. And we can do a lot to improve the safety. To establish a good immune system of computer [63–66], we have to make a lot of researches. What's more, we have to use some modern technology to improve the safety. We can see a lot from the current work in the example of [67]. At the same time, we can draw people's interest to help with the further development. As a result, we can gain much from the immune system, too.

We can use so many ways to generate the safety system of the computers. What's more, we can use some basic methods [68, 69] to update the development of the computer system. Besides, we can improve the statement of the computers or other elements. And all the software can help to improve the specific feature of the element. Thus, we can have a database of the researches and thus we can make our behaviors checked too.

In order to build up a pattern of normal behavior for a particular information bank of the software, we have to watch out for the systems all the time. But, sometimes we may forget the old information bank and as a result, we may leave much information out of date. According to the explanation of the author, we can say this system is not very dear because

we can use it in our daily life. And thus, it will not be wasted. Besides, we can say they are very important because this information may be very good too.

There is another method that can replace this system. It is the instruction of the lately information. Besides, we can say they can be used as the networks of computers rather than some personal computers. This technology can be used in the field of network service and other kinds of services. We can see the works in [70, 71]. From the two systems, we can easily see the possibilities of the possible buildings. This work was advanced in Kim and Bentley [72]. The two people have made great contribution to the development of the intrusion detection. In this paper, they can improve the quality of the buildings. From the very beginning, we can see it in [63]. What's more, we have made some efforts for the development of the system, too. But, there are still many negative things that we have to deal with. According to [73, 74], we can see the harmful aspects of the system. So, we have to handle them. As for the [63], we can see three aspects of the system. The first one is definition of the system. The second one is the detection of the system. And the last one is the observation of the system. But, in this thesis, we will stress the computer viruses of the system.

In order to apply that information, we have to make great contribution to the problem. We can see the work in [63]. According to the information, we can use the method to improve the safety of the computers. What we can learn is that we can tell the ordinary computers from those abnormal computers. We can see separate ways in [75–77]. But, they have something in common. That is they all use the metaphor to improve the immune system. As a result, people can adapt to other situations like COM and .EXE files. If an unknown virus is detected, a sample is captured. From this sample, we can see some useful information about the virus. What's more, we will know about the processing system of it. This is analogous to how the innate immune system works, as the first line of defense. From the processing system, we can see that much of the virus is stimulated or discouraged one way or another. But, they all happen in the controlled surrounding. As for this situation, we can say that they are more about the development of the virus system. Thus, we can see a lot from the virus. The signature extraction mechanism is based on immune system metaphors. For example, we usually think that when we duplicate a product, we will produce a lot of

possible code of signatures. Thus, we can have many samples to choose and we are likely to choose the best sample. This is achieved by generating large numbers of samples that are chosen at random. As a result, we can see the potential virus easily. But, sometimes we can see that some virus has been found. If we want to get rid of the bad virus, we have to make efforts to identify them at first. Then, we can try to remove them. There are many ways to make this definition and we can try our best to watch out for those bad virus. However, if there are virus that is difficult to identify, we have to obtain other ways to improve the situation.

We can see many books in [78–80]. Those books mainly talk about how to choose the most useful information. As a result, we can also see a large number of experiments, too. Thus, if we want to apply the algorithm to the Mackey Glass series [81] and was used to create suitable test date and some simulated dates. The authors claim that the algorithm is successful in detecting any shift from the normal pattern effectively without any prior knowledge of the anomaly.

Work in [37] looks at applying immune system metaphors to extended genetic algorithms for search optimization problems. Here, the authors propose an extension to a standard GA by the inclusion of immune system metaphors of B cells, using a combination of memory cells and suppression cells. This augmentation to the algorithm creates a memory of best case for searching, allowing for the reinforcement of good solutions within the search space and to use those good solutions to further exploration.

Work proposed in [37, 82] suggest a framework for creating a production line control system based on immune system metaphors. In order to realize this, the authors use the immunological metaphors of: (i) B cells, which present detector agents a monitor for changes in the environment (for example a malfunction); (ii) T cells, these are implemented as a mediator agent to help activate the B-cell (or detector agent to create a specific response to a change (iii) an inhibitor agent, which corresponds to antigens that can kill specific B cells (this helps also to control the production of possible changes), and (iv) a restoration agent which corresponds to help T cells and assists in detecting malfunctions etc. The authors suggest that their framework offers decision making in real time and tolerance to changing environments.

Krishna Kumar and Neidhoefer [82, 83] create an adaptive intelligent control mechanism that is applied to the problem of aircraft control. Rather than using explicit immune system metaphors, the authors employ immune system notation to explain their system, and a GA to produce the adaptation required to changing environments.

McCoy and Devarajan [84] applies an artificial immune system to extract features from aerial images. Here the authors propose that, rather than generating a single optimum detector via a GA, generating a random set of detectors and applying those detectors to an image. If any detectors fail to classify correctly, then they are deleted otherwise the detectors remain on the image. The authors applied their algorithm to extracting a road from an image and found the results to be encouraging whereas [62] used immune system metaphors for the recognizing of spectra for chemical analysis.

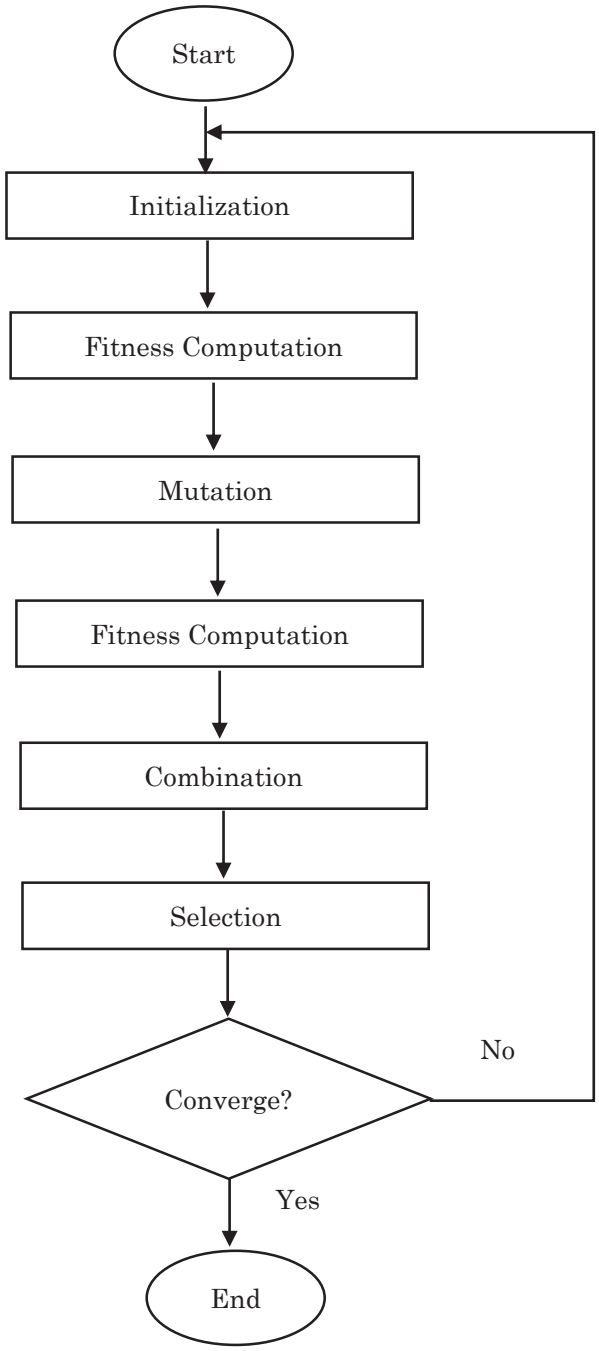

Figure 2.2: The overall framework of Algorithm of EP.

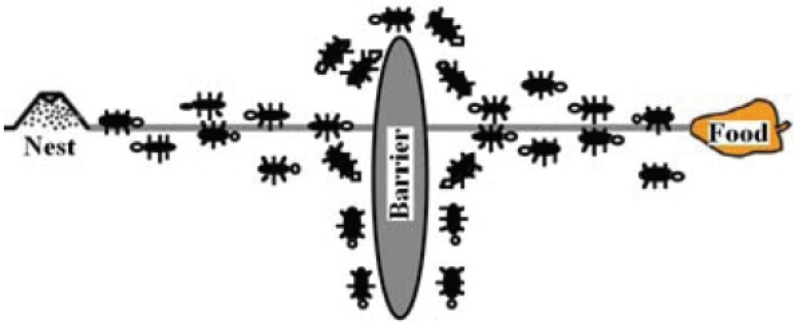

Figure 2.3: Real ant behaviour of obtaining the shortest path between the nest and the food.

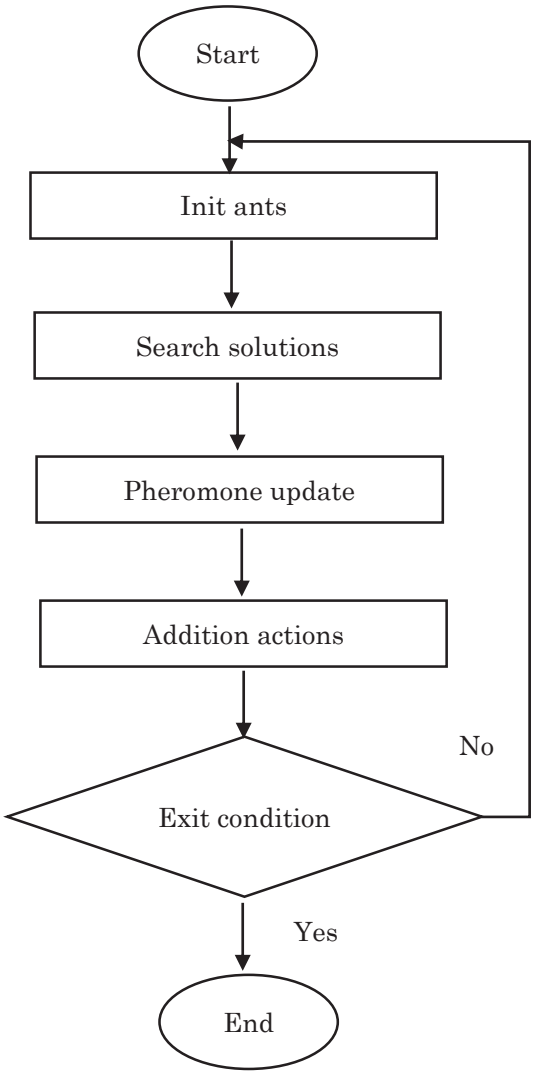

Figure 2.4: The overall framework of Algorithm of ACO.

Chapter 3

Immune Algorithm combined with Estimation of Distribution

3.1 Introduction

The traveling salesman problem (TSP) is a classical problem of combinatorial optimization which has been extensively studies over the past few decades [85]. The objective of the generalized TSP is to find a minimum total cost Hamiltonian cycle. Formally, consider a graph $G = \{N, E\}$, where N is a set of nodes representing cities and E is a set of arcs connecting these nodes. The distance between the city i and the city j is denoted by $d(i, j)$. Therefore, a generalized TSP consists of finding a permutation π of cities in the graph G that minimizes the total tour length $D(\pi)$:

$$D(\pi) = \sum_{i=1}^{N-1} d(\pi_i, \pi_{i+1}) + d(\pi_N, \pi_1) \tag{3.1}$$

There are several practical uses of TSP, such as vehicle routing, drilling, logistic, transportation, planning, gene sequencing problems, etc. Over the years, TSP has been the testing ground for numerous techniques inspired from a variety of sources.

Traditional mathematical exact techniques had been employed to solve TSPs, such as linear programming [86], dynamic programming [87], and branch and cut algorithm [88]. Although exact algorithms can find optimal solutions for TSP, their performance are limited due to the scale of instances, i.e., only small scale instances of TSP can be well solved by these exact algorithms.

Nowadays, attempts are being made to solve the optimization problems by using metaheuristics, which are mostly nature inspired, like genetic algorithms (GA), simulated annealing (SA), artificial neural networks (ANN), artificial immune systems (AIS), particle swarm optimization (PSO), ant colony optimization (ACO), artificial bee colony (ABC), estimation of distribution algorithms (EDA), tabu search (TS), greedy randomized adaptive search procedure (GRASP), and so on. All these metaheuristics can be applied to a wide variety of problems. Some algorithms give a better solution for some particular problems than others. However, there is no specific algorithm to achieve the best solution for all optimization problems [89]. Thus, it is further required to have an effective hybridization of metaheuristics with other types of algorithms for solving complex problems, which can lead to more efficient behavior and greater flexibility in application. The growing interest in this scenario of hybrid metaheuristics and some of the typical application to the variants of TSP can be summarized in Table 3.1.

It is worth emphasizing that Table 3.1 is aiming to give some insights into the research community of the hybrid metaheuristics, rather than to give a comprehensive survey or a taxonomy (readers may refer to [132] for an example) of it. It is observed from Table 3.1 that continuous research is been carried out to hybridize different algorithms to achieve high performances for TSPs. Experimental results in these research works consistently demonstrate that pure algorithms almost always inferior to hybridizations.

To contribute more in this research area, this work investigates the effect of hybridizing AIS with EDA as rare investigation is reported for such hybridization. A novel immune algorithm (IA) based on the clonal selection principle is utilized to performance the search, while EDA is used to construct a probability model for further sampling new solutions. Two different probability updating models, i.e., the univariate marginal distribution algorithm (UMDA) and the population-based incremental learning (PBIL) are introduced based on the permutation representation of TSP. In addition, a refinement local heuristic is also designed for those sampled new solutions to guarantee their feasibilities. The hybridization of IA with EDA, namely, IA-EDA, is validated based on a number of TSP instances with size up to 100 thousand cities. The simulation results indicate that IA-EDA is superior to the original IA, and some other state-of-art algorithms.

The remaining sections of the paper is organized as follows. Section 2 gives a brief description of immune algorithms, while Section 3 describes the related background of EDA and its algorithmic framework. The proposed IA-EDA is described in detail in Section 4. In Section 5, extensive simulations results are presented. Finally, we give some general remarks and further works to conclude this paper.

3.2 Immune Algorithm

Artificial immune system (AIS) [133] is one of the nature-inspired algorithms. It is a population based problem-solving technique and mimics the mechanisms of the biological immune response, which depicts the procedures of responses when a biological immune system is exposed to an antigen. The most commonly used mechanisms of the biological immune system are clonal selection proliferation, negative selection, immune network, danger theory, and dendritic cell model [134, 135]. Among them, the clonal selection algorithm is a special class of AIS, and it is inspired by the clonal selection principle. Recently, clonal selection algorithm is very popular in the AIS community and brings about larger number of applications, such as optimization, learning, clustering and so on [136]. For solving optimization problems, clonal selection algorithm utilizes a collective learning process of a population of antibodies, and undergoes a cycle process of clonal proliferation, maturation and antibody selection. Based on a fitness function, the clonal proliferation favors better antibodies to reproduce more often than those are relatively worse. During the period of maturation, descendants of antibodies are generated using randomized learning operators. Thereafter, fitter antibodies are selected to be reserved to enter into the next generation. The general clonal selection principle can be illustrated in Fig. 3.1.

To date, several variants of the clonal selection algorithm have been developed for solving TSP. The first one is CLONALG [137] which is inspired by the two most important features of the affinity maturation in cells. One is that the proliferation of each antibody is proportional to its affinity, while the other is that the mutation suffered of an antibody is inversely proportional to its affinity. These two features make the clonal selection algorithm distinct from other evolutionary-like algorithms, such as GA. By introducing the

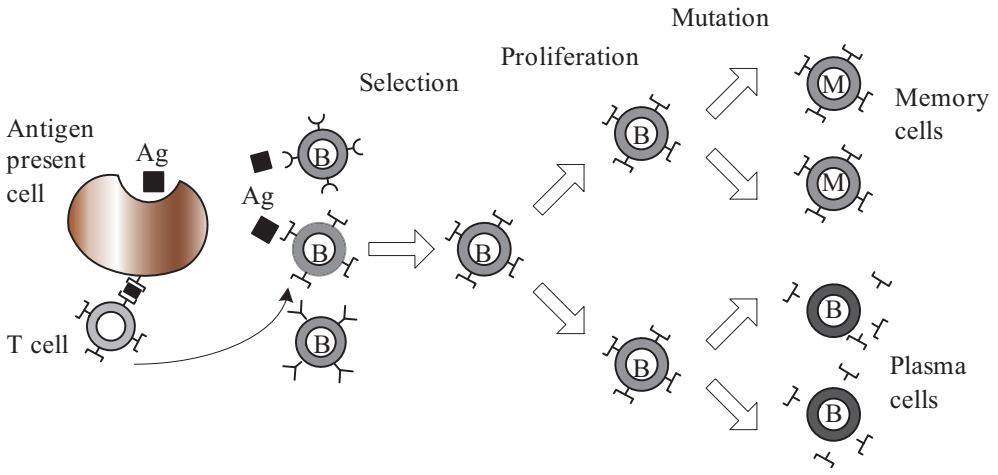

Figure 3.1: The clonal selection principle.

immune memory strategy, Liu et al. [138] proposed a MCSA for TSP, and further developed in [139]. After realizing the information exchange during different antibodies through receptor editing [140], idiotypic network mechanism [141], and the feedback interaction between B and T cells [142], the performance of the clonal selection algorithm are greatly improved. Moreover, attempts are made by incorporating other search algorithms such as ACO [120], chaotic search [122], greedy search [121], self-organizing map [119], quantum search [117, 118] into the conventional clonal selection algorithm to achieve high-performing hybrid algorithms.

