Improved artificial bee colony algorithms for robot path planning

Yibing Cui

CENTRALE LILLE

THESE

Présentée en vue
d'obtenir le grade de

DOCTEUR

En

Spécialité : Automatique, Génie informatique, Traitement du Signal et des Images

Par

Yibing CUI

DOCTORAT DELIVRE PAR CENTRALE LILLE

Titre de la thèse :

Algorithmes améliorés de colonies d'abeilles artificielles pour la planification de la trajectoire des robots

Improved artificial bee colony algorithms for robot path planning

---

Soutenue le 05 Décembre 2022 devant le jury d'examen :

Président	Mme. Nathalie MITTON	Directrice de Recherche, INRIA Lille-Nord Europe
Rapporteur	M. Yangquan CHEN	Professeur, University of California, USA
Rapporteur	M. Andreas RAUH	Professeur, Carl von Ossietzky University of Oldenburg, Germany
Membre	Mme. Nathalie MITTON	Directrice de Recherche, INRIA Lille-Nord Europe
Membre	M. Philippe MATHIEU	Professeur, University of Lille, France
Membre	M. Yongguang YU	Professeur, Beijing Jiaotong University, China
Invité	Mme. Sara IFQIR	Professeur Associée, Centrale Lille, France
Invité	M. Wei HU	Professeur Associé, Beijing Jiaotong University, China
Directeur de thèse	M. Ahmed RAHMANI	Professeur, Centrale Lille, France

Thèse préparée au Centre de Recherche en Informatique Signal et Automatique de Lille
CRIStAL, UMR CNRS 9189 - Centrale Lille
Ecole Doctorale MADIS

À mes parents,

à toute ma famille,

à mes professeurs,

et à mes chèr(e)s ami(e)s.

Acknowledgements

This research work has been realized in Ecole Centrale de Lille, in laboratory "Centre de Recherche en Informatique, Signal et Automatique de Lille (CRIStAL)", with the research team "System of Systems Engineering (SoftE)".

First and foremost, I would like to express my deepest gratitude to my supervisor Prof. Ahmed RAHMANI. Over the past three years, his patient guidance, understanding, and constant support have enabled me to gain a lot, including academic research experience as well as life insights. These lessons learned from him will benefit me significantly in both my professional and personal lives. Moreover, he offered his continuous advice and insight while this thesis has been written.

Words cannot express my appreciation to my thesis reviewers, Prof. Yangquan CHEN and Prof. Andreas RAUH, for their invaluable patience and feedback. My sincere thanks also go to all the jury members for their kind acceptance to participate in my defense. Their insightful comments and expertise shone a light on my drawbacks and the research road in the future.

I would also like to thank Mr. Rochdi MERZOUKI, the leader of the group SoftE in laboratory CRIStAL, for providing us great opportunities to exchange with other scholars. In addition, many thanks to Ms. Sara IFQIR, a very kind teacher in our group. Her patience and knowledge helped me a lot and gave me much encouragement.

At the same time, I would like to extend my sincere thanks to Prof. Yongguang YU for his guidance and precious suggestions. I am also very grateful to Dr. Wei HU for the great help he has given me. He has continued to assist me in many aspects throughout my Ph.D., while his research work has inspired me on the path of research.

During the past three years, I had the pleasure of working with my colleagues Yunlong ZHANG, Xinyong WANG, Yiwen CHEN, and Xiaoqin ZHAI, to name

a few. And special thanks to the staff at Centrale Lille and CRIStAL, Vanessa FLEURY, Bénédicte FIEVET, Dominique DEREMETZ, etc. for their kind help.

I also could not have undertaken this journey without the support from the CSC (China Scholarship Council), who financed my research work.

Finally, and most importantly, I want to thank my dear family and friends for their encouragement, support and love, especially my parents. The trust from all these lovely people has kept my spirits and motivation high throughout this process.