In this paper, we adopt the simple receptor editing embedded clonal selection algorithm [140] as the immune algorithm (IA) to be combined with EDA. The reason to choose this variant is twofold. First, compared with CLONALG which can simultaneously solve continuous and discrete optimization problems, the receptor editing embedded clonal selection algorithm is a specialized immune algorithm for solving TSP where the receptor editing operator is proposed specially for permutation encoding based problems. Second, no sophisticated (but generally time-consuming or needing complex data structure) mutation operators (such as Lin-Kernighan [143], edge assembly crossover [144], etc.) are utilized in the hybrid algorithm, suggesting that 1) the effect of hybridization of two algorithms can be clearly observed through comparing the hybrid one with the single component algorithm respectively, and 2) potential improvement can be further achieve by incorporating such

sophisticated mutation operators if adequate computational time burden can be afforded in the actual application.

3.3 Estimation of Distribution Algorithm

Estimation of distribution algorithm (EDA) is a new area of evolutionary computation, signaling a paradigm shift in genetic and evolutionary computation research [145]. Incorporating (automated) linkage learning techniques into a graphical probabilistic model, EDA exploits a feasible probabilistic model built around superior solutions found thus far while efficiently traversing the search space [146–149]. Thus, it has a theoretical foundation in probability theory and serves as a population-based search tool. An algorithmic framework of most EDAs can be described as:

Framework of EDA
Pop = Initialize Population(); /*Initialization*/
while Stopping criteria are not satisfied **do**
Pop_{sel} = Select(Pop); /*Selection*/
$Prob$ = Estimate(Pop_{sel}); /*Estimation*/
Pop = Sample($Prob$); /*Sampling*/
end-while

At the beginning of the EDA search, a solution population Pop and a solution distribution model $Prob$ is initialized, and this is followed by a main search loop, consisting of three principal stages. The first stage is to select some best individuals from Pop, according to the fitness criteria. These individuals are used in the second stage in which the solution distribution model $Prob$ is updated or recreated. The third stage consists of sampling the updated solution distribution model to generate new solution offspring. These new solutions are evaluated and incorporated into the original population, replacing some or all of the old ones. This process is repeated until the termination criterion is met.

3.4 Hybridization of IA and EDA

It is observed from the description of IA and EDA that the two algorithms have different tendency to search the optimum solution. It is expected that the hybridization of two distinct algorithms can take advantage of the characteristics of both algorithms and thus has the ability to overcome the inherent disadvantages of each component algorithm. The overall framework of the hybrid IA-EDA is shown in Fig. 3.2, where initialization, affinity evaluation, clone operator, hyper-mutation, receptor editing, probabilistic modeling, sampling, refinement operator, and apoptosis are iterated until a pre-specified termination criterion is fulfilled.

3.4.1 Permutation Representation

Various data structure can be used to code the tour for TSP, such as the *Permutation* representation, the *Matrix* representation which is usually adopted in the ANN [150] and the *Tree* representation (involving Splay Tree [151], Two-Level Tree [152] and Segment Tree [153]) whose implementation is efficient but programming is more complex. Considering the simplicity of programming and the analogy with the gene representation in the immune system, the *Permutation* $\pi = (\pi_1, \pi_2, ..., \pi_i, ..., \pi_N)$ is utilized in this paper, where $\pi_i = k$ denotes that the city k is in the position i in the tour. Thus this permutation represents a feasible solution for TSP that the first city to be visited is the value of π_1 and the ith city to be visited is the value of π_i. The last city to be visited before going back to the city π_1 is the city π_N.

3.4.2 Modeling and Sampling of EDA

This section describes the two different types of EDA including the univariate marginal distribution algorithm (UMDA) and the population-based incremental learning (PBIL). The original versions of the EDAs are in binary representation [145] which is not suitable for the permutation representation of TSP. Thus, the probabilistic modeling and sampling approaches are altered to manipulate the permutation representation for dealing with TSP.

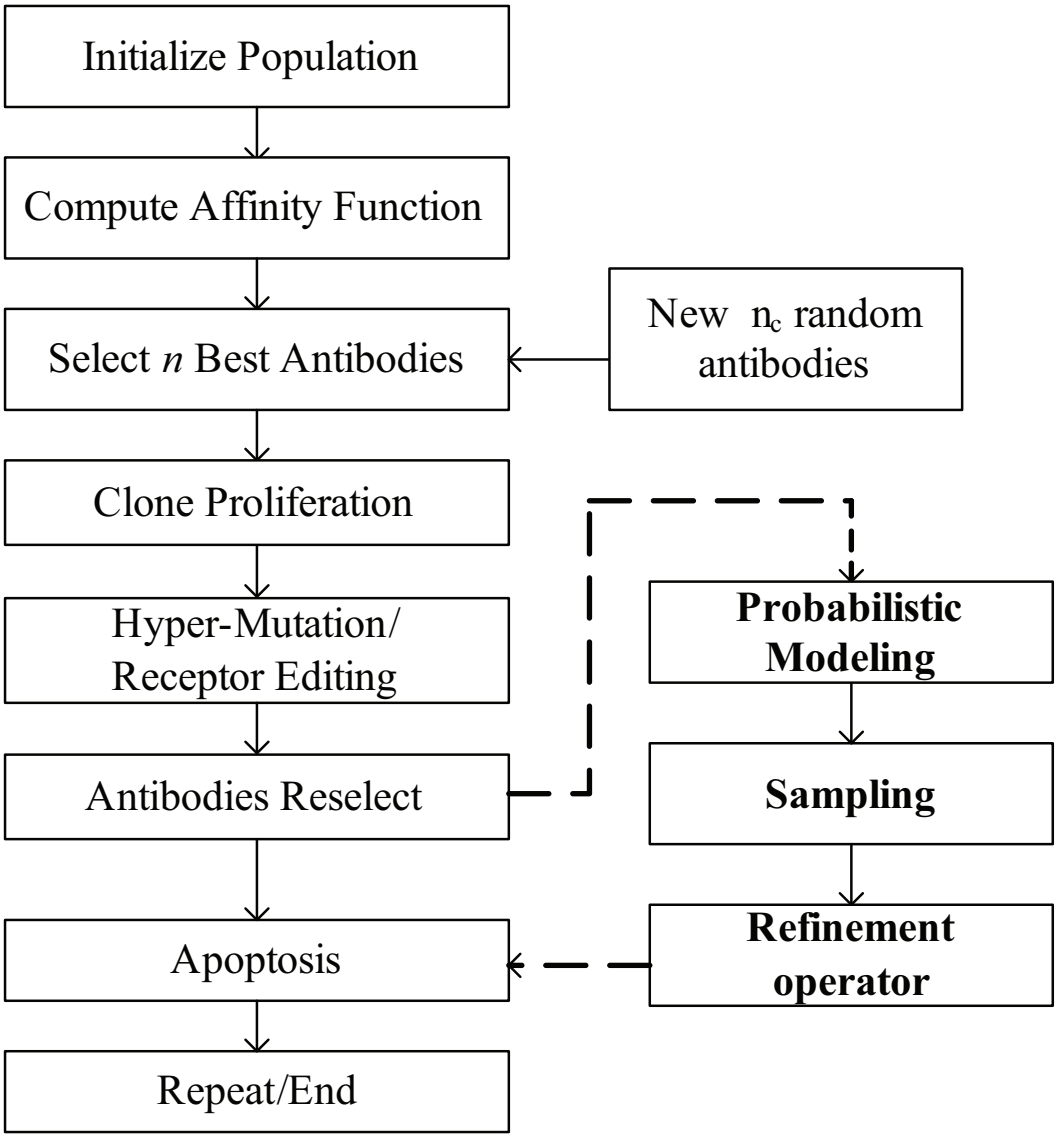

Figure 3.2: The overall framework of the proposed IA-EDA.

UMDA makes use of univariate modeling for the learning of the probability distribution of the cities in each position of the permutation without consideration of the linkage dependencies between the cities [148]. The modeling is constructed based on a $N \times N$ probability matrix $P^t(\pi_i)$ which models the probability of the cities in TSP:

$$P^t(\pi_i) = \begin{bmatrix} p^t(\pi_1 = c_1) & \dots & p^t(\pi_N = c_1) \\ p^t(\pi_1 = c_2) & \dots & p^t(\pi_N = c_2) \\ & \dots & \\ p^t(\pi_1 = c_N) & \dots & p^t(\pi_N = c_N) \end{bmatrix} \tag{3.2}$$

where $P^t(\pi_i)$ is the probability distribution of the cities at iteration t of the algorithm. $p^t(\pi_i = c_j)$ is the probability of city j to be located at the i-th position of the permutation in an antibody. c_j denotes the city j ($c_j = j$), and N is the number of cities. The modeling considers the frequency of existence of the cities in each location of the antibody.

The probability of the cities in each position of the antibody is calculated according to the following equations.

$$p^t(\pi_i = c_j) = \frac{\sum_{k=1}^{N} \Delta_k(\pi_i = c_j) + 1/N}{m + m/N} \tag{3.3}$$

$$\Delta_k(\pi_i = c_j) = \begin{cases} 1 & if \;\; \pi_i = c_j \\ 0 & otherwise \end{cases} \tag{3.4}$$

where m is the population size of antibodies. It should be noted that the term $1/N$ is added to set the upper and lower bounds to the probability of each city. This is important as the probability of 0.0 and 1.0 will make no progress in latter evolutions since a probability of 0.0 indicates that there will never be an iteration of this particular city in the position of the antibody, and this will generate infeasible solutions for TSP. Likewise, a probability of 1.0 suggests that there will always be the iteration the same city in the same position of the antibody, which will drastically decrease the diversity of the population, and thus bring down the search performance of the algorithm.

On the other hand, PBIL [146] is another version of EDA which uses the same modeling approach as UMDA. The primary difference of the PBIL with the UMDA is in terms of its probability updating rule. To be specific, in PBIL, the probability of the cities in each

position of the antibody is calculated by considering the probability of the cities in current and previous iterations of the evolution. The updating rule of PBIL can be shown as:

$$pr^t(\pi_i = c_j) = \alpha p^t(\pi_i = c_j) + (1 - \alpha)p^{t-1}(\pi_i = c_j) \tag{3.5}$$

where $\alpha \in [0, 1]$ is the learning parameter of PBIL. $pr^t(\pi_i = c_j)$ is the eventual probability of the city j at the i-th population of the antibody at the iteration t. $p^t(\pi_i = c_j)$ is obtained according to Eqs. (3.3) and (3.4). α is set to 1.0 initially because there is no prior probability distribution from any previous iteration of evolution. In this situation, PBIL is similar to UMDA. To differentiate PBIL from UMDA, α would never be set to 1.0 over course of the evolution process.

In addition, a roulette wheel selection based sampling method is used for both UMDA and PBIL. New antibodies are generated by sampling the computed probability distribution, according to the following equation.

$$\pi_j = \begin{cases} c_1 & if\ r \leq p^t(\pi_j = c_1) \\ c_2 & if\ p^t(\pi_j = c_1) \leq r \leq \sum_{i=1}^{2} p^t(\pi_j = c_i) \\ & \dots \\ c_n & if\ \sum_{i=1}^{n-1} p^t(\pi_j = c_i) \leq r \leq \sum_{i=1}^{n} p^t(\pi_j = c_i) \end{cases} \tag{3.6}$$

where π_j is a newly generated city at the j-th position of an antibody, r is a uniformly generated random number between 0 and 1.

3.4.3 Refinement operator

It is important to point out that the new antibodies generated by EDA cannot guarantee the feasibility. In other words, circumstances that some cities may not be visited at all or be visited more than once exist in the solutions sampled by EDA. Thus, a simple refinement local search operator is proposed not only to fix the feasibility of antibodies, but also to improve the quality of these antibodies in a local search manner. The pseudo code is described in Algorithm 1.

Algorithm 1– **Refinement operator**

For $i = 1 : Q$

input an antibody π_i sampled by EDA.

identify the location of the repeated cities in the antibody π_i, $L = \{L_1, ..., L_j, ..., L_R\}$.

identify the unvisited cities in π_i, $V = \{V_1, ..., V_k, ..., V_R\}$.

For $k = 1 : R$

calculate the probability for all unvisited cities V_k to be located at the position L_j according to the following equation:

$$p_{jk} = \frac{d(\pi_{L_j-1}, V_k)}{\sum_{l=1}^{R} d(\pi_{L_j-1}, V_l)} \tag{3.7}$$

insert the unvisited city V_k to the L_j^{th} position of the antibody and discard that city from V.

End-For k

End-For i

For example, given a TSP instance with $N = 7$ and an antibody $\pi = \{3, 2, 2, 4, 3, 1, 5\}$ generated by EDA. The antibody is clearly an infeasible solution for TSP. Using Algorithm 1, we got $L = \{3, 5\}$ and $V = \{6, 7\}$. If the distance between cities 2 and 6 is smaller than that between 2 and 7 (i.e. $d(2, 6) < d(2, 7)$), then it is more possible to generate a refined solution $\{3, 2, 6, 4, 7, 1, 5\}$ than the other one $\{3, 2, 7, 4, 6, 1, 5\}$, and vice versa.

3.4.4 Structure of IA-EDA

IA-EDA works as follows.

Step 1) First, m antibodies are randomly generated to create an initial population, which can be represented as $(A_1, A_2, ..., A_m)$. Initialize the probability distribution model in Eq. (2) with all element in the matrix has the same value $1/N$, revealing that the probability model has no guiding information for initial search.

Step 2) Calculate the affinity of all antibodies ($D(A_1)$, $D(A_2)$,...,$D(A_m)$)) according to

Eq. (1).

Step 3) Select the n ($< m$) fittest antibodies based on their affinities.

Step 4) Clone all antibodies with a rate proportional to its affinity. The amount of clone generated for these antibodies is given by $(n-i)\times Q/n$, where Q denotes the clone size and i represents the rank of the antibody in the population.

Step 5) Mutate all antibodies under either hyper-mutation operator or receptor editing. The hyper-mutation performs a randomly point mutation, while receptor editing carries out an inverse operator of gene fragment [140].

Step 6) Update the population using mutated clones. Replace the parent antibody if its offspring antibody has a higher affinity.

Step 7) Update the probability distribution model using UMDA described by Eqs. (3)(4) and PBIL by Eq. (5) respectively.

Step 8) Sample new antibodies based on Eq. (6).

Step 9) Perform the refinement operator introduced in Algorithm 1.

Step 10) Insert all newly sampled antibodies by EDA into the population and select the n best ones from it. The rest antibodies are removed from the population (i.e. apoptosis process).

Step 11) Replace the worst n_c (generally $n_c = \lceil 0.1 \times n \rceil$) with new randomly generated antibodies to maintain the diversity of the population.

Step 12) Repeat Step 4)-11) until the maximum number of generation G_{max} to evolve is reached.

3.5 Experiment Results and Discussions

In this section, the performance of the proposed IA-EDA is investigated by applying the proposed algorithm to solve TSP benchmark instances taken from the standard TSPLIB library [154]. There are three types of the selected instances involving EUC_2D, GEO and ATT. The meanings of them are euclidean distance, geographical distance and pseudo-euclidean distance respectively. A detailed description can be found in TSPLIB. Then we listed these instances in Table 3.4 and indicated their corresponding maximum numbers

of generation used in our implementation. The tested benchmark instances are with sizes ranging from 51 to 100000 cities. Moreover, each instance is run for 30 independent replications to make a statistical analysis.

The proposed IA-EDA was implemented in C++ language under Visual Studio 2010 and run on a PC with an Intel 1.70 GHz CPU. In addition to the maximum generation number, the other parameters of IA-EDA are fixed on the following values: the number of initial antibodies m is set to 100, the population size n is set to 50, the clonal size of proliferation Q is set to 50, the complementary intensity between hyper-mutation and receptor editing is set to 0.5, i.e., two mutation operators have the same probability to be performed, and the learning parameter α of PBIL is set to 0.9. These parameter are experimentally obtained by testing the IA-EDA on the different instances.

To investigate the effects of EDA on IA, two variants of IA-EDA and the original IA [140] are implemented on all tested instances. The IA-EDA using UMDA is denoted as IA-UMDA, while the other one using PBIL is named as IA-PBIL. The characteristic difference between IA-UMDA and IA-PBIL is the updating mechanism of the probability distribution model. IA-UMDA only utilizes the current information of antibodies generated by IA to update the probability model, while IA-PBIL further makes use of the accumulative information of the probability model. For the sake of perspicuity, two assessment criteria of performance are used to analyze the experimental results. The optimum tour length that listed in Table 3.4 are labeled as D_{opt}. The *PDM* and *PDB* which indicates the percentage deviation from the D_{opt} of the mean distance D_m and best distance D_b respectively were defined as follows:

$$PDM = \frac{D_m - D_{opt}}{D_{opt}} \times 100 \tag{3.8}$$

$$PDB = \frac{D_b - D_{opt}}{D_{opt}} \times 100 \tag{3.9}$$

Table 3.5 summarizes the experimental results for all tested TSP instances based on IA, IA-UMDA, and IA-PBIL, where the best solution obtained by the algorithm for each instance is highlighted in boldface. From Table 3.5, it is apparent that IA-UMDA and

IA-PBIL performs much better than the original IA on all test 19 instances, suggesting that the EDA technique (either UMDA or PBIL) is capable of enabling IA to find much better solutions. Furthermore, IA-PBIL can obtain the global optimal solution in 30 runs for eight (out of nineteen) instances, and performs better than its two competitors for 16 instances expect gr96, rat195, and kroA200. It indicates that, in most cases, it is better to utilize the accumulative information of the probability model to sample solutions in EDA. The averaged *PDM* and *PDB* for 19 instances is 1.43 and 0.88 respectively. Thus, it can be concluded that IA-PBIL can produce better solutions than IA and IA-UMDA within reasonable computational times.

To further analyze the search dynamic of algorithms and the obtained solutions, Fig. 3.3 depicts the comparative results by means of convergence graphs and box-and-whisker diagrams of solutions during IA, IA-UMDA and IA-PBIL over 30 replication runs for instances st70, pr136 and att532. Similar results can be also acquired for the other instances. The algorithms' behaviors on the selected three instances are quite illuminating to further elaborate the effects of EDA on the search dynamics of the algorithm. To be specific, IA-UMDA and IA-PBIL can produce significantly better solutions than IA after the first generation although all of them use randomly generated initial populations. In other words, the EDA together with the heuristic refinement local search operator can greatly improve the qualities of the initial solutions. The reason seems to be twofold: 1) since there is no information exchange procedure in IA, EDA can realize the information exchange through the probability model during different antibodies, thus being able to sample promising solutions for IA, and 2) the refinement local search operator utilizes the problem-depended knowledge (at least the distance matrix of the TSP instance) which can also facilitate the search to some extent. Generally, IA-UMDA and IA-PBIL have faster convergence speed than IA. Although the convergence graphs of IA-UMDA and IA-PBIL exhibit almost the same search dynamics, IA-PBIL can produce better solutions than IA-UMDA.

Further considerations deal with the significant differences during the behavior of IA, IA-UMDA and IA-PBIL. Table 3.6 summarizes the results of the Wilcoxon signed ranks test [33] and the average rankings of the algorithms obtained by Friedman test [155, 156] on all tested TSP instances. In Table 3.6, R^+ denotes the sum of ranks for the problems in

which the base algorithm (i.e. the one before "vs") outperformed the competitive one (i.e. the latter), and R^- the sum of ranks for the opposite. The associated asymptotic p-values are also computed to identify whether a statistical hypothesis test is significant or not. It is worth pointing out that, from the statistical point of view, the Wilcoxon signed ranks test is more sensitive and safer than the t-test, because it does not assume normal distributions and meanwhile the outliers have less effect on the Wilcoxon test than on the t-test [33]. Moreover, the Friedman test applying the post hoc method of Iman-Davenport [156] is another nonparametric statistical test which can rank the algorithms for each problem separately; the best performing algorithm should have the rank of 1, the second best rank 2, etc. The values in the last three columns record the average ranking of the three algorithms for each TSP instance. As Table 3.6 states, IA-PBIL and IA-UMDA show significant improvement over IA at a level significance $\alpha = 0.05$ for all 19 TSP instances. IA-PBIL performs significantly better than IA-UMDA on pr107, pr124, pr152, ja9847, and mona-lisa. Friedman test shows that IA performs the worst during three algorithms, and IA-PBIL acquires a total averaged rank of 1.41 which is a bit smaller than that of 1.58 obtained by IA-UMDA, indicating that IA-PBIL performs statistically better than IA-UMDA.

To make a clear illustration of the solutions obtained by IA-EDA (either IA-UMDA or IA-PBIL), Fig. 3.7 shows the best routes found by IA-EDAs (three obtained by IA-UMDA and the others obtained by IA-PBIL) and their corresponding total tour length.

Finally, an intensive comparison with existing six state-of-art hybrid metaheuristics in the literature is carried out to further validate the performance of the proposed IA-EDAs. Table 3.8 summarizes the computational results of the two variants of IA-EDA and its six competitors in terms of the average tour length (Avg.), the standard deviation of the obtained tour lengths and the *PDM* defined in Eq. (8) which is used to reflect the percentage relative error. As can be seen from Table 3.8, the proposed IA-EDAs perform better than its competitors on st70, rd100 and bier127, and can generate very competitive solutions for the rest instances, suggesting that the proposed hybrid metaheuristic search algorithm by combining IA with EDA is a promising alternative method for solving TSP. In addition, it is worth pointing out that the complexity of algorithmic construction of IA-EDAs is significantly simpler than the two algorithms in [107, 129] since there are more than two

partial components in the latter hybrid algorithms, although two algorithms in [107, 129] can produce a bit better solutions than IA-EDAs.

3.6 Conclusions

This study proposed combining EDA and IA to create a new hybrid that can efficiently solve TSPs. In this method, EDA is used to realize the information exchange during different solutions generated by IA through the probabilistic model. EDA enables IA to quickly converge on promising search areas. To refine the solutions sampled by EDA, a heuristic local search operator is also proposed to repair the infeasible solutions, and further facilitate the search by making use of the problem-dependent knowledge of TSP. The effects of two kinds of EDAs including the univariate marginal distribution algorithm and the population-based incremental learning are investigated by taking into consideration average tour length, best tour length, convergence performance and the distribution of solutions on 19 different TSP instances with size up to 100,000 cities. A non-parametric statistical test based on the Wilcoxon signed ranks test and the Friedman test consistently demonstrates the effects of EDA on IA. Intensive comparative results of other six hybrid metaheuristics reported recently in the literature verify the superiority of IA-EDA, and show that IA-EDA can produce better or competitive solutions than other hybrid algorithms within reasonable computational times.

Finally, it is important to remark that the proposed algorithm assessed in this work did not make use of "strong" operators, in the sense that no efficient route improvement heuristics available to date, such as Lin-Kerninghan [143], edge assembly crossover [144] or quantum interference crossover [117] were incorporated to perform the search. In spite of that, the proposed algorithm was capable of finding solutions that are on average less than one percentage worse than the best known results for all tested instances, thus showing the strength of the hybrid strategy.

In the future we plan to apply the proposed IA-EDA in solving more optimization problems such as the vehicle routing problem and the sequential ordering problem.

Table 3.1: Summary of different hybrid metaheuristics and their applications to variants of TSP.

Researchers	Hybrid metaheuristic	Application
Deng et al. [90]	GA + ACO + PSO	generalized TSP
Wang et al. [91]	GA + ACO	generalized TSP
Mavrovouniotis and Yang [92]	GA + ACO	dynamic TSP
Baraglia et al. [93]	compact GA + local search (Lin-Kernighan)	generalized TSP
Tsai et al. [94]	parallel GA + ACO	generalized TSP
Nguyen et al. [95]	steady-state GA + local search (Lin-Kernighan)	generalized TSP
Xing et al. [96]	GA + local determinate/stochastic search + global optimization	generalized TSP
Lin et al. [97]	GA + Newton-Raphson search	generalized TSP
Dong et al. [98]	cooperative GA + ACO	generalized TSP
Abbattista et al. [99]	GA + ACO	generalized TSP
Lope and Coelho [100]	GA + PSO	blind TSP
Chen and Flann [101]	GA + SA	generalized TSP
Creput and Koukam [102]	GA + ANN (self-organization map)	generalized TSP
Jin et al. [103]	GA + ANN	generalized TSP
Glover et al. [104]	GA + TS	generalized TSP
Shim et al. [105]	GA + EDA	multiobjective TSP
Marinakis et al. [106]	GA + GRASP	generalized TSP
Chen and Chien [107]	GA + SA + ACO + PSO	generalized TSP
Martin and Otto [108]	SA + local search (Lin-Kernighan)	generalized TSP
Malek et al. [109]	SA + TS	generalized TSP
Geng et al. [110]	SA + greedy search	generalized TSP
Pepper et al. [111]	SA + Monte-Carlo method (demon algorithm)	generalized TSP
Shim et al. [112]	SA + EDA + hill climbing + evolutionary gradient search	multiobjective multiple TSP
Marinakis and Marinaki [113]	GRASP + PSO	probabilistic TSP

Table 3.2: Summary of different hybrid metaheuristics and their applications to variants of TSP, continued.

Marinakis and Marinaki [114]	GRASP + ABC + neighborhood search	generalized TSP
Hernández-Pérez [115]	GRASP + variable neighborhood search	one-commodity TSP
Marinakis and Marinaki [116]	GRASP + ABC	probabilistic TSP
Dai et al. [117]	AIS + quantum search	generalized TSP
Dai et al. [118]	AIS + bi-direction quantum search	generalized TSP
Masutti and de Castro [119]	AIS + ANN	generalized TSP
Gao et al. [120]	AIS + ACO	generalized TSP
Guo et al. [121]	AIS + greedy search	generalized TSP
Gao et al. [122]	AIS + chaotic search	generalized TSP
Wei et al. [123]	ACO + chaotic search	generalized TSP
Duan and Yu [124]	ACO + Memetic algorithm	generalized TSP
Gunduz et al. [125]	ACO + ABC	generalized TSP
Saenphon et al. [126]	ACO + opposite gradient search	generalized TSP
Shuang et al. [127]	ACO + PSO	generalized TSP
Elloumi et al. [128]	ACO + PSO	generalized TSP
Mostafa et al. [129]	ACO + PSO + 3-opt	generalized TSP
Saadatmand-Tarzjan et al. [130]	ANN (Hopfield) + ANN (self-organization map)	generalized TSP
Muhammed and Tareq [131]	ANN (elastic net) + iterative improvement	generalized TSP

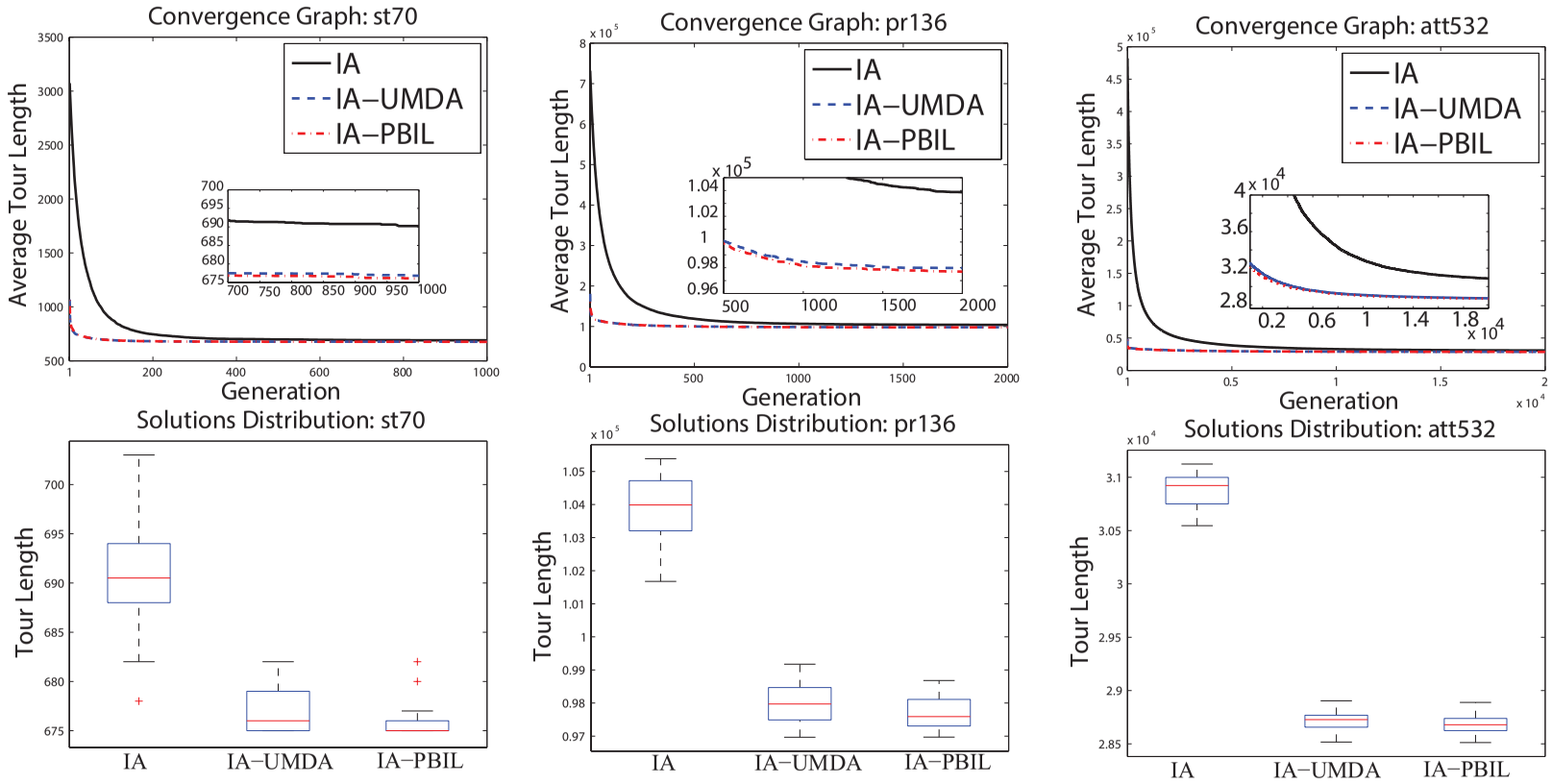

Table 3.3: Comparison of the average convergence performance and the distribution of the obtained solutions using box-and-whisker plots during IA, IA-UMDA and IA-PBIL over 30 replication runs for instances st70, pr136 and att532 respectively.

Table 3.4: Problem instances used in our simulation and the maximum number of generation for each instance.

Instance	Size	Type	Optimum	G_{max}
eil51	51	EUC_2D	426	1000
st70	70	EUC_2D	675	1000
eil76	76	EUC_2D	538	1000
gr96	96	GEO	55209	1000
rd100	100	EUC_2D	7910	1000
eil101	101	EUC_2D	629	1000
lin105	105	EUC_2D	14379	1000
pr107	107	EUC_2D	44303	1000
pr124	124	EUC_2D	59030	1000
bier127	127	EUC_2D	118282	2000
pr136	136	EUC_2D	96772	2000
pr152	152	EUC_2D	73682	2000
rat195	195	EUC_2D	2323	2000
kroA200	200	EUC_2D	29368	5000
lin318	318	EUC_2D	42029	10000
pcb442	442	EUC_2D	50778	10000
att532	532	ATT	27686	20000
ja9847	9874	EUC_2D	491924	20000
mona-lisa	100000	EUC_2D	5757191	20000

Table 3.5: Experimental results of the original IA, and the two variants of IA-EDA, i.e., IA-UMDA and IA-PBIL.

Problem	D_{opt}	IA			IA-UMDA			IA-PBIL		
		PDM	PDB	T(s)	PDM	PDB	T(s)	PDM	PDB	T(s)
eil51	426	2.59	0.94	4.0	0.13	0	7.1	0.12	**0**	7.5
st70	675	2.26	0.44	6.3	0.27	0	11.7	0.16	**0**	12.2
eil76	538	5.25	3.34	6.5	0.17	0	13.8	0.12	**0**	14.2
gr96	55209	4.95	2.73	6.7	0.81	**0.42**	17.6	0.76	0.54	18.4
rd100	7910	6.14	2.39	6.7	0.18	0	18.6	0.17	**0**	19.5
eil101	629	7.01	4.29	7.4	1.53	0.48	19.0	1.53	**0**	19.7
lin105	14379	4.98	1.45	7.3	0	0	19.3	0	**0**	19.9
pr107	44303	3.98	0.95	7.8	0.53	0.10	19.4	0.33	**0**	19.6
pr124	59030	6.50	2.82	7.9	0.80	0.10	19.7	0.59	**0.10**	20.1
bier127	118282	5.15	2.60	16.0	0.51	0.25	39.2	0.51	**0.04**	41.1
pr136	96772	7.33	5.07	16.9	1.25	0.20	41.4	0.95	**0.20**	43.5
pr152	73682	4.29	2.07	17.6	0.23	0	43.6	0.09	**0**	47.5
rat195	2323	14.1	11.2	14.8	1.45	**0.47**	48.9	1.41	0.77	51.2
kroA200	29368	7.90	6.32	38.2	1.28	**0.49**	177.3	1.31	0.67	184.1
lin318	42029	10.0	7.91	102.9	2.74	2.06	601.2	2.48	**1.55**	645.7
pcb442	50778	16.3	13.2	133.5	3.91	2.99	789.5	3.69	**2.71**	801.6
att532	27686	11.5	10.3	306.2	3.73	3.00	1576.1	3.56	**2.98**	1656.4
ja9847	491924	14.9	12.5	403.4	6.75	4.57	2045.2	4.77	**4.25**	2457.1
mona-lisa	5757191	21.5	19.6	501.5	9.71	6.53	3345.6	4.58	**2.86**	3865.3
Average:		8.24	5.80	84.8	1.89	1.14	466.0	1.43	0.88	523.4

Table 3.6: Results of the Wilcoxon's signed ranks test at a level of significance $\alpha = 0.05$ and the average rankings of the algorithms obtained by Friedman test on all tested TSP instances.