Lille, France *Yibing CUI*
September, 2022

Contents

List of Figures

LIST OF FIGURES

List of Tables

LIST OF TABLES

Abbreviations and notations

List of abbreviations

ABC	Artificial bee colony algorithm
ACO	Ant colony optimization algorithm
CS	Cuckoo search algorithm
DE	Differential evolution algorithm
EAs	Evolutionary algorithms
FA	Firefly algorithm
FES	Number of function evaluations
GA	Genetic algorithm
max_FES	Maximal number of function evaluations
MRPP	Multi-robot path planning
OBL	Opposition-based learning
PSO	Particle swarm optimization algorithm
RL	Reinforcement learning
RPP	Robot path planning
SA	Simulated annealing algorithm
SIAs	Swarm intelligence algorithms
SN	Number of swarm
SRPP	Single robot path planning
TLBO	Teaching-learning based optimization algorithm
TS	Tabu search
ABCDC	ABC algorithm with dynamic population composition
ABC_RL	RL-based ABC algorithm
ABCL	Learning-based ABC algorithm
FOABC	Fractional-order ABC algorithm

List of notations

min	getting the minimum value in a given list of values
max	getting the maximum value in a given list of values
R^n	the n-dimensional Euclidean real vector space
$rand$	random numbers
$\|p\|$	absolute value of a scalar p
$\lceil q \rceil$	ceiling function (the least integer greater than or equal to q)
$\emptyset$	empty set
$\oplus$	entry-wise multiplications

Chapter 1

Introduction

Contents

1.1 Background and motivation

With the accelerated development of technology, optimization problems can be found in a wide range of fields, including engineering, finance, biology, medicine, transportation, robotics, and artificial intelligence, to mention a few. At the same time, people often strive to optimize their profit while minimizing various costs in reality. For instance, the path planning in robotics or intelligent transportation

requires creating an ideal path while taking into account various environmental conditions in order to decrease travel consumption and boost transportation efficiency; the feature selection in image analysis demands for extracting the most relevant features from a large amount of data to reduce the subsequent computational dimensions. In this context, effectively addressing these problems has significant implications for raising productivity and lowering expenses in the related tasks. Nevertheless, resources like time and money are constantly limited in practical applications. Therefore, it is crucial to optimize the utilization of these resources while complying with certain constraints.

Hence, constructing mathematical models becomes a crucial approach for handling various optimization problems effectively. At the same time, numerous optimization methods have been proposed to improve the final results. And researchers have continued to further enhance the approaches up to today due to the increasing requirements in all aspects. In addition, various simulation tools have emerged as crucial research techniques.

To resolve optimization problems, the manner of designing optimization models is also essential. Meanwhile, it is normal that a task can alternatively be accomplished through different ways of problem formulation. After properly defining the objective function, the desired solution can be found via various optimization methods. It is worth pointing out that the solution might be the exact optimal solution or a high-quality one that was produced rapidly. In this case, selecting or designing appropriate optimization algorithms for different situations is indeed a meaningful research direction.

Thus, in this context, lots of optimization methods have been recommended which can be classified in a number of ways. According to the nature of the algorithms, they can be generally categorized into two groups, namely deterministic and stochastic algorithms (Yang, 2020). The former follows a rigorous searching process, which starts from an initial point, then generates the search directions and iteration steps based on specific rules. Afterwards, the algorithm iterates and updates the search coordinate until the termination condition is satisfied. In other words, the deterministic optimization methods obtain the same result over multiple runs. This category includes many traditional approaches which have been proposed since the last century, such as Newton-like methods (e.g., Newton's and Quasi-Newton methods) (Dennis & Moré, 1977; Fletcher, 2013; Li

et al., 2019). In fact, these methods usually reach the optimum through differentiation (Chopra & Ansari, 2022). More precisely, Hessians or gradients need to be evaluated during the searching process. In this case, the deterministic approaches may have difficulties in determining the optimal solution quickly enough when tackling non-continuous, non-convex, or large-scale optimization problems (Flor-Sánchez *et al.*, 2022; Lin *et al.*, 2012). However, it is important to recognize that many real-world optimization problems are complex or have discontinuous objective functions.

The second category of approaches, in contrast, can be distinguished by the fact that randomness is a crucial component of stochastic optimization methods (Yang, 2020). Due to the involvement of randomness, uncertainty exists in their search process as well as the final results. Hence, the solutions obtained by this kind of method are different for each run. Heuristic and meta-heuristic are the two main types of stochastic optimization algorithms, although there is little distinction between them.