Problem	IA-PBIL vs IA				IA-PBIL vs IA-UMDA				IA-UMDA vs IA				IA	IA-UMDA	IA-PBIL
	R^+	R^-	p-value	Sig.	R^+	R^-	p-value	Sig.	R^+	R^-	p-value	Sig.			
eil51	465.0	0.0	0	YES	248.5	216.5	0.375	NO	465.0	0.0	0	YES	3	1.5333	1.4667
st70	465.0	0.0	0	YES	285.0	180.0	0.216	NO	465.0	0.0	0	YES	3	1.5667	1.4333
eil76	465.0	0.0	0	YES	244.5	220.5	0.755	NO	465.0	0.0	0	YES	3	1.5	1.5
gr96	465.0	0.0	2.0E-6	YES	262.5	172.5	0.323	NO	465.0	0.0	2.0E-6	YES	3	1.5833	1.4167
rd100	465.0	0.0	2.0E-6	YES	195.0	240.0	1	NO	465.0	0.0	2.0E-6	YES	3	1.45	1.55
eil101	465.0	0.0	0	YES	203.0	232.0	1	NO	465.0	0.0	0	YES	3	1.4167	1.5833
lin105	465.0	0.0	2.0E-6	YES	232.5	232.5	0.992	NO	465.0	0.0	2.0E-6	YES	3	1.5	1.5
pr107	465.0	0.0	2.0E-6	YES	359.5	75.5	0.002	YES	465.0	0.0	2.0E-6	YES	3	1.6833	1.3167
pr124	465.0	0.0	2.0E-6	YES	337.5	127.5	0.024	YES	465.0	0.0	2.0E-6	YES	3	1.7	1.3
bier127	465.0	0.0	2.0E-6	YES	216.5	248.5	1	NO	465.0	0.0	2.0E-6	YES	3	1.4	1.6
pr136	465.0	0.0	2.0E-6	YES	317.0	148.0	0.080	NO	465.0	0.0	2.0E-6	YES	3	1.5667	1.4333
pr152	465.0	0.0	2.0E-6	YES	361.5	73.5	0.000	YES	465.0	0.0	2.0E-6	YES	3	1.75	1.25
rat195	465.0	0.0	1.0E-6	YES	248.0	187.0	0.476	NO	465.0	0.0	2.0E-6	YES	3	1.55	1.45
kroA200	465.0	0.0	2.0E-6	YES	243.0	222.0	0.821	NO	465.0	0.0	1.0E-6	YES	3	1.5667	1.4333
lin318	465.0	0.0	2.0E-6	YES	326.5	138.5	0.051	NO	465.0	0.0	2.0E-6	YES	3	1.6667	1.3333
pcb442	465.0	0.0	2.0E-6	YES	299.5	135.5	0.074	NO	465.0	0.0	2.0E-6	YES	3	1.5833	1.4167
att532	465.0	0.0	2.0E-6	YES	316.0	149.0	0.084	NO	465.0	0.0	2.0E-6	YES	3	1.6	1.4
ja9847	465.0	0.0	2.0E-6	YES	360.0	75.0	0.002	YES	465.0	0.0	2.0E-6	YES	3	1.7	1.3
mona-lisa	465.0	0.0	2.0E-6	YES	380.5	54.5	0.000	YES	465.0	0.0	2.0E-6	YES	3	1.8	1.2

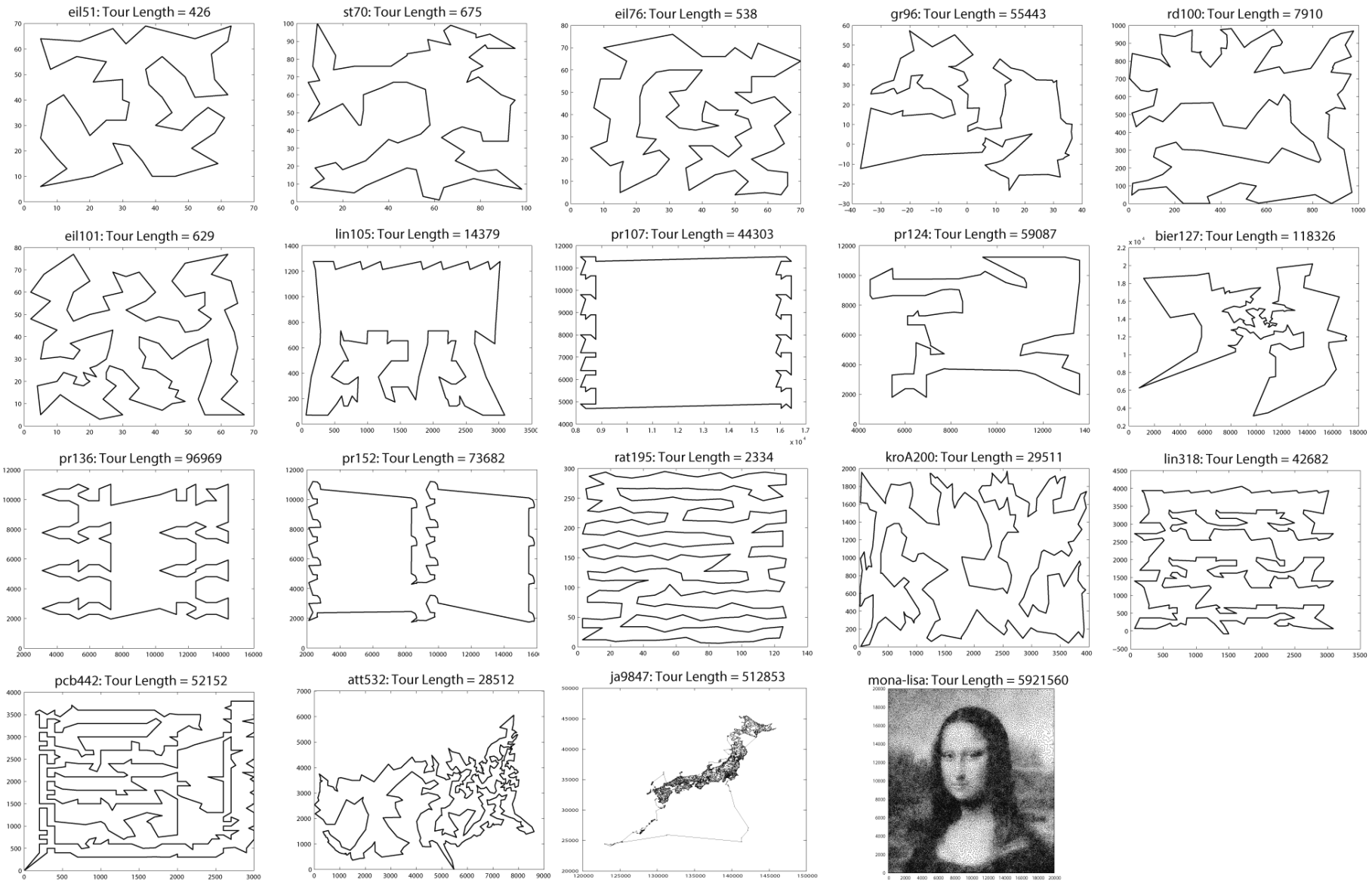

Table 3.7: Best routes found by IA-EDA for tested 19 TSP instances.

Table 3.8: Computational results of the two variants of the proposed IA-EDA and its competitor algorithms in the literature. D_{opt} is the best known solution; Avg. is the average tour length; SD is the standard deviation; *PDM* defined in Eq. (8) shows the percentage relative error.

Method	Problem	eil51	st70	eil76	rd100	eil101	lin105	bier127	kroA200	lin318
	D_{opt}	426	675	538	7910	629	14379	118282	29368	42029
GA+ACO (2004) [94]	Avg.	430.68	-	555.70	-	-	-	-	-	-
	SD	-	-	-	-	-	-	-	-	-
	PDM	1.10	-	3.29	-	-	-	-	-	-
AIS+ANN (2009) [119]	Avg.	437.47	-	556.33	8199.77	648.63	14400.17	120886.33	30190.27	43696.87
	SD	4.2	-	5.30	80.77	3.85	44.03	1158.79	273.38	410.06
	PDM	2.69	-	3.41	3.66	3.12	0.15	2.20	2.80	3.97
GA+SA+ACO+PSO (2011) [107]	Avg.	427.27	-	540.20	7987.57	635.23	14406.37	119421.83	29738.73	**43002.90**
	SD	0.45	-	2.94	62.06	3.59	37.28	580.83	356.07	307.51
	PDM	0.30	-	0.41	0.98	0.99	0.19	0.96	1.26	2.32
CGA+ACO (2012) [98]	Avg.	-	-	542.00	-	-	-	-	29946.00	-
	SD	-	-	-	-	-	-	-	-	-
	PDM	-	-	0.74	-	-	-	-	1.97	-
ACO+ABC (2015) [125]	Avg.	443.39	700.58	557.98	-	683.39	-	-	-	-
	SD	5.25	7.51	4.10	-	6.56	-	-	-	-
	PDM	4.08	3.79	3.71	-	8.65	-	-	-	-
ACO+PSO+3-opt (2015) [129]	Avg.	**426.45**	678.20	**538.30**	-	**632.70**	14379.15	-	**29646.05**	-
	SD	0.61	1.47	0.47	-	2.12	0.48	-	114.71	-
	PDM	0.11	0.47	0.06	-	0.59	0.00	-	0.95	-
AIS + UMDA (proposed method)	Avg.	426.57	676.83	538.90	7924.47	638.60	14379.00	**118886.63**	29744.30	43182.13
	SD	0.68	2.35	1.47	21.07	3.33	0.00	176.59	113.68	154.07
	PDM	0.13	0.27	0.17	0.18	1.53	0.00	0.51	1.28	2.74
AIS + PBIL (proposed method)	Avg.	426.53	**676.07**	538.67	**7923.90**	638.63	**14379.00**	118888.73	29753.07	43072.27
	SD	0.68	1.91	1.15	22.97	4.05	0.00	264.95	125.22	192.72
	PDM	0.12	0.16	0.12	0.17	1.53	0.00	0.51	1.31	2.48

Chapter 4

A Near-Optimal Graph Planarization Algorithm using Probability Model based Particle Swarm Optimization

4.1 Introduction

Graph planarization problem (GPP) arises from practical applications of circuit board layout, facility layout, automatic graph drawing, VLSI circuit design [157]. It has significant theoretical importance related with networks design and analysis, computational geometry, and other topological problems. Generally, GPP needs to carry out two tasks including the maximum planar subgraph acquisition and the plane embedding. For a given graph G, the objective of the graph planarization problem is composed by two sub-problems: one is to find a plane which can embed the edges into it, and the other is to make the cardinality of the planar sub-graph as large as possible.

GPP has been demonstrated to belong to NP-hard problems [158] and several algorithms have been proposed to solve it with low computational complexity. A triple-valued gravitational search algorithm [159] was presented with the novelty of finding multiple optimal solutions for GPP simultaneously. In this study, we propose a novel particle swarm optimization (PSO) to solve the GPP. PSO is originally developed by Kennedy and Eberhart [160] for optimization. The technique was inspired by bird flocking and animal social behaviors. In PSO, the particles operate like a swarm that flies through the hyperdimensional space to search for possible optimal solutions. The behavior of the particles

is influenced by their tendency to learn from their past personal experience and from the success of their peers to adjust the flying speed and direction. Recent works [161–163] showed the superiority of PSO when solving various difficult problems. Surprisingly, PSO has not been applied on solving GPP. The potential applicability of PSO on GPP is first verified in this work, and thereafter a powerful probability model based PSO named PMPSO is proposed to further improve the performance of the algorithm. Experimental results based on 19 benchmark GPP instances with size up to 150 vertices and 1064 edges demonstrated the effectiveness of the proposed algorithm. The rest of the paper is organized as follows: Section II presents a brief description of the proposed PMPSO. Experimental results and discussions are given in Section III. Finally, some general remarks are presented to conclude the paper.

4.2 Probability Model based PSO

As an alternative to genetic algorithms, the PSO technique has ever since turned out to be a competitor in the field of numerical optimization [163]. A PSO consists of a population refining its knowledge of the given search space. Each individual (i.e. particle) in PSO has its own position which represents the solution for the optimization problem at hand, and its velocity which is the resultant attraction results from its personal previous best particle (pbest) and the global best particle in the whole population (gbest). During each iteration each particle accelerated in the direction of pbest as well as gbest, indicating that if a particle discovers a promising new solution, all the other particles are tended to move closer to it and thus explore the search space more thoroughly in the process. The position and velocity of the *i*th dimension is defined as

$$V_i = \omega V_i + c_1 r_1 (gbest_i - X_i) + c_2 r_2 (pbest_i - X_i) \tag{4.1}$$

where ω is the inertia weight, and c_1 and c_2 are acceleration coefficients. r_1 and r_2 are randomly generated values between 0 and 1. The new position of a particle is calculated

using

$$X_i = X_i + V_i \tag{4.2}$$

To solve GPP, the single-row routing representation that is originally proposed in Takefuji and Lee's algorithm [164] are adopted in this research. According to this The existence of a crossing between two upper edges (v_i, v_j) and (v_p, v_q) (or two lower edges) is determined by the following conditions as shown in Fig. 4.1(d):

$$\text{if } v_i < v_p < v_j < v_q \text{ or } v_p < v_i < v_q < v_j \tag{4.3}$$

Consequently, under this representation, each of the edge has three possible states according to whether, in accordance with a determined vertex sequence the edge is a lower edge, is not considered or is an upper edge.

Based on the single-row routing representation method, the position of each particle X^j $(j = 1, 2, ..., N)$ in PSO can be expressed as $X^j = (x_1^j, x_2^j, ..., x_n^j)$, where N denotes the population size and $n = |E|$ is the number of edges in the graph. As can be seen from Fig. 4.1(c), the edge in a planar graph is either an upper or a lower edge. Here, we define that x_i^j=1 means an edge to be drawn as an upper one, and x_i^j=0 to be a lower one. The velocity V_i will determine a probability threshold. If V_i is higher, the particle is more likely to choose 1, and lower values favor the 0 choice. The sigmoid function which is a straightforward function in neural networks can accomplish this as:

$$s(V_i) = \frac{1}{1 + exp(-V_i)} \tag{4.4}$$

The sigmoid function squashes its input into the requisite range and has properties that make it agreeable to be used as a probability threshold. Then the original position update rule in Eq. (2) is thus tuned to as in the following.

$$X_i = \begin{cases} 1 & if \;\; rand() < s(V_i) \\ 0 & otherwise \end{cases} \tag{4.5}$$

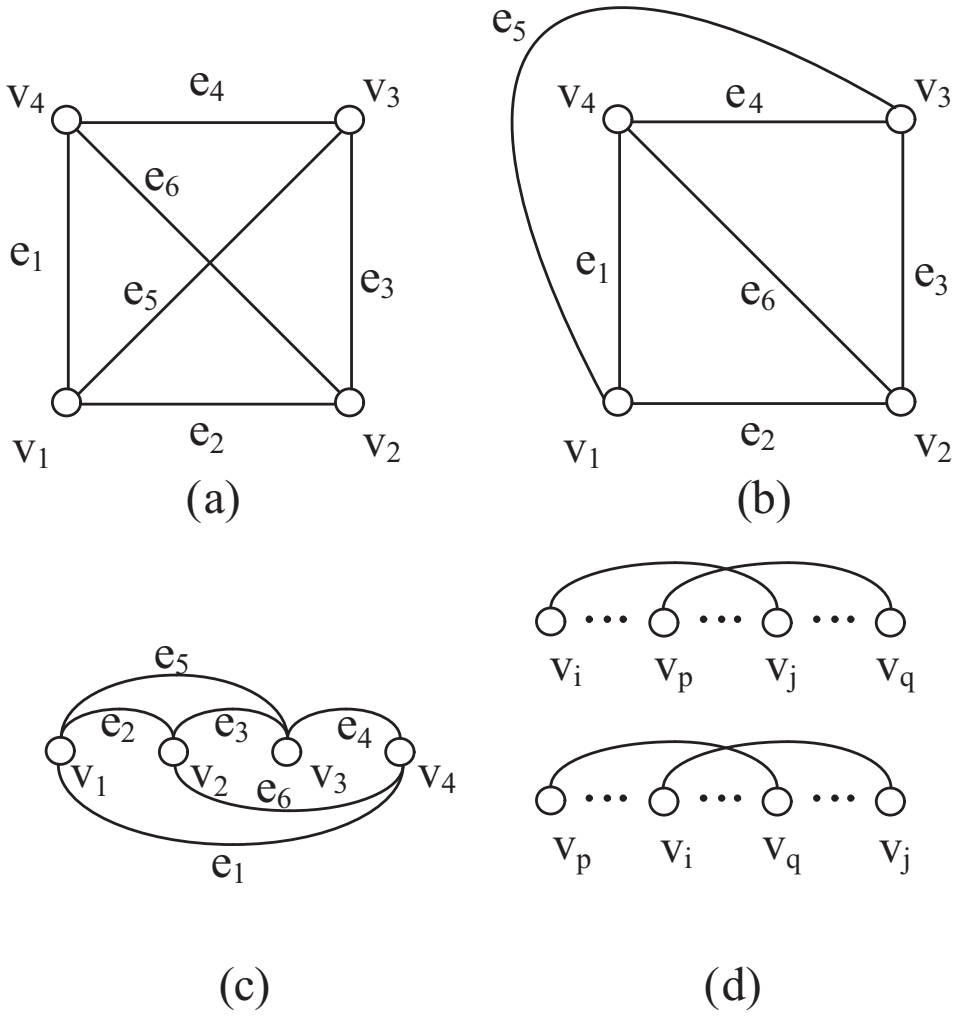

Figure 4.1: Single-row routing representation used in GPP: (a) A graph with four vertices and six edges, (b) A planar graph, (c) A possible planar graph, (d) Violation conditions.

It should be pointed out that the position of a particle obtained by Eq. (5) might not result in a feasible solution for GPP. It is important to eliminate the crossing edges which satisfies the conditions shown in Eq. (3). By realizing this, all solutions generated by PSO can be guaranteed to be feasible. Furthermore, the vertex sequence is generated based on a Hamiltonian cycle generation method [165].

Preliminary experimental results as shown in Section III shows that PSO can solve GPP with limited effectiveness. To further improve the problem-solving capacity of PSO, we incorporated a probability model into it, and therein put forward a probability mod-

el based particle swarm optimization (PMPSO). The inspiration of the probability model arises from the estimation of distribution algorithm [145] which is a new area of evolutionary computation, signaling a paradigm shift in genetic and evolutionary computation research. Incorporating linkage learning techniques into a graphical probabilistic model, a feasible probabilistic model is built around superior solutions found thus far while efficiently traversing the search space. Therefore, it has a theoretical foundation in probability theory and serves as a population-based search tool.

In this study, we used the univariate marginal distribution to construct the probability model. The probability vector which characterizes the distribution of promising solutions in the search space is defined as $P = (p_1, p_2, ..., p_k, ..., p_n)$ where p_k $(k = 1, 2, ..., n)$ the probability that the value of the kth position of a promising solution is 1. Then the current solution obtained by PSO is mutated based on the probability vector P. It can be expected that the mutated offspring falls in or close to a promising area in the search space. Such mutation perturbs the current solution X and guides PSO to search in binary solution space in the following way.

Probability Model based Mutation

Mutate the current solution X based on probability model P

For $i = 1$ to n **do**

If rand()$< \alpha$

if rand()$< p_k$, set $X_i(t+1) = 1$, otherwise set $X_i(t+1) = 0$;

Otherwise $X_i(t+1) = X_i(t)$

End-for

where rand() produces a random number distributed uniformly in the interval of [0, 1]. In the mutation procedure, a bit is sampled from the probability vector P randomly or directly copied from the current solution $X(t)$, which is controlled or balanced by the parameter α. The larger the α, the more elements of the new starting point are sampled from the vector P.

As there are a lot of information of elements are sampled based on the probability matrix P, it can be expected that they fall in or close to the promising area. As a result, the random sampling mechanism can also provide diversity for the search afterward. Initially,

the probability vector P is set as $P = (0.5, ..., 0.5, ..., 0.5)$. The probability vector can be learned and updated at each iteration of PSO for modeling the distribution of promising solutions using

$$p_k(t+1) = (1-\lambda)p_k(t) + \lambda X_i(t) \tag{4.6}$$

where the parameter λ is a learning rate.

The PMPSO works as follows.

Input: a given non-planar graph $G = (V, E)$ with $|V| = m$, $|E| = n$

01: initialize user-specified parameters, and randomly generate N positions for the particles in PSO

02: **do:**

03: **For** each particle $i = 1$ to N

04: generate a vertices sequence π_i for particle i using Hamiltonian cycle generation method

05: move and update particles using Eqs. (1)(4)(5)

06: refine the solution by eliminating the edges satisfying Eq. (3)

07: evaluate the number of edges in the resultant planar sub-graph

08: update the probability vector using Eq. (6)

09: carry out the probability model based mutation procedures

10: **End-for**

11: **While:** the maximum iteration number achieves T_{max}

Output: an optimal particle and the corresponding planar sub-graph $G' = (V, E')$

4.3 Experimental Results

In order to evaluate the performance of PMPSO when applied it on the GPP, a set of nineteen benchmark problems described in literature [165] was used in the experiments. The problem instances are summarized in Table 4.1, where the information data we record for each instance are the name of the input graph, the number of vertices ($|V|$), the number of edges ($|E|$), the Euler upper bound ($3|V| - 6$) on the number of edges in a maximum planar

Table 4.1: Problem instances used in the experiments.

Graph	No. Vertices	No. Edges	Upper bound	Best-known pre. (Ref.)
G1	10	22	20*	20 ([159, 164–166])
G2	45	85	82*	82 ([165–167])
G3	10	24	24*	24 ([165–167])
G4	10	25	24*	24 ([165–167])
G5	10	26	24*	24 ([165–167])
G6	10	27	24*	24 ([165–167])
G7	10	34	24*	24 ([165–167])
G8	25	69	69*	69 ([166, 167])
G9	25	70	69*	69 ([166, 167])
G10	25	71	69*	69 ([166, 167])
G11	25	72	69*	69 ([167])
G12	25	90	69*	67 ([165, 167])
G13	50	367	144	135 ([167])
G14	50	491	144	143 ([166, 167])
G15	50	582	144	144 ([167])
G16	100	451	294	196 ([167])
G17	100	742	294	236 ([167])
G18	100	922	294	246 ([165, 167])
G19	150	1064	444	311 ([166, 167])

* Actual known optimal size of the planar subgraph.

subgraph, and the number of edges in the best-known solution for the GPP. The algorithm is coded using C++ under the Visual Studio 2010 platform running on a personal PC (Intel(R) Core i5, 1.70GHz with 4GB RAM).

First analysis deals with the parameter sensitivities of the algorithm. The user-defined parameters in PMPSO are N, T_{max}, w, c_1, c_2, L, α and λ. The population size N directly influences the trade-off tween the search performance and the computational times. In order to make the subsequent comparison fair with other population-based algorithms, the reasonable same value of $N = 20$ was utilized in the experiments. The maximum iteration number T_{max} determines the termination condition of the algorithm. The bigger the value of T_{max}, the more computational times the algorithm needs to cost. In preliminary experiments, we found that both PSO and PMPSO had been convergent to global or local optimal solutions for all instances within 1000 iterations. Thus, $T_{max} = 1000$ was sufficient to evaluate the performance of proposed algorithms in all experiments. The inertia weight

Table 4.2: Parameter sensitivity analysis for w, c_1 and c_2 on $G9$.

w	$c_1 = 1.0$	$c_1 = 1.5$	$c_1 = 2.0$	$c_1 = 2.5$	$c_1 = 3.0$
	$c_1 = 3.0$	$c_1 = 2.5$	$c_1 = 2.0$	$c_1 = 1.5$	$c_1 = 1.0$
0.1	66	67	67	67	67
0.3	67	67	67	67	69
0.5	67	67	67	68	68
0.7	67	68	69	67	67
0.9	67	67	69	69	68

Table 4.3: Parameter sensitivity analysis for α and λ on $G9$.

α	$\lambda = 0.1$	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 0.7$	$\lambda = 0.9$
0.1	69	69	68	67	67
0.3	69	69	67	67	67
0.5	69	69	67	68	67
0.7	68	68	67	67	67
0.9	67	67	69	69	67

w together with acceleration coefficients c_1 and c_2 are PSO related parameters. The constant inertia weight w set from 0.1 to 0.9 with increment of 0.2. The different w is tested with a set of c_1 and c_2 which fulfills the condition of $c_1 + c_2 = 4$ as suggested in [170]. The simulation results are summarized in Table 4.2. The results in this table suggested that $w = 0.9$, $c_1 = 2.0$ and $c_2 = 2.0$ is an acceptable choice for these parameters. Similar observation can be obtained based on other benchmark instances. By incorporating the parameter L into the Hamiltonian cycle generation method, more diversity of the vertex sequence can be obtained, thus enabling the algorithm to search more different areas in the search space. Following the suggestion in [159, 165], $L = 3$ is verified to be effective on the performance of the algorithm. In addition, the probability model related parameters α and λ are analyzed. α and λ are defined to take values within the following discrete range $\{0.1, 0.3, 0.5, 0.7, 0.9\}$ respectively. Experimental results in Table 4.3 show that the parameter combinations $\alpha \in \{0.1, 0.2, 0.3\}$ and $\lambda \in \{0.1\}$ can obtain good results.

Further considerations address the performance of PSO and PMPSO. Table 4.4 summarized the comparative simulation results during all tested algorithms based on nineteen

benchmark graph with size up to 150 vertices and 1064 edges. The results are the number of edges in the obtained planar subgraph from the best solution based on 30 independent runs. From this table, it is clear that PSO can obtain acceptable solutions for GPP, while PMPSO can perform much better than PSO and is able to find competitive solutions when compared with other state-of-art algorithms. The typical solutions found by PMPSO on $G2$, $G7$ and $G13$ are illustrated in Fig. 4.5.

4.4 Conclusions

In this research, a novel probability model based particle swarm optimization is proposed to solve the graph planarization problems. The particle swarm optimization is for the first time used to handle the binary solutions based on the singlerow routing representation where the value 1 represents an edge can be drawn as an upper edge in the planar subgraph, while 0 denotes an lower edge. To further improve the search performance of PSO and alleviate its inherent local minimum trapping problem, a probability vector based on the univariate marginal distribution is incorporated. By doing so, the solutions generated by PSO are manipulated as accumulative information to construct the probability vector, and a random sampling mutation procedure is used to guide the search to promising areas. Experimental results based on nineteen GPP benchmark instances show that the proposed PMPSO can find competitive solutions for GPP when compared it with other algorithms. In the future, we plan to apply the PMPSO to other combinatorial optimization problems, such as maximum independent set problems, graph coloring problems, maximum diversity problems, and so on.

Table 4.4: Simulation results during seven algorithms.

Graph	Ref. [164]	Ref. [168]	Ref. [165]	Ref. [167]	Ref. [169]	Ref. [166]	Ref. [159]	PSO	PMPSO
G1	20	20	20	20	20	20	20	20	20
G2	80	80	82	82	80	82	82	80	82
G3	21	22	24	24	22	24	24	22	24
G4	22	22	24	24	22	24	24	22	24
G5	22	22	24	24	22	24	24	22	24
G6	22	22	24	24	22	24	24	22	24
G7	23	23	24	24	23	24	24	23	24
G8	58	61	68	69	61	69	69	63	68
G9	59	61	69	69	61	69	69	61	69
G10	58	61	68	69	61	69	69	65	69
G11	60	61	68	69	61	68	69	67	69
G12	61	63	67	67	63	65	69	65	69
G13	70	82	129	135	84	131	137	133	133
G14	100	109	138	143	114	143	143	138	143
G15	101	115	142	144	119	142	144	126	144
G16	92	100	183	196	101	192	205	187	203
G17	116	126	215	236	127	225	241	210	241
G18	115	135	234	246	138	237	257	234	259
G19	127	138	291	311	145	311	328	311	328

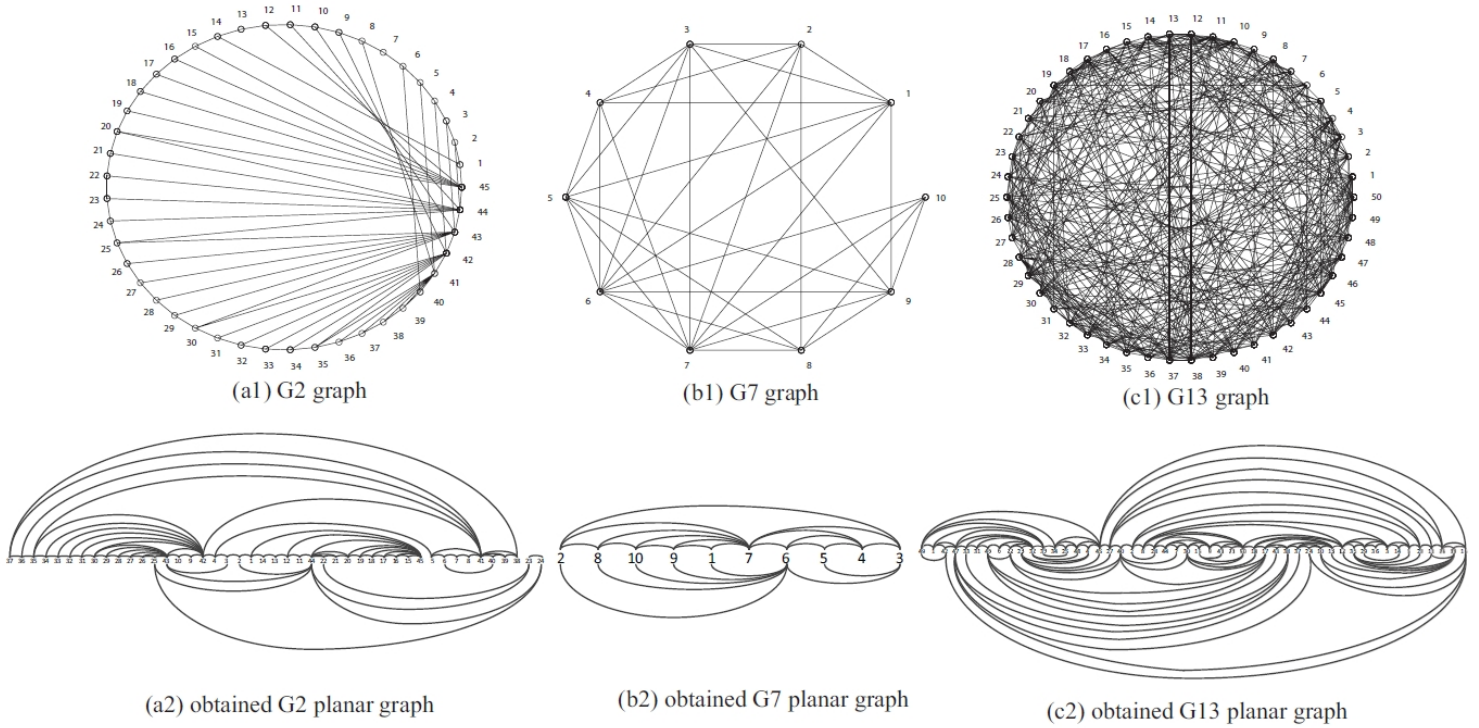

Table 4.5: Typical planar subgraphs on G2, G7 and G13 obtained by PMPSO.

Chapter 5

Ant Colony Optimization with Neighborhood Search for Dynamic TSP

5.1 Introduction

Ant colony optimization (ACO) proposed by Marco Dorigo et al. was first used to handle the static TSP and performed well [171, 172]. The ACO algorithm simulates the foraging behavior of ant colony to search for a optimal route, and ants communicate with each other via the pheromone they deposit on the trail. The pheromone guides ants to find a feasible solution. The more the previous ants release the pheromone on the trail, the higher the probability of the following ants choosing this trail will be. After several iterations, ants will acquire the best route finally. Although ACO has been applied successfully to several combinatorial optimization problems [173–175], most of the realistic applications are subject to dynamic optimization problems. It means the objective function, the constraints or the problem instances can change with time and the purpose is not to find a static optimal solution but to track the new one to the dynamic problems.

The conventional ACO may not be fit for solving the dynamic problems since the pheromone of the previous route can force ants to follow when the environment changes. This will cause inefficient response to environmental change. Therefore, several strategies were proposed to enhance the ability to adapt to the changing environment [176]. The simplest assumption is to reset the pheromone on all the trails once the environment changes, which is expensive on computing time and not reasonable. In [177], local and global restar

strategy was proposed to reinitialize the pheromone to response to dynamic environments. In [30], a generating or increasing diversity approach was used to resolve dynamic TSP (DTSP). A memory-based method was introduced to store several solutions to match the previous environments [29, 178]. The P-ACO algorithm used the population-list to store the iteration-best ants for updating the pheromone on the trails [179]. The ACO algorithm with immigrants schemes was verified to be a good approach to DTSP [178], and the random immigrants ACO (RIACO) performed well in the environment changing quickly and significantly. These approaches can solve the different DTSPs according to the strategies respectively.

In this paper, the ACO with neighborhood search (NS-ACO) is proposed to handle the DTSP composed by random traffic factors. Three moving operations are introduced to adjust the solutions constructed by ants to adapt to the new environments. The experiments implement the comparison among NS-ACO, the conventional ACS and RIACO on the DTSPs of different scales. The experimental results demonstrate the superiority of our proposed algorithm which can track the optimal solution effectively and efficiently.