The heuristic is a way by trial and error to generate solutions, which comes from a Greek verb ($\varepsilon\upsilon\rho\iota\sigma\kappa\omega$) with the meaning "to find/discover". As mentioned above, the complexity of optimization problems keeps increasing. And solving complex problems requires extensive evaluation to determine an exact solution. Nevertheless, the time taken to find this exact solution is often unacceptable (Pearl, 1984). Moreover, many of the practical applications have been considered as NP-hard problems. And their deterministic polynomial time methods hardly exist (Li & Jiang, 2000). In this context, the heuristic techniques demonstrate their advantages for the aforementioned issues by presenting a way to minimize the number of evaluations and find a solution within a reasonable time. In fact, feasible solutions can be found to a challenging optimization problem in a reasonable time frame, but there is no guarantee that the solutions are optimal. Note that machine learning methods can also be classified into this category as they aim to improve their performance in iterative trial and error.

The so-called meta-heuristic algorithm is a further extension of the heuristics. The prefix "meta-" denotes "beyond" or "higher-level", and they usually outperform the heuristic methods (Yang, 2020). Meta-heuristics are not problem-specific, and they can find sufficiently good approximate solutions by exploring

the search space. Moreover, they also allow parallel implementation (Abdel-Basset *et al.*, 2018). Such algorithms has become increasingly popular in recent decades because they are simple to understand, gradient-free, flexible, and effective in targeting various practical problems (Chopra & Ansari, 2022). Many of the meta-heuristic algorithms are designed by modeling the intelligent behavior of biological species. More descriptions of meta-heuristic algorithms are presented in the next subsection.

Figure 1.1 is drawn to clearly illustrate the hierarchy of the aforementioned optimization algorithms. As previously mentioned, there are numerous classification schemes for countless optimization algorithms, and here we simply provide one of the most popular ones.

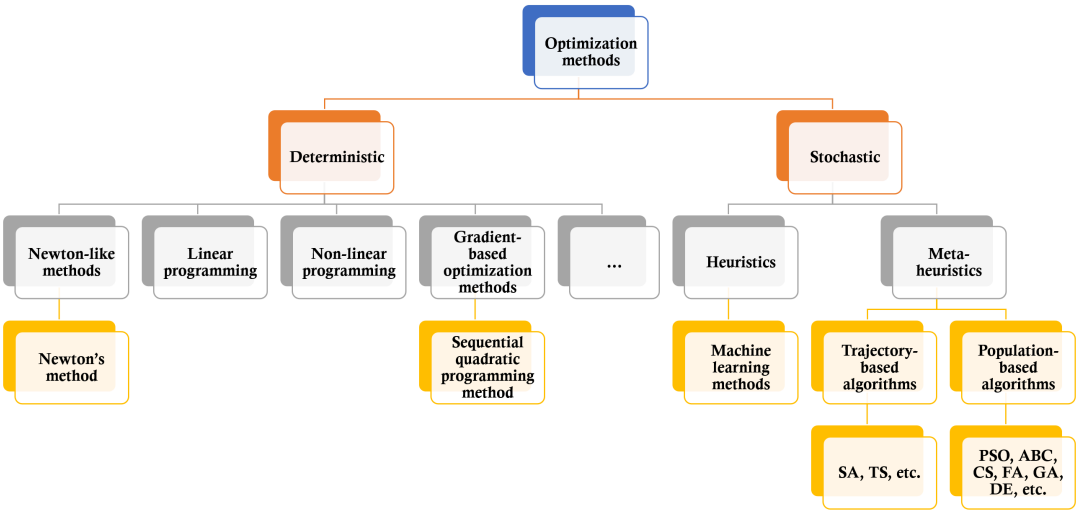


Fig. 1.1. Classification of optimization methods

In addition, another class of optimization methods has emerged in recent years, combining stochastic and deterministic algorithms. The hybrid methods aim to take advantages of both algorithms by utilizing the gradient information as well as the solution search strategies of meta-heuristic algorithms (Dillen *et al.*, 2021; Flor-Sánchez *et al.*, 2022; Gholizadeh *et al.*, 2020).

Based on the background above, it can be observed that meta-heuristic algorithms are very promising optimization methods with a broad variety of applications. Meanwhile, it is important to keep in mind that due to the growing need for effectiveness, every well-known meta-heuristic algorithm can be further enhanced. For instance, the convergence speed and the capability to avoid local

optima are always expected to be better. And precisely handling the trade-off between exploration and exploitation for all optimization tasks is also challenging. Therefore, this thesis focused on a class of meta-heuristic algorithms represented by the Artificial Bee Colony (ABC) algorithm and proposed a series of improved variants by analyzing the characteristics and weaknesses of the ABC algorithm. Furthermore, problems with a higher application value are also taken into account. The proposed enhanced ABC versions have successfully solved different types of optimization problems, including a series of robot path planning problems.