This paper is organized as follows. Section 2 describes the structure of DTSP. Section 3 presents our proposed NS-ACO algorithm. Section 4 shows the comparative experiments and analyzes the experimental results. Section 5 draws a conclusion.

5.2 Brief Introduction to DTSP

The conventional TSP is one of the most ordinary and popular NP-hard problems. A large number of algorithms mentioned in references resolving TSP and enhacing the solution's quality are divided two categories: the exact and the approximation algorithms [25–27]. But this kind of TSP that is static and ideal does not accord with the realistic applications. Nowadays, DTSP, the variant of TSP, possesses the dynamic environments such as replacing the cities with time [28, 29], changing the cost of the cities' arcs [30] and so on. Addressing the DTSP is not to obtain the global optimal solution completely anymore, but to track the new ones with the changing environments. Therefore, the efficiency and effect of the algorithms are the key factors of solving DTSP.

In this paper, we utilize the DTSP comprised by random traffic factors. This model introduces the cost of the edge between cities i and j. The cost is $D_{ij} \times T_{ij}$, where D_{ij} is the travelled distance between cities i and j, and T_{ij} is the traffic factor indicating the traffic jam between cities i and j. After every f iterations, T_{ij} changes randomly according to a probability value m whose changing range is limited in $[T_L, T_H]$, where T_L and T_H denote the lower and upper bounds of traffic factor respectively. The value of T_{ij} determines the degree of the traffic jam. The larger the T_{ij} is, the severer traffic jam the corresponding edge indicates. Furthermore, $T_{ij} = 1$ represents no traffic jam. In this way, dynamic environments are generated randomly by the frequency f and the magnitude of the change m.

5.3 Proposed ACO Algorithm

Our NS-ACO algorithm adopts a short-term memory proposed by Michalis Mavrovouniotis et al [178]. The short-term memory stores K iteration-best ants in the current iteration which are used to update the pheromone in the next iteration, and the corresponding previous ants and pheromone are removed. So no ant can exist more than one iteration in case the current environment changes. The solution construction of our proposed algorithm is the same as the conventional ACO algorithm's. The possibility p_{ij}^k of ant k moving from city i to city j is described as follow:

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{l \in N_i^k} \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}} & \text{if } j \in N_i^k \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

where N_i^k indicates the set of the cities which can be visited by ant k at the city i. τ_{ij} denotes the pheromone between city i and city j. $\eta_{ij} = 1/d_{ij}$ is the heuristic information, where d_{ij} is the distance between cities i and j. The parameters α and β determine the relative importance of pheromone τ and heuristic information η respectively.

After executing the solution construction, the neighborhood search is added into the ACO algorithm to optimize the solution. The neighborhood search contains three local

search processes: swap operation, insertion operation and 2-opt operation. Each operation is implemented with 1/3 probability to generate a new solution which may be better than the current solution. The swap operation indicates the positions i and j of a solution are chosen randomly and exchanged each other. The insertion operation is to select randomly position i and position j of a solution ($i \neq j$) firstly. Then if $i < j$, move position i+1 to position i, and move position i+2 to position i+1, go on until position j, that is to say, move the positions from $i+1$ to j to the left one step distance respectively. Finally position i replaces position j. Otherwise, move the positions from $i-1$ to j to the right one step distance respectively and position i replaces position j. Likely, after choosing the positions i and j of a solution randomly the 2-opt operation reverse the segment between i and j. The neighborhood search can effectively optimize the solutions ants construct. Whereafter the pheromone update is implemented by the short-term memory which stores K iteration-best ants in iteration t, which is described as follow:

$$\tau_{ij} = \tau_{ij} + \Delta\tau_{ij}^{k} \tag{5.2}$$

where $\Delta\tau_{ij}^{k} = (\tau_{max} - \tau_0)/K$, τ_{max} and τ_0 indicate the maximum and initial values of pheromone respectively and K is the size of the short-term memory. In this paper, $\tau_{max} = \tau_0 + \sum_{k=1}^{K} \Delta\tau_{ij}^{k}$. Then in the next iteration $t+1$, the short-term memory needs to remove the previous ants to make space for the new K iteration-best ants, and accordingly the pheromone deposited previously is subtracted, described as follow:

$$\tau_{ij} = \tau_{ij} - \Delta\tau_{ij}^{k} \tag{5.3}$$

It should be noted that there is no pheromone evaporation. NS-ACO algorithm is depicted in Algorithm 1, where K_s indicates the short-term memory, N is the number of cities, t is the iteration times and M denotes the number of ants.

Algorithm 1 NS-ACO Algorithm

```
01: Initialize the parameters α, β, K_s, N, M, t
02: Initialize pheromone matrix
03: while termination criteria not be satisfied do
04:   Construct dynamic environments
05:   Clear solutions and K_s
06:   Construct solutions by ants
07:   for k = 1 to M do
08:     Execute neighborhood search:
09:       r= generate random number
10:       if (r=swap operation) then
11:         Generate a swap solution
12:       end if
13:       if (r=insertion operation) then
14:         Generate an insertion solution
15:       end if
16:       if (r=2-opt operation) then
17:         Generate a 2-opt solution
18:       end if
19:     Compare and select the best solution
20:   end for
21:   Find K iteration-best solutions and add it into K_s
22:   Update K_s using Eq. (5.2) and Eq. (5.3)
23:   Compare and record the global optimal solution
24:   t=t+1
25: end while
```

5.4 Experiment and Analysis

5.4.1 Experimental settings

In order to demonstrate the performance of our proposed NS-ACO algorithm on the DTSP, we evaluate NS-ACO, the conventional ACS [171] and RIACO [178] on several DTSPs composed by benchmark TSP instances from TSPLIB. eil51 and pr76 indicate small-scale DTSP. kroB100 and pr124 denote medium-scale DTSP. kroB150 and kroB200 represent large-scale DTSP. The ACS is one of the best ACO algorithms for the static TSP since the pheromone evaporation ρ and the pheromone decay coefficient φ change the pheromone of the trails to guide ants to find the optimal solution. RIACO enhances the diversity of population due to its random immigrant scheme, and the short-term memory with the replacement rate r shows efficient performance for DTSP.

In the experiments, all the DTSPs are constructed by random traffic factors set in $[1, 5]$, i.e., $T_L = 1$ and $T_H = 5$, respectively. The changing frequency f is set to 10 and 100, which represents the environment changes quickly and slowly respectively. The changing magnitude m is set to 0.1 and 0.9, indicating the small and large degree of environmental changes respectively. Consequently, for each DTSP, 4 dynamic instances are generated to test and analyze the property of three algorithms. Furthermore, for each algorithm on each dynamic instance, the experiment is independently run 30 times and 1000 iterations each time. The experimental data is calculated in terms of the following expression [178]:

$$F^{bs} = \frac{1}{T}\sum_{t=1}^{T}(\frac{1}{N}\sum_{i=1}^{N} f_{ti}^{bs}) \tag{5.4}$$

where T is the iteration times, N is the number of running and f_{ti}^{bs} is the value of the best solution found in the the i-th running of the t-th iteration. The parameter settings of three algorithms are shown in Table 5.1. Some parameters are referred in [171, 178], where C^{nn} is the length of the tour obtained by the nearest-neighborhood heuristic and q_0 determines the probability of search.

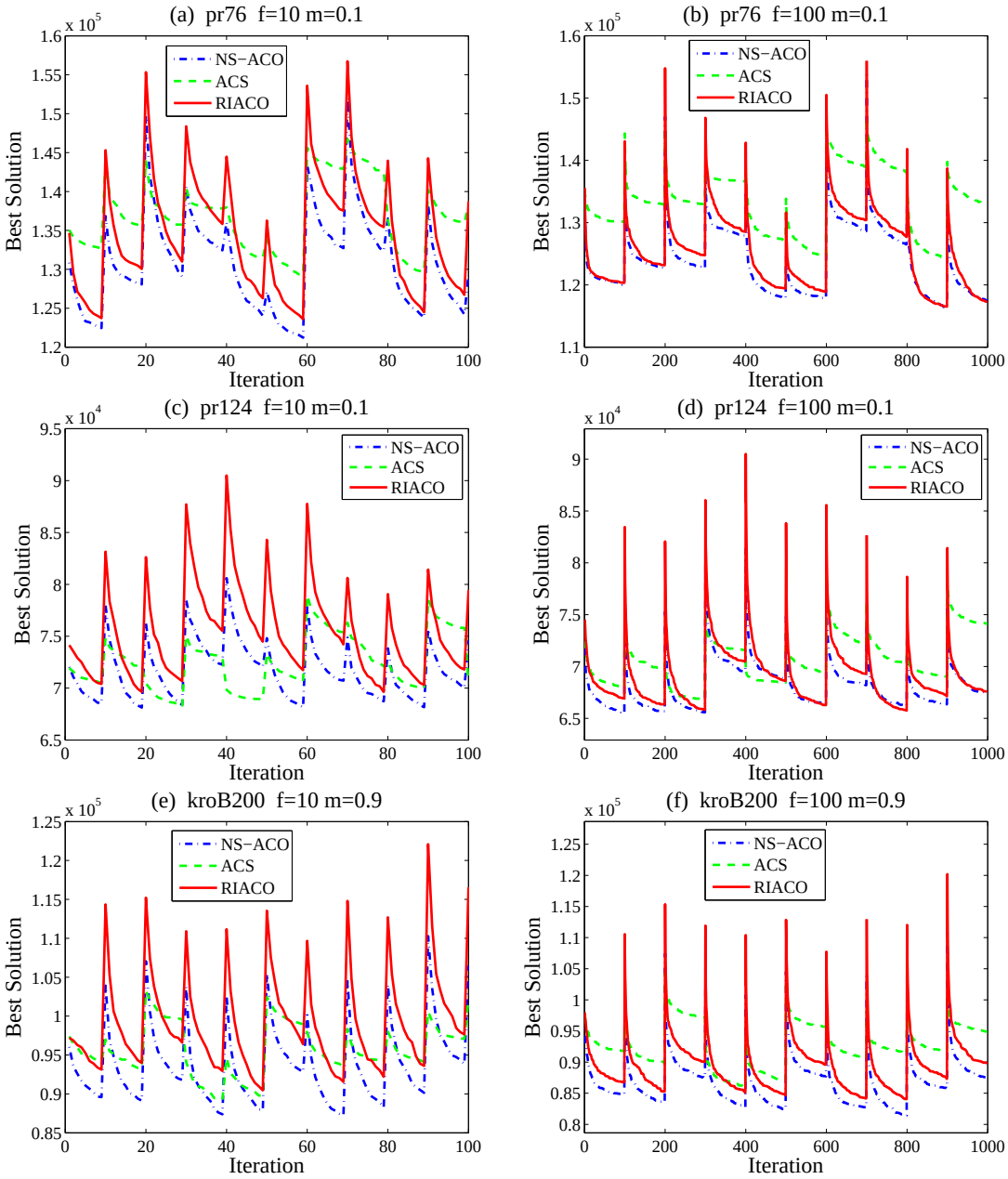

Figure 5.1: The convergence graph of the best solution obtained by NS-ACO, ACS and RIACO.

Table 5.1: Parameter settings for three algorithms in the experiment.

Algorithm	τ_0	α	β	ρ	φ	q_0	K	r	M
NS-ACO	$1/C^{nn}$	1	5	–	–	0	6	–	30
ACS	$1/C^{nn}$	1	5	0.5	0.5	0.9	–	–	30
RIACO	$1/C^{nn}$	1	5	–	–	0	6	0.4	30

5.4.2 Experimental analysis

Table 5.2 shows the experimental results based on the Eq. (5.4) among NS-ACO, ACS and RIACO to reveal the average performance of the solutions to each DTSP. The corresponding statistical results of the Wilcoxon signed ranks test at a level of significance $\alpha = 0.05$ are displayed in Table 5.3, where the comparison among three algorithms is implemented by the symbols +, − and ∼ which represent the performance of the former algorithm is better, worse and no significance than that of the latter algorithm respectively. From Table 5.2 and Table 5.3, several conclusions can be observed. Firstly, our proposed NS-ACO algorithm generally outperforms ACS and RIACO on all the DTSPs, which declares the neighborhood search is proper and effective for optimizing solutions to find the optimal solution and three operations randomly selected matching the random changing environments well. The framework of NS-ACO not only maintains the diversity of the solutions but also improves the quality of the optimal solution. The results verify NS-ACO has strong robustness and self-adaptability on DTSP and its ability of tracking the optimal solution is significant. Secondly, except NS-ACO, RIACO performs better on small-scale DTSP such as eil51 and pr76, due to its random immigrant scheme. Nevertheless, on the quickly changing environments of the medium and large scale DTSP, the performance of RIACO declines and is inferior to that of ACS. The reason is that the random immigrants enhance the diversity of the population and destroy the pheromone of the current optimal trail but RIACO has insufficient time to find the optimal solution. On the contrary, ACS can obtain the optimal solution since it utilizes the pheromone evaporation to quickly locate in the optimal trail. Thirdly, when addressing the slowly changing DTSP, RIACO always outperforms ACS since its random immigrants expand the search space to acquire a better

Table 5.2: Comparison of the experimental results among NS-ACO, ACS and RIACO.

Algorithm	NS-ACO		ACS		RIACO	
	$f = 10$		$f = 10$		$f = 10$	
m	0.1	0.9	0.1	0.9	0.1	0.9
eil51	476.6	1179.7	491.8	1249.2	481.1	1190.9
pr76	130376.5	321490.3	136592.0	336443.9	133712.0	334157.8
kroB100	26704.9	66891.2	26777.2	69894.8	27531.1	69750.6
pr124	71915.8	182928.4	72212.2	190884.8	75817.7	198052.5
kroB150	32553.6	81293.5	32556.3	83854.5	34181.2	85702.8
kroB200	37119.4	93944.6	37127.9	96157.3	39539.5	99797.8
	$f = 100$		$f = 100$		$f = 100$	
m	0.1	0.9	0.1	0.9	0.1	0.9
eil51	452.4	1107.7	477.4	1179.8	454.1	1112.7
pr76	123818.7	299372.9	133243.4	328153.8	125356.0	304831.6
kroB100	25155.0	61748.0	26229.8	67462.6	25438.2	63048.4
pr124	68090.5	168722.2	70658.7	184965.0	69098.0	174331.6
kroB150	30607.7	73670.1	31700.7	79693.8	31436.0	76062.7
kroB200	34628.4	86780.4	36195.1	92780.8	35664.6	89619.6

solution within enough time whereas ACS is prone to converge to a local optimum due to the slow pheromone evaporation.

To illustrate the convergence characteristics of three algorithms for DTSP intuitively, Fig. 5.1 is plotted to show the performance of each iteration during three algorithms. Fig. 5.1 depicts the small and medium scale DTSP with $m = 0.1$, $f = 10$ or 100 and the large-scale DTSP with $m = 0.9$, $f = 10$ or 100. The horizontal axis and the vertical axis indicate the iteration times and the best-so-far solution respectively. In Fig. 5.1, each DTSP has 10 environmental changes where Fig. 5.1(a)(c)(e) and Fig. 5.1(b)(d)(f) describe 100 and 1000 iterations respectively. From Fig. 5.1, we can observe that our proposed NS-ACO performs best on each DTSP and RIACO outperforms ACS on slowly changing environments and the small-scale DTSP, and ACS is superior to RIACO on the medium and large scale DTSP whose environment changes quickly. Therefore, our conclusion is demonstrated by Fig. 5.1 again.

Table 5.3: Statistical results of the Wilcoxon signed ranks test at a level of significance $\alpha = 0.05$ among NS-ACO, ACS and RIACO.

Algorithm	NS-ACO:ACS				NS-ACO:RIACO				RIACO:ACS			
	$f = 10$		$f = 100$		$f = 10$		$f = 100$		$f = 10$		$f = 100$	
m	0.1	0.9	0.1	0.9	0.1	0.9	0.1	0.9	0.1	0.9	0.1	0.9
eil51	+	+	+	+	+	+	+	+	+	+	+	+
pr76	+	+	+	+	+	+	+	+	+	+	+	+
kroB100	+	+	+	+	+	+	+	+	−	+	+	+
pr124	+	+	+	+	+	+	+	+	−	−	+	+
kroB150	∼	+	+	+	+	+	+	+	−	−	+	+
kroB200	∼	+	+	+	+	+	+	+	−	−	+	+

5.5 Conclusion

In this paper, we proposed NS-ACO algorithm which adopts neighborhood search to deal with the DTSP comprised by random traffic factors. Swap, insertion and 2-opt operations are randomly utilized to optimize the solutions constructed by ants for tracking the optimal solution efficiently and effectively. The short-term memory is used to store several current iteration-best solutions to enhance the diversity of solutions. To demonstrate the superiority of the proposed algorithm, we compare NS-ACO with the conventional ACS and RIACO on different scales DTSP. The experimental results declare that NS-ACO performs significantly on both convergence and solution quality well. It has the proper ability of convergence that is different from the premature phenomenon of ACS , and it increases the diversity of solutions instead of adding random immigrants which have a high risk in destroying the pheromone of the optimal trail. Therefore, it can be concluded that our proposed NS-ACO is superior to the other two algorithms and shows the promising property for solving the DTSPs.

Chapter 6

Conclusions and Future Works

6.1 Conclusions of the Dissertation

In this dissertation, we introduced three hybrid metaheuristics. They are summarized as follows.

In chapter 3, we proposed combining EDA and IA to create a new hybrid that can efficiently solve TSPs. In this method, EDA is used to realize the information exchange during different solutions generated by IA through the probabilistic model. EDA enables IA to quickly converge on promising search areas. To refine the solutions sampled by EDA, a heuristic local search operator is also proposed to repair the infeasible solutions, and further facilitate the search by making use of the problem dependent knowledge of TSP. The effects of two kinds of EDAs including the univariate marginal distribution algorithm and the population-based incremental learning are investigated by taking into consideration average tour length, best tour length, convergence performance and the distribution of solutions on 19 different TSP instances with size up to 100,000 cities. A non-parametric statistical test based on the Wilcoxon signed ranks test and the Friedman test consistently demonstrates the effects of EDA on IA. Intensive comparative results of other six hybrid metaheuristics reported recently in the literature verify the superiority of IA-EDA, and show that IA-EDA can produce better or competitive solutions than other hybrid algorithms within reasonable computational times.

In chapter 4, a novel probability model based particle swarm optimization is proposed to solve the graph planarization problems. The particle swarm optimization is for the first time

used to handle the binary solutions based on the singlerow routing representation where the value 1 represents an edge can be drawn as an upper edge in the planar subgraph, while 0 denotes an lower edge. To further improve the search performance of PSO and alleviate its inherent local minimum trapping problem, a probability vector based on the univariate marginal distribution is incorporated. By doing so, the solutions generated by PSO are manipulated as accumulative information to construct the probability vector, and a random sampling mutation procedure is used to guide the search to promising areas. Experimental results based on nineteen GPP benchmark instances show that the proposed PMPSO can find competitive solutions for GPP when compared it with other algorithms.

In chapter 5, we proposed NS-ACO algorithm which adopts neighborhood search to copy with the DTSP comprised by random traffic factors. Swap, insertion and 2-opt operations are randomly utilized to optimize solutions constructed by ants for tracking the optimal solution efficiently and effectively. The short-term memory is used to store several current iteration-best solutions to enhance the diversity of solutions. To demonstrate the superiority of our proposed algorithm, we compare NS-ACO with the conventional ACS and RIACO on different scale of DTSP. The experimental results suggest that NS-ACO performs significantly on both convergence and solution quality under the whole DTSP. It not only implements fast convergence unlike the premature phenomenon of ACS but also increases the diversity of solutions instead of adding random immigrants which have a high risk to destroy the pheromone of the optimal trail. Therefore, it can be concluded that our proposed NS-ACO is superior to other algorithms and shows the promising property for addressing the DTSP.

6.2 Suggestion for Future Research

Based on the work I have done, I will focus on the following points in my future research. First of all, go a step further on various kinds of algorithm mechanisms of Bio-inspired computation. That is because only with deeper understanding of Bio-inspired computation, it could be possible to keep digging in the study of the construction (or architecture) of new algorithm. Then, improve the performance of the current existent Bio-inspired com-

putation and apply them to solve engineering problems in new fields. Last but not least, Bio-inspired computation can combine with other computational intelligence algorithms for solving much complex problems.

Bibliography

[1] J. S. Levinton, J. E. Ward, and R. J. Thompson, "Biodynamics of particle processing in bivalve molluscs: models, data, and future directions," *Invertebrate Biology*, pp. 232–242, 1996.

[2] E. Mishra, M. Das, and T. Panda, "Swarm intelligence optimization: editorial survey," *International Journal of Emerging Technology and Advanced Engineering*, vol. 3, no. 1, pp. 217–230, 2013.

[3] E. C.-K. Tsao, J. C. Bezdek, and N. R. Pal, "Fuzzy kohonen clustering networks," *Pattern recognition*, vol. 27, no. 5, pp. 757–764, 1994.

[4] N. Siddique and H. Adeli, *Computational intelligence: synergies of fuzzy logic, neural networks and evolutionary computing*. John Wiley & Sons, 2013.

[5] L. Rutkowski, *Computational intelligence: methods and techniques*. Springer Science, Business Media, 2008.

[6] G. Beni and J. Wang, "Swarm intelligence in cellular robotic systems," in *Robots and Biological Systems: Towards a New Bionics*. Springer, 1993, pp. 703–712.

[7] G. G. Meyer, K. Främling, and J. Holmström, "Intelligent products: A survey," *Computers in industry*, vol. 60, no. 3, pp. 137–148, 2009.

[8] L. Kari and G. Rozenberg, "The many facets of natural computing," *Communications of the ACM*, vol. 51, no. 10, pp. 72–83, 2008.

[9] D. W. Mock, *More than kin and less than kind: The evolution of family conflict*. Harvard University Press, 2004.

[10] K. Sims, "Evolving 3d morphology and behavior by competition," *Artificial life*, vol. 1, no. 4, pp. 353–372, 1994.

[11] C. Nüsslein-Volhard and E. Wieschaus, "Mutations affecting segment number and polarity in drosophila," *Nature*, vol. 287, no. 5785, pp. 795–801, 1980.

[12] W. H. Durham, *Coevolution: Genes, culture, and human diversity*. Stanford University Press, 1991.

[13] R. Poli, "Analysis of the publications on the applications of particle swarm optimisation," *Journal of Artificial Evolution and Applications*, vol. 2008, p. 3, 2008.

[14] A. P. Engelbrecht, *Computational intelligence: an introduction*. John Wiley & Sons, 2007.

[15] R. Srinivasan, "Xdr: External data representation standard," Tech. Rep., 1995.

[16] W. Banzhaf, "Genotype-phenotype-mapping and neutral variation?a case study in genetic programming," in *International Conference on Parallel Problem Solving from Nature*. Springer, 1994, pp. 322–332.

[17] A. Papagelis and D. Kalles, "Breeding decision trees using evolutionary techniques," in *ICML*, vol. 1, 2001, pp. 393–400.

[18] L. Spector and A. Robinson, "Genetic programming and autoconstructive evolution with the push programming language," *Genetic Programming and Evolvable Machines*, vol. 3, no. 1, pp. 7–40, 2002.

[19] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin, "Aspect-oriented programming," in *European conference on object-oriented programming*. Springer, 1997, pp. 220–242.

[20] K. E. Kinnear, *Advances in genetic programming*. MIT press, 1994, vol. 1.

[21] J. L. Pressman and A. B. Wildavsky, *Implementation: how great expectations in Washington are dashed in Oakland: or, why it's amazing that federal programs work*

at all, this being a saga of the Economic Development Administration as told by two sympathetic observers who seek to build morals on a foundation of ruined hopes. Univ of California Press, 1984.

[22] J. Knowles, "Parego: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 50–66, 2006.

[23] P. Sahoo, "Exergoeconomic analysis and optimization of a cogeneration system using evolutionary programming," *Applied thermal engineering*, vol. 28, no. 13, pp. 1580–1588, 2008.

[24] M. Dorigo and L. M. Gambardella, "Ant colonies for the travelling salesman problem," *BioSystems*, vol. 43, no. 2, pp. 73–81, 1997.

[25] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling salesman problem," *Operations Research*, vol. 21, no. 3, pp. 498–516, 1973.

[26] T. Stutzle and H. Hoos, "Max-min ant system and local search for the traveling salesman problem," in *Evolutionary Computation, 1997., IEEE International Conference on*, 1997, pp. 309–314.

[27] M. Dorigo, "Ant colony optimization," *Scholarpedia*, vol. 2, no. 3, p. 1461, 2007.

[28] M. Mavrovouniotis and S. Yang, "A memetic ant colony optimization algorithm for the dynamic travelling salesman problem," *Soft Computing*, vol. 15, no. 7, pp. 1405–1425, 2011.

[29] M. Guntsch, M. Middendorf, and H. Schmeck, "An ant colony optimization approach to dynamic tsp," in *Genetic and Evolutionary Computation Conference*, 2003.

[30] C. J. Eyckelhof and M. Snoek, "Ant systems for a dynamic tsp," in *Ant Algorithms*. Springer, 2002, pp. 88–99.

[31] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, 2006.

[32] M. Dorigo and T. Stützle, "Ant colony optimization: overview and recent advances," *Techreport, IRIDIA, Universite Libre de Bruxelles*, 2009.

[33] J. Derrac, S. Garcia, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, pp. 3–18, 2011.

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

[35] H. Bersini and F. J. Varela, "Hints for adaptive problem solving gleaned from immune networks," in *International Conference on Parallel Problem Solving from Nature*, 1990, pp. 343–354.

[36] Y. Ishida, "Fully distributed diagnosis by pdp learning algorithm: towards immune network pdp model," in *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*, 1990, pp. 777–782.

[37] S. Endoh, N. Toma, and K. Yamada, "Immune algorithm for n-tsp," in *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*, 1998, pp. 3844–3849.

[38] F. Varela, A. Coutinho, B. Dupire, and N. Nelson, "Cognitive networks: immune, neural and otherwise," 1988.

[39] D. E. Cooke and J. E. Hunt, "Recognising promoter sequences using an artificial immune system." in *ISMB*, 1995, pp. 89–97.

[40] J. E. Hunt and D. E. Cooke, "An adaptive, distributed learning system based on the immune system," in *Systems, Man and Cybernetics, 1995. Intelligent Systems*

for the 21st Century., IEEE International Conference on, vol. 3. IEEE, 1995, pp. 2494–2499.

[41] ——, "Learning using an artificial immune system," *Journal of network and computer applications*, vol. 19, no. 2, pp. 189–212, 1996.

[42] J. R. Quinlan, *C4. 5: programs for machine learning*. Elsevier, 2014.

[43] J. Kolodner, *Case-based reasoning*. Morgan Kaufmann, 2014.

[44] N. K. Jerne, "Towards a network theory of the immune system." in *Annales d'immunologie*, 1974, pp. 373–389.

[45] V. Slavov and N. I. Nikolaev, "Immune network dynamics for inductive problem solving," in *International Conference on Parallel Problem Solving from Nature*. Springer, 1998, pp. 712–721.

[46] N. Mitsumoto, T. Fukuda, F. Arai, H. Tadashi, and T. Idogaki, "Self-organizing multiple robotic system (a population control through biologically inspired immune network architecture)," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 2. IEEE, 1996, pp. 1614–1619.

[47] J. D. Farmer, N. H. Packard, and A. S. Perelson, "The immune system, adaptation, and machine learning," *Physica D: Nonlinear Phenomena*, vol. 22, no. 1, pp. 187–204, 1986.

[48] D.-W. Lee and K.-B. Sim, "Artificial immune network-based cooperative control in collective autonomous mobile robots," in *Robot and Human Communication, 1997. RO-MAN'97. Proceedings., 6th IEEE International Workshop on*. IEEE, 1997, pp. 58–63.

[49] D.-W. Lee, H.-B. Jun, and K.-B. Sim, "Artificial immune system for realization of cooperative strategies and group behavior in collective autonomous mobile robots," in *Proc. of the AROB'99 (International Symposium on Artificial Life and Robotics)*, 1999, pp. 232–235.

[50] A. Ishiguro, S. Ichikawa, and Y. Uchikawa, "A gait acquisition of a 6-legged robot using immune networks," in *Intelligent Robots and Systems' 94.'Advanced Robotic Systems and the Real World', IROS'94. Proceedings of the IEEE/RSJ/GI International Conference on*, vol. 2. IEEE, 1994, pp. 1034–1041.

[51] F. Abbattista, G. Di Gioia, G. Di Santo, and A. M. Fanelli, "An associative memory based on the immune networks," in *Neural Networks, 1996., IEEE International Conference on*, vol. 1. IEEE, 1996, pp. 519–523.

[52] M. Kayama, Y. Sugita, Y. Morooka, and S. Fukuoka, "Distributed diagnosis system combining the immune network and learning vector quantization," in *Industrial Electronics, Control, and Instrumentation, 1995., Proceedings of the 1995 IEEE IECON 21st International Conference on*, vol. 2. IEEE, 1995, pp. 1531–1536.

[53] T. Kohonen, "Learning vector quantization," in *Self-Organizing Maps*. Springer, 1995, pp. 175–189.

[54] P. Hajela, J. Yoo, and J. Lee, "Ga based simulation of immune networks applications in structural optimization," *Engineering Optimization*, vol. 29, no. 1-4, pp. 131–149, 1997.

[55] N. Toma, S. Endo, and K. Yamanda, "Immune algorithm with immune network and mhc for adaptive problem solving," in *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on*, vol. 4. IEEE, 1999, pp. 271–276.

[56] K. Mori, M. Tsukiyama, and T. Fukuda, "Multi-optimization by immune algorithm with diversity and learning. 2nd int. conf. on multi-agent systems," in *Workshop Notes on Immunity-Based Systems*, 1996, pp. 118–123.

[57] D. Dasgupta, "Parallel search for multi-modal function optimization with diversity and learning of immune algorithm," in *Artificial immune systems and their applications*. Springer, 1999, pp. 210–220.

[58] K. Mori, M. Tsukiyama, and T. Fukuda, "Application of an immune algorithm to multi-optimization problems," *TRANSACTIONS-INSTITUTE OF ELECTRICAL ENGINEERS OF JAPAN C*, vol. 117, pp. 593–598, 1997.

[59] ——, "Immune algorithm and its application to factory load dispatching planning. 1994 japan-usa symposium on flexible automation," 1994.

[60] J.-S. Chun, M.-K. Kim, H.-K. Jung, and S.-K. Hong, "Shape optimization of electromagnetic devices using immune algorithm," *IEEE transactions on magnetics*, vol. 33, no. 2, pp. 1876–1879, 1997.

[61] K. Mori, M. Tsukiyama, and T. Fukuda, "Adaptive scheduling system inspired by immune system," in *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*, vol. 4. IEEE, 1998, pp. 3833–3837.

[62] D. Dasgupta, "Immunity-based intrusion detection system: a general framework," in *Proc. of the 22nd NISSC*, vol. 1, 1999, pp. 147–160.

[63] S. Forrest, A. S. Perelson, L. Allen, R. Cherukuri *et al.*, "Self-nonself discrimination in a computer," in *IEEE Symposium on Security and Privacy*. Oakland, 1994, pp. 202–212.

[64] P. D'haeseleer, S. Forrest, and P. Helman, "An immunological approach to change detection: Algorithms, analysis and implications," in *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*. IEEE, 1996, pp. 110–119.

[65] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A sense of self for unix processes," in *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*. IEEE, 1996, pp. 120–128.

[66] S. Forrest, S. A. Hofmeyr, and A. Somayaji, "Computer immunology," *Communications of the ACM*, vol. 40, no. 10, pp. 88–96, 1997.

[67] A. Somayaji, S. Hofmeyr, and S. Forrest, "Principles of a computer immune system," in *Proceedings of the 1997 workshop on New security paradigms*. ACM, 1998, pp. 75–82.

[68] S. A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion detection using sequences of system calls," *Journal of computer security*, vol. 6, no. 3, pp. 151–180, 1998.

[69] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting intrusions using system calls: Alternative data models," in *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on*. IEEE, 1999, pp. 133–145.

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

[71] J. Kim and P. Bentley, "The human immune system and network intrusion detection," in *7th European Conference on Intelligent Techniques and Soft Computing (EUFIT'99), Aachen, Germany*, 1999, pp. 1244–1252.

[72] ——, "Negative selection and niching by an artificial immune system for network intrusion detection," in *Proc. of GECCOf99*, 1999, pp. 149–158.

[73] T.-W. T. Wiedmann, J. Wang, N. B. Pliam, and H. C. Wuh, "Immunotherapy composition and method," Dec. 1 1998, uS Patent 5,843,452.

[74] J. K. Percus, O. E. Percus, and A. S. Perelson, "Predicting the size of the t-cell receptor and antibody combining region from consideration of efficient self-nonself discrimination." *Proceedings of the National Academy of Sciences*, vol. 90, no. 5, pp. 1691–1695, 1993.

[75] J. O. Kephart *et al.*, "A biologically inspired immune system for computers," in *Artificial Life IV: proceedings of the fourth international workshop on the synthesis and simulation of living systems*, 1994, pp. 130–139.

[76] J. O. Kephart, G. B. Sorkin, D. M. Chess, and S. R. White, "Fighting computer viruses," *Scientific American*, vol. 277, no. 5, pp. 56–61, 1997.

[77] J. O. Kephart, G. B. Sorkin, and M. Swimmer, "An immune system for cyberspace," in *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, vol. 1. IEEE, 1997, pp. 879–884.

[78] D. Dasgupta and S. Forrest, "Novelty detection in time series data using ideas from immunology," in *Proceedings of the international conference on intelligent systems*, 1996, pp. 82–87.

[79] D. Dasgupta, "Using immunological principles in anomaly detection," *Proceedings of the Artificial Neural Networks in Engineering (ANNIE'96), St. Louis, USA*, 1996.

[80] D. DasGupta, "An overview of artificial immune systems and their applications," in *Artificial immune systems and their applications*. Springer, 1993, pp. 3–21.

[81] M. C. Mackey, L. Glass *et al.*, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, no. 4300, pp. 287–289, 1977.