1.2 Overview of meta-heuristic algorithms

The meta-heuristic algorithm has continuously attracted attention because of its outstanding performance in solving tough optimization problems. This kind of algorithm has the advantages of clear structure, high flexibility, and the ability to avoid local optima. They are usually inspired by some intelligent natural phenomena and use mathematical representations to resolve real-world problems. It is worth pointing out that problems are considered black boxes when using meta-heuristic algorithms. In other words, these methods are not problem-specific, which means that they can handle a variety of problems without requiring any changes.

Such algorithms are always examined from the perspectives of exploration and exploitation (diversification and intensification). Exploration is the process of developing the entire search space by producing diverse solutions. The diversity of solutions enables the algorithms to gather more information effectively. Actually, randomness plays a crucial role in keeping the algorithm with good exploration ability. And randomness might provide the algorithm more opportunities to escape when a local optimum is reached. On the other hand, exploitation implies concentrating on the search in promising areas by taking advantage of the current search results. The algorithms are then expected to gradually converge to the optimal solution. Nonetheless, it should be emphasized that attempting to achieve the best in both exploration and exploitation simultaneously is a big challenge. More precisely, if we concentrate on boosting the exploration ability, the solutions will be more dispersed. As a result, the convergence speed of the

algorithm will be slowed down. And if we focus on enhancing the exploitation ability, the algorithm may converge too soon to some seemingly good solutions, which could eventually cause it to miss the actual global optimum. Therefore, for all the meta-heuristic algorithms, it is difficult but essential to find a nice balance between these two components.

Scholars have proposed various ways to classify meta-heuristics, the most common of which are the following two: "trajectory-based & population-based" and "nature-inspired & non-nature-inspired" (Abdel-Basset *et al.*, 2018). Gendreau & Potvin (2005) classified meta-heuristics as trajectory-based and population-based. The trajectory-based methods reach the optimal solution with one initial solution, while the population-based algorithms have a group of randomly generated initial solutions. For instance, the simulated annealing (SA) algorithm (Kirkpatrick *et al.*, 1983) is one of the earliest and most popular trajectory-based algorithms, and particle swarm optimization (PSO) (Kennedy & Eberhart, 1995) is one of the most well-known population-based algorithms. In general, population-based meta-heuristics are more focused on exploration, whereas trajectory-based solutions pay more attention to exploitation.

Meta-heuristics can also be classified into two groups: nature-inspired and non-nature methods (Doering *et al.*, 2019). And the distinction between these categories can be deduced from their names. It is worth mentioning that the second group can be further divided into two types: evolutionary algorithms (EAs) and swarm intelligence algorithms (SIAs). The former such as the genetic algorithm (GA) (Holland, 1975) and the differential evolution (DE) algorithm (Storn & Price, 1997), obtains the best individual through a series of evolutionary operators. Meanwhile, SIAs achieve the optima by starting the search process from different positions simultaneously (Song *et al.*, 2019). Karaboga (2005) believed that self-organization and division of labor are two necessary and sufficient conditions for swarm intelligence. For examples, PSO (Kennedy & Eberhart, 1995), artificial bee colony (ABC) (Karaboga, 2005), and firefly algorithm (FA) (Yang, 2009) are widespread SIAs.

Looking back at history, it is difficult to pinpoint the exact time that meta-heuristics initially appeared. In the 1960s, the genetic algorithm (GA) was created by John Holland and his colleagues (Holland, 1975, 1992), which was very successful and meaningful. GA was inspired by the Darwinian evolution and natural

selection of biological systems. It represented the process via mathematical operators: mutation, crossover, and selection. Moreover, GA is still popular today due to its effectiveness in solving a wide range of optimization problems. Kirkpatrick *et al.* (1983) proposed a local search meta-heuristic based on a physical annealing process for solids, named the simulated annealing (SA) algorithm. It established a link between the thermodynamic behavior and the search for the global minimum of the discrete optimization problems (Nikolaev & Jacobson, 2010). Then, another well-known meta-heuristic algorithm, namely ant colony optimization (ACO), was proposed by Dorigo (1992). The ACO drew inspiration from the foraging behavior of the ant colony. The ants are found to deposit pheromones in the places they pass by, which allows the ant colony to determine the quality of paths. And this mechanism was represented in ACO for solving optimization problems (Dorigo *et al.*, 2006).

Kennedy & Eberhart (1995) proposed the particle swarm optimization (PSO) algorithm inspired by the swarming intelligence of bird flocks and shoals. In PSO, individuals (or particles) start with some initial random positions in the search space. Hence, by communicating the current best solution and sharing the global best solution, the optimal solution can be effectively approached. It can be said that PSO is a very essential SIA, which is beneficial for many subsequent algorithms and their improved versions. Furthermore, Storn & Price (1997) proposed the differential evolution (DE) algorithm, which differs from GA in the reproduction mechanism and the solution forms. DE also contains the three major operators, namely mutation, crossover, and selection. It can effectively obtain information from the population and its parents. This algorithm has also played a crucial role in improving other algorithms. As a result, it can be observed that the two decades of the 1980s and 1990s were the most active periods for meta-heuristic algorithms.

Afterward, Karaboga (2005) proposed the ABC algorithm by simulating the foraging behavior of bee colonies. Correspondingly, three phases were constructed, including employed bees, onlooker bees, and scout bees. Later, Yang & Deb (2009) proposed the cuckoo search (CS) algorithm via Lévy flights (Lévy, 1938; Shlesinger, 1989) inspired by the breeding behavior of cuckoo species. More recently, novel meta-heuristic methods have been constantly developed. For instance, stochastic fractal search (SFS) (Salimi, 2015), monarch butterfly opti-

mization (MBO) (Wang *et al.*, 2019), slime mould algorithm (SMA) (Li *et al.*, 2020b), and colony predation algorithm (CPA) (Tu *et al.*, 2021), etc.

It has been demonstrated that most of these approaches are easy to acquire, simple to implement, and powerful to find out optimum as well. All of these algorithms do, however, have some flaws. Therefore, it is crucial to investigate how to improve their performances and discover more practical applications (Chen *et al.*, 2019c; Das & Jena, 2020; Das *et al.*, 2016b; Gao *et al.*, 2020; Hu *et al.*, 2019; Li & Yin, 2014; Liu, 2016; Zhou *et al.*, 2021b).

1.3 Overview of Artificial bee colony (ABC) algorithm

Among various meta-heuristic methods, the ABC algorithm (Karaboga, 2005) is one of the most popular algorithms. Being inspired by the intelligent foraging behaviors of bee colonies, Karaboga proposed the ABC algorithm. ABC has demonstrated its superiority like fewer parameters and excellent exploration ability compared to the other meta-heuristic algorithms. Meanwhile, its structure is clear and easy to understand. Therefore, it has been widely studied and implemented in various applications. In the followings, the search process of the basic ABC is presented in detail. Then, related literature is reviewed and summarized in subsection 1.3.2.

Remark 1.1 *For simplicity of the descriptions, we assume that the concerned optimization problems can be transformed into minimization problems. Therefore, in the presentation of the algorithms, the objective function value of a candidate solution can directly reveal its quality.*

1.3.1 The standard ABC algorithm

There are three types of bees in ABC to model the foraging behavior of honeybees: employed bees, onlooker bees, and scout bees. The object is to search for the food source with the best quality, where the food source represents the feasible solution to the optimization problem. All the candidate solutions are vectors of D dimensions for solving a D-dimensional problem.

After the initialization phase, the ABC algorithm starts to repeat the three phases until the determination condition is achieved. Firstly, the employed bees are supposed to seek nectars in the search space and share the information with onlooker bees by dancing. In the next step, onlooker bees are supposed to select the food sources found by employed bees according to the nectar qualities and further exploit better food sources around the areas. The scout bees are responsible for keeping the diversity of population by replacing the food sources that haven't been updated during certain cycles. The principle phases are described as follows.

1.3.1.1 Initialization phase

Firstly, SN candidate solutions are randomly generated in the search space. For a $D-$dimensional optimization problem, $x_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,D})$ denotes the i^{th} food source (candidate solution). The following Eq.(1.1) is used for initialization,

$$x_{i,j}^0 = x_{l,j} + rand(0,1) \times (x_{u,j} - x_{l,j}), \tag{1.1}$$

where $x_{i,j}^0$ is the initial value of the j^{th} variable of the i^{th} candidate solution, $i = 1, \cdots, SN$ and $j = 1, \cdots, D$. The $x_{l,j}$ and $x_{u,j}$ denote the lower and upper bounds of the j^{th} dimension, respectively. $rand(0,1)$ is a uniform random number in the range of $[0,1]$. Notice that in the original ABC algorithm, the number of employed bees and onlooker bees are both SN, namely the number of food sources.

1.3.1.2 Employed bee phase

Each employed bee is associated to a candidate solution and is responsible to search for new food source positions. The search strategy in the $(t+1)^{th}$ iteration is expressed as below:

$$v_{i,j} = x_{i,j}^t + \theta_{i,j} \times (x_{i,j}^t - x_{k,j}^t), \tag{1.2}$$

where j is a randomly chosen variable to be updated, and $k \neq i$ is randomly selected among $\{1, \cdots, SN\}$. $\theta_{i,j}$ is the scale factor which is a random real number in $[-1,1]$. And $v_{i,j}$ is obtained considering the information of $x_{i,j}$ and food source $x_{k,j}$ who is chosen from the swarm.

Then v_i is compared to the x_i^t via the objective function $f(\cdot)$. The greedy selection method

$$x_i^{t+1} = \begin{cases} v_i & if\ f(v_i) < f(x_i^t), \\ x_i^t & otherwise, \end{cases} \tag{1.3}$$

is adopted. If the objective function value of v_i is better, v_i will replace x_i^t and the counter $trial_i$ will be reset as 0. Otherwise, x_i remains the same and its $trial_i$ plus one.

1.3.1.3 Onlooker bee phase

In this step, the onlooker bees are expected to further exploit around the promising food sources. So each onlooker bee selects one candidate solution according to the selecting probabilities via the roulette wheel selection method. The fitness values of all the food sources in the $(t+1)^{th}$ interation (fit_i^{t+1}, $i = 1, \cdots, SN$) are evaluated with the following Eq.(1.4),

$$fit_i^{t+1} = \begin{cases} \dfrac{1}{1+f_i^{t+1}} & if\ f_i^{t+1} \geqslant 0, \\ 1 + |f_i^{t+1}| & otherwise, \end{cases} \tag{1.4}$$

where f_i^{t+1} is the objective function value associated with x_i^{t+1}.

Then the corresponding probabilities of all the candidate solutions can be calculated with Eq.(1.5) in terms of their fitness values.

$$p_i^{t+1} = \frac{fit_i^{t+1}}{\sum_{m=1}^{SN} fit_m^{t+1}}. \tag{1.5}$$

It can be found that, when the fitness value of a food source is large, it is considered more qualified. Hence, it is more likely to be selected and further exploited by the onlookers. Then the chosen food sources are updated via Eq.(1.2), and the greedy selection is conducted. If v_i wins the previous food source successfully, then the $trial_i$ will be reset to 0. Otherwise, the $trial_i$ will be added by one.

1.3.1.4 Scout bee phase

In this phase, the counter $trial_i$, $(i = 1, \cdots, SN)$ associated with each candidate solution is compared with the *limit*. If a food source hasn't been improved within predetermined period then it will be abandoned by its employed bee. And a novel food source will be generated with Eq.(1.1). The corresponding counters are reset to zero at the same time.

To represent the framework of ABC more clearly, its flowchart and pseudo-code can be found in Figure 1.2 and Algorithm 1, respectively.

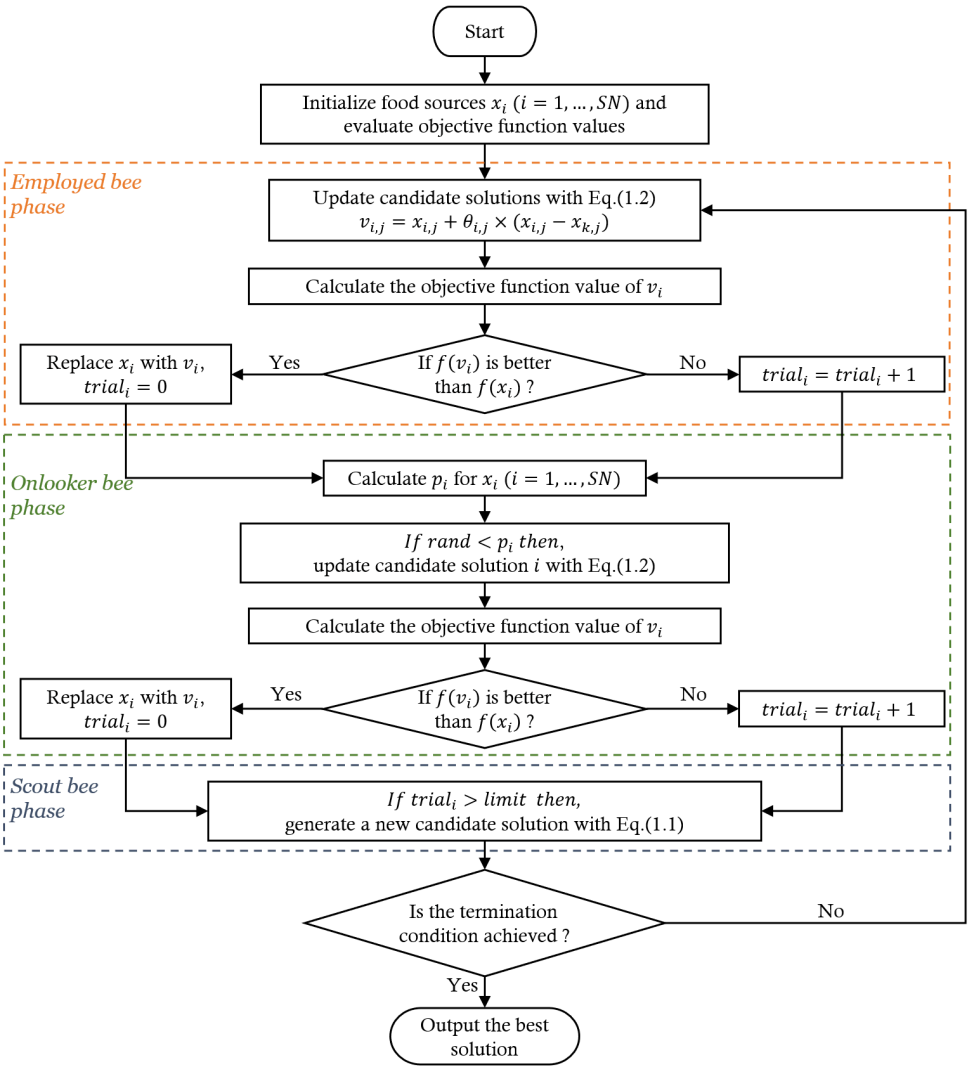


Fig. 1.2. The flowchart of the standard ABC algorithm

Algorithm 1 Pseudo-code of the standard ABC algorithm

Initialize the candidate solutions with Eq.(1.1)
Evaluate the objective function values of the initial population
repeat
    % `The employed bee phase` %
  for $i = 1 \rightarrow SN$ **do**
    Generate v_i with Eq.(1.2) and evaluate v_i
    if $f(v_i) < f(x_i^t)$ **then**
      Replace x_i^t with v_i
      $trial_i = 0$
    else
      $x_i^{t+1} = x^t$, $trial_i = trial_i + 1$
    end if
  end for
  Calculate the fitness values of all the food sources by Eq.(1.4)
  Calculate the probability p_i^{t+1} all the food sources by Eq.(1.5)
    % `The onlooker bee phase` %
  for $t = 1 \rightarrow SN$ **do**
    Select a food source x_i^{t+1} based on the probability values
    Generate v_i with Eq.(1.2) and evaluate v_i
    if $f(v_i) < f(x_i^{t+1})$ **then**
      Replace x_i^{t+1} with v_i
      $trial_i = 0$
    else
      $trial_i = trial_i + 1$
    end if
  end for
    % `The scout bee phase` %
  for $i = 1 \rightarrow SN$ **do**
    if $trial_i > limit$ **then**
      Generate a new x_i^{t+1} with Eq.(1.1) and evaluate its objective function value
      $trial_i = 0$
    end if
  end for
  Store the best solution so far
until the termination condition is reached.

1.3.2 Related work on ABC algorithm improvement

It is widely recognized that ABC does well in diversification but is relatively poor in intensification. In this case, numerous studies have been done attempting to further enhance its performance. The existing improvement strategies can be summarized into three collections: modifying solution search equations, improving the selection mechanism, and hybridizing with other effective algorithms or auxiliary techniques. Of course, there are many improved ABC versions that used more than one category of strategies at the same time.

1.3.2.1 Modification of solution search equations

For meta-heuristic methods, their performance is highly dependent on the way of producing new feasible solutions. And the solution search equation is one of the most essential components of ABC that can considerably affect the final results. Nonetheless, the one-dimensional search equation of ABC was deemed not efficient enough. In this case, basically all the improved ABC versions modified the solution search equation. It has been observed that the search strategies based on PSO and DE algorithms are very popular, which have been incorporated into ABC in many ways.

Being inspired by the PSO algorithm, Zhu & Kwong (2010) proposed gbest-guided ABC (GABC) algorithm that has been widely studied and compared until now. In GABC, information of the global best solution was added to the search equation for enhancing the search efficiency. Actually, this kind of strategy has been used repeatedly in many other related works as the global best solution can provide a useful search direction. Banharnsakun *et al.* (2011) proposed a best-so-far ABC with an enhanced solution update method. More precisely, the information of the current best solution is shared in the onlooker bee phase. Moreover, the areas of searching for new candidates were adjusted during the process. Gao *et al.* (2012) employed the improved search equations which adopted information of the best individual. Meanwhile, being inspired by the mutation operator of DE, the proposed search strategies (i.e., *"ABC/best/1" and "ABC/best/2"*) considered more information about the neighbors. The chaotic maps and the opposition-based learning (OBL) method were also incorporated into the proposed algorithm to help with the convergence rate. Akay & Karaboga (2012) introduced a modification rate (MR) into ABC in order to control the number

of variables that can be inherited from the previous solution, which has a similar role as the crossover rate (CR) in DE. Later Xiang & An (2013) used the best-so-far solution search equation along with another modified equation to update the solutions, and the better one was chosen greedily. Gao *et al.* (2014) proposed an enhanced ABC with two new search equations for the employed bees and onlookers, respectively. The best solution was adopted differently in these two equations. Meanwhile, a new way was utilized to determine and compare the candidate solutions with more robustness. Besides, Imanian *et al.* (2014) introduced more concepts from the PSO algorithm to improve ABC. In other words, the velocities of candidate solutions were updated by using the global best solution and local best position. Gao *et al.* (2015a) proposed a Gaussian-based search equation to produce new candidate solutions in the onlooker bee phase. As a result, the valuable information hidden in the best individual can be exploited.

In addition, the introduction of the concept of the elite is another effective improvement method that aims to better guide the swarm. Xiang *et al.* (2014) proposed a particle swarm inspired multi-elitist ABC algorithm. The food sources were modified using the global best solution and an elitist randomly chosen from an elitist archive. Xiang *et al.* (2015) utilized a multi-objective ABC with an elitism strategy and fixed-size archive. An enhanced search equation was used in employed and onlooker bee phases based on the elites selected from the archive. Cui *et al.* (2016) designed a depth-first search framework. The information of the current best solution and elite solutions were involved in two novel search equations. A novel ABC algorithm was devised by Kong *et al.* (2018) after being impressed by the natural phenomena of following an elite group. Besides, a breadth-first search strategy was adopted in employed bee phase while a stochastic depth-first search strategy was used in onlooker bee phase. A high-efficient ABC variant was proposed (Song *et al.*, 2019). Two novel search strategies considering the best individual were utilized and an elite group was built to generate new solutions via an adaptive selection mechanism.

Babaoglu (2015) introduced a distribution-based solution update rule where the mean and standard deviation of two selected candidate solutions were calculated to generate a new solution. Kıran & Fındık (2015) regarded the single parameter as the reason for the slow convergence rate. To overcome this weakness, authors added the directional information to the algorithm. Zhang *et al.*

(2018) built a cellular structure and adopted a Gaussian-based search equation. The new search equation was combined with a local attractor together to allow the algorithm to converge to the optimum. And a cellular automata model was introduced because it could maintain the population diversity and interaction inside neighborhoods at the same time. Moreover, in (Wang *et al.*, 2020), the best solution within a predefined neighborhood was selected for generating new solutions. Xiao *et al.* (2021) proposed an ABC with adaptive neighborhood size. In the improved ABC algorithm, a global-best-based search equation and a new Gaussian perturbation were adopted. Xiang *et al.* (2021) utilized a pure crossover operation to facilitate information sharing. And a novel frequency of perturbation was proposed to enlarge the number of dimensions to be updated each time.

It is worth mentioning that, in many improved ABC