[82] E. Hart, P. Ross, and J. Nelson, "Producing robust schedules via an artificial immune system," in *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*. IEEE, 1998, pp. 464–469.

[83] K. Kalmanje and J. Neidhoefer, "Immunized adaptive critic for an autonomous aircraft control application," in *Artificial immune systems and their applications*. Springer, 1999, pp. 221–241.

[84] D. F. McCoy and V. Devarajan, "Artificial immune systems and aerial image segmentation," in *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, vol. 1. IEEE, 1997, pp. 867–872.

[85] G. Gutin and A. P. Punnen, *The traveling salesman problem and its variations*. Springer, 2002, vol. 12.

[86] G. Laporte, "The traveling salesman problem: An overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, no. 2, pp. 231–247, 1992.

[87] O. Ergun and J. B. Orlin, "A dynamic programming methodology in very large scale neighborhood search applied to the traveling salesman problem," *Discrete Optimization*, vol. 3, no. 1, pp. 78–85, 2006.

[88] M. Padberg and G. Rinaldi, "Branch-and-cut approach to a variant of the traveling salesman problem," *Journal of Guidance, Control, and Dynamics*, vol. 11, no. 5, pp. 436–440, 1988.

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

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

[91] R. L. Wang, X. F. Zhou, and K. Okazaki, "Ant colony optimization with genetic operation and its application to traveling salesman problem," *IEICE Trans. on Fundamentals of Electronics Communications and Computer Sciences*, vol. E92A, no. 5, pp. 1368–1372, 2009.

[92] M. Mavrovouniotis and S. X. Yang, "A memetic ant colony optimization algorithm for the dynamic travelling salesman problem," *Soft Computing*, vol. 15, no. 7, pp. 1405–1425, 2011.

[93] R. Baraglia, J. I. Hidalgo, and R. Perego, "A hybrid heuristic for the traveling salesman problem," *IEEE Trans. Evolutionary Computation*, vol. 5, no. 6, pp. 613–622, 2001.

[94] C.-F. Tsai, C.-W. Tsai, and C.-C. Tseng, "A new hybrid heuristic approach for solving large traveling salesman problem," *Information Sciences*, vol. 166, no. 1, pp. 67–81, 2004.

[95] H. D. Nguyen, I. Yoshihara, K. Yamamori, and M. Yasunaga, "Implementation of an effective hybrid ga for large-scale traveling salesman problems," *IEEE Trans. on System, Man and Cybernetics, Part B: Cybernetics*, vol. 37, no. 1, pp. 92–99, 2007.

[96] L. N. Xing, Y. W. Chen, K. W. Yang, F. Hou, X. S. Shen, and H. P. Cai, "A hybrid approach combining an improved genetic algorithm and optimization strategies for the asymmetric traveling salesman problem," *Engineering Applications of Artificial Intelligence*, vol. 21, no. 8, pp. 1370–1380, 2008.

[97] W. Lin, J. G. Delgado-Frias, D. C. Gause, and S. Vassiliadis, "Hybrid newton-raphson genetic algorithm for the traveling salesman problem," *Cybernetics and System*, vol. 26, no. 4, pp. 387–412, 1995.

[98] G. Dong, W. W. Guo, and K. Tickle, "Solving the traveling salesman problem using cooperative genetic ant systems," *Expert systems with applications*, vol. 39, no. 5, pp. 5006–5011, 2012.

[99] F. Abbattista, N. Abbattista, and L. Caponetti, "An evolutionary and cooperative agents model for optimization," in *IEEE International Conference on Evolutionary Computation*, vol. 2, 1995, pp. 668–671.

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

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

[102] J. C. Creput and A. Koukam, "A memetic neural network for the euclidean traveling salesman problem," *Neurocomputing*, vol. 72, no. 4-6SI, pp. 1250–1264, 2009.

[103] H.-D. Jin, K.-S. Leung, M.-L. Wong, and Z.-B. Xu, "An efficient self-organizing map designed by genetic algorithms for the traveling salesman problem," *IEEE Trans. on*

Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 33, no. 6, pp. 877–888, 2003.

[104] F. Glover, J. P. Kelly, and M. Laguna, "Genetic algorithms and tabu search: hybrids for optimization," *Computers & Operations Research*, vol. 22, no. 1, pp. 111–134, 1995.

[105] V. A. Shim, K. C. Tan, J. Y. Chia, and J. K. Chong, "Evolutionary algorithms for solving multi-objective travelling salesman problem," *Flexible services and manufacturing journal*, vol. 23, no. 2, pp. 207–241, 2011.

[106] Y. Marinakis, A. Migdalas, and P. M. Pardalos, "A hybrid geneticłgrasp algorithm using lagrangean relaxation for the traveling salesman problem," *Journal of Combinatorial Optimization*, vol. 10, no. 4, pp. 311–326, 2005.

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

[108] O. C. Martin and S. W. Otto, "Combining simulated annealing with local search heuristics," *Annals of Operations Research*, vol. 63, no. 1, pp. 57–75, 1996.

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

[110] X. T. Geng, Z. H. Chen, W. Yang, D. Q. Shi, and K. Zhao, "Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search," *Applied Soft Computing*, vol. 11, no. 4, pp. 3680–3689, 2011.

[111] J. W. Pepper, B. L. Golden, and E. Wasil, "Solving the traveling salesman problem with annealing-based heuristics: a computational study," *IEEE Trans. on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 32, no. 1, pp. 72–77, 2002.

[112] V. A. Shim, K. C. Tan, and C. Y. Cheong, "A hybrid estimation of distribution algorithm with decomposition for solving the multiobjective multiple traveling salesman problem," *IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 42, no. 5, pp. 682–691, 2012.

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

[114] Y. Marinakis, M. Marinaki, and G. Dounias, "Honey bees mating optimization algorithm for the euclidean traveling salesman problem," *Information Sciences*, vol. 181, no. 20, pp. 4684–4698, 2011.

[115] H. Hernandez-Perez, I. Rodriguez-Martin, and J. J. Salazar-Gonzalez, "A hybrid grasp/vnd heuristic for the one-commodity pickup-and-delivery traveling salesman problem," *Computers & Operations Research*, vol. 36, no. 5, pp. 1639–1645, 2009.

[116] Y. Marinakis and M. Marinaki, "A hybrid honey bees mating optimization algorithm for the probabilistic traveling salesman problem," in *IEEE Congress on Evolutionary Computation*, 2009, pp. 1762–1769.

[117] H. W. Dai, Y. Yang, C. H. Li, J. Shi, S. C. Gao, and Z. Tang, "Quantum interference crossover-based clonal selection algorithm and its application to traveling salesman problem," *IEICE Trans. on Information & Systems*, vol. E92-D, no. 1, pp. 78–85, 2009.

[118] 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.

[119] T. A. Masutti and L. N. de Castro, "A self-organizing neural network using ideas from the immune system to solve the traveling salesman problem," *Information Sciences*, vol. 179, no. 10, pp. 1454–1468, 2009.

[120] S. C. Gao, W. Wang, H. W. Dai, F. J. Li, and Z. Tang, "Improved clonal selection algorithm combined with ant colony optimization," *IEICE Trans. on Information & Systems*, vol. E91-D, no. 6, pp. 1813–1823, 2008.

[121] G. Pan, K. Li, A. Ouyang, and K. Li, "Hybrid immune algorithm based on greedy algorithm and delete-cross operator for solving tsp," *Soft Computing*, pp. 1–12, 2014.

[122] S. C. Gao, Z. Tang, H. W. Dai, and J. C. Zhang, "An improved clonal selection algorithm and its application to traveling salesman problems," *IEICE Trans. Fundamentals of Electronics Communications and Computer Sciences*, vol. E90-A, no. 12, pp. 2930–2938, 2007.

[123] Z. Wei, F. Z. Ge, Y. Lu, L. X. Li, and Y. X. Yang, "Chaotic ant swarm for the traveling salesman problem," *Nonlinear Dynamics*, vol. 65, no. 3, pp. 271–281, 2011.

[124] H. Duan and X. Yu, "Hybrid ant colony optimization using memetic algorithm for traveling salesman problem," in *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*. IEEE, 2007, pp. 92–95.

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

[126] T. Saenphon, S. Phimoltares, and C. Lursinsap, "Combining new fast opposite gradient search with ant colony optimization for solving travelling salesman problem," *Engineering Applications of Artificial Intelligence*, vol. 35, pp. 324–334, 2014.

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

[128] W. Elloumi, H. El Abed, A. Abraham, and A. M. Alimi, "A comparative study of the improvement of performance using a pso modified by aco applied to tsp," *Applied Soft Computing*, vol. 25, pp. 234–241, 2014.

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

[130] M. Saadatmand-Tarzjan, M. Khademi, M. R. Akbarzadeh-T, and H. A. Moghaddam, "A novel constructive-optimizer neural network for the traveling salesman problem," *IEEE Trans. on Systems, Man and Cybernetics Part B: Cybernetics*, vol. 37, no. 4, pp. 754–770, 2007.

[131] M. Al-Mulhem and T. Al-Maghrabi, "Efficient convex-elastic net algorithm to solve the euclidean traveling salesman problem," *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 28, no. 4, pp. 618–620, 1998.

[132] E.-G. Talbi, "A taxonomy of hybrid metaheuristics," *Journal of Heuristics*, vol. 8, no. 5, pp. 541–564, 2002.

[133] D. Dasgupta, S. Yu, and F. Nino, "Recent advances in artificial immune systems: Models and applications," *Applied Soft Computing*, no. 11, pp. 1574–1587, 2011.

[134] J. Timmis, "Artificial immune systems: Today and tomorrow," *Natural Computing*, vol. 6, no. 1, pp. 1–18, 2007.

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

[136] B. Haktanirlar Ulutas and S. Kulturel-Konak, "A review of clonal selection algorithm and its applications," *Artificial Intelligence Review*, vol. 36, no. 2, pp. 117–138, 2011.

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

[138] R. C. Liu, L. C. Jiao, and H. F. Du, "Clonal strategy algorithm based on the immune memory," *Journal of Computer Science and Technology*, vol. 20, no. 5, pp. 728–734, 2005.

[139] R. Liu, L. Jiao, Y. Li, and J. Liu, "An immune memory clonal algorithm for numerical and combinatorial optimization," *Frontiers of Computer Science in China*, vol. 4, no. 4, pp. 536–559, 2010.

[140] S. C. Gao, H. Dai, G. Yang, and Z. Tang, "A novel clonal selection algorithm and its application to traveling salesman problem," *IEICE Trans. on Fundamentals of Electronics Communications and Computer Science*, vol. E90-A, no. 10, pp. 2318–2325, 2007.

[141] S. C. Gao, H. Dai, J. Zhang, and Z. Tang, "An expanded lateral interactive clonal selection algorithm and its application," *IEICE Trans. on Fundamentals of Electronics Communications and Computer Science*, vol. E91-A, no. 8, pp. 2223–2231, 2008.

[142] S. C. Gao, R.-L. Wang, M. Ishii, and Z. Tang, "An artificial immune system with feedback mechanisms for effective handling of population size," *IEICE Trans. on Fundamentals of Electronics Communications and Computer Science*, vol. E93-A, no. 2, pp. 532–541, 2010.

[143] D. Karapetyan and G. Gutin, "Lin-kernighan heuristic adaptations for the generalized traveling salesman problem," *European Journal of Operational Research*, vol. 208, no. 3, pp. 221–232, 2011.

[144] H.-K. Tsai, J.-M. Yang, Y.-F. Tsai, and C.-Y. Kao, "An evolutionary algorithm for large traveling salesman problems," *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, no. 4, pp. 1718–1729, 2004.

[145] M. Pelikan, M. W. Hauschild, and F. G. Lobo, "Estimation of distribution algorithms," in *Springer Handbook of Computational Intelligence*. Springer, 2015, pp. 899–928.

[146] M. Hauschild and M. Pelikan, "An introduction and survey of estimation of distribution algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 3, pp. 111–128, 2011.

[147] C.-H. Chen and Y.-P. Chen, "Quality analysis of discretization methods for estimation of distribution algorithms," *IEICE Trans. on Information & Systems*, vol. 97, no. 5, pp. 1312–1323, 2014.

[148] P. Larranaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Press, 2001.

[149] V. A. Shim, K. C. Tan, and C. Y. Cheong, "A hybrid estimation of distribution algorithm with decomposition for solving the multiobjective multiple traveling salesman problem," *IEEE Trans. on System, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 42, no. 5SI, pp. 682–691, 2012.

[150] J. J. Hopfield and D. W. Tank, "Neural computation of decision in optimization problems," *Biol. Cybern.*, vol. 52, pp. 141–152, 1985.

[151] D. D. Sleator and R. E. Tarjan, "Self-adjusting binary search trees," *J. Assoc. Comput. Mach.*, vol. 32, pp. 652–686, 1985.

[152] M. Chrobak, T. Szymacha, and A. Krawczyk, "A data structure useful for finding hamiltonian cycles," *Theor. Comput. Sci.*, vol. 71, pp. 419–424, 1990.

[153] M. L. Fredman, D. S. Johnson, L. A. McGeoch, and G. Ostheimer, "Data structure for traveling salesmen," *J. Algorithms*, vol. 18, pp. 432–479, 1995.

[154] G. Reinelt, "Tsplib – a traveling salesman problem library," *ORSA Journal on Computing*, vol. 3, pp. 376–384, 1991.

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

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

[157] T. Nishizeki and N. Chiba, *Planar Graphs: Theory and Algorithms*. North Holland, 1988.

[158] P. Liu and R. Geldmacher, "On the deletion of nonplanar edges of a graph," in *Proc. of the 10th SE conference on combinatorics, graph theory and computing*, Boca Raton, FL, 1977, pp. 727–738.

[159] 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.

[160] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the sixth international symposium on micro machine and human science*, vol. 1. New York, NY, 1995, pp. 39–43.

[161] A. A. A. Esmin, R. A. Coelho, and S. Matwin, "A review on particle swarm optimization algorithm and its variants to clustering high-dimensional data," *Artificial Intelligence Review*, vol. 44, no. 1, pp. 23–45, 2015.

[162] A. Khare and S. Rangnekar, "A review of particle swarm optimization and its applications in solar photovoltaic system," *Applied Soft Computing*, vol. 13, no. 5, pp. 2997–3006, 2013.

[163] K. Kameyama, "Particle swarm optimization-a survey," *IEICE transactions on information and systems*, vol. 92, no. 7, pp. 1354–1361, 2009.

[164] Y. Takefuji and K. C. Lee, "A near-optimum parallel planarization algorithm," *Science*, vol. 245, no. 4923, pp. 1221–1223, 1989.

[165] O. Goldschmidt and A. Takvorian, "An efficient graph planarization two-phase heuristics," *Networks*, vol. 24, pp. 69–73, 1994.

[166] 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. E92-D, no. 12, pp. 2498–2507, 2009.

[167] M. G. C. Resende and C. C. Ribeiro, "A GRASP for graph planarization," *Networks*, vol. 29, pp. 173–189, 1997.

[168] R. L. Wang, Z. Tang, and Q. P. Cao, "An efficient parallel algorithm for planarization problem," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 49, no. 3, pp. 397–401, 2002.

[169] R. L. Wang and K. Okazaki, "Solving the graph planarization problem using an improved genetic algorithm," *IEICE Trans. Fundamentals*, vol. E89-A, no. 5, pp. 1507–1512, 2006.

[170] 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.

[171] M. Dorigo, M. Birattari, and T. Stützle, "Ant colony optimization," *Computational Intelligence Magazine, IEEE*, vol. 1, no. 4, pp. 28–39, 2006.

[172] M. Drigo, V. Maniezzo, and A. Colorni, "The ant system: optimization by a colony of cooperation agents," *IEEE Transactions of Systems, Man, and Cybernetics*, no. Part B, pp. 29–41, 1996.

[173] G. Di Caro, F. Ducatelle, and L. M. Gambardella, "Anthocnet: an ant-based hybrid routing algorithm for mobile ad hoc networks," in *Parallel Problem Solving from Nature-PPSN VIII*. Springer, 2004, pp. 461–470.

[174] M. Dorigo and G. D. Caro, "The ant colony optimization meta-heuristic," *New Ideas in Optimization*, vol. volume 28, no. 3, pp. 11–32, 1999.

[175] A. E. Rizzoli, R. Montemanni, E. Lucibello, and L. M. Gambardella, "Ant colony optimization for real-world vehicle routing problems," *Swarm Intelligence*, vol. 1, no. 2, pp. 135–151, 2007.

[176] C. Cruz, J. R. González, and D. A. Pelta, "Optimization in dynamic environments: a survey on problems, methods and measures," *Soft Computing*, vol. 15, no. 7, pp. 1427–1448, 2011.

[177] M. Guntsch and M. Middendorf, "Pheromone modification strategies for ant algorithms applied to dynamic tsp," in *Applications of Evolutionary Computing*. Springer, 2001, pp. 213–222.

[178] M. Mavrovouniotis and S. Yang, "Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors," *Applied Soft Computing*, vol. 13, no. 10, pp. 4023–4037, 2013.

[179] M. Guntsch and M. Middendorf, "A population based approach for aco," in *Applications of Evolutionary Computing*. Springer, 2002, pp. 72–81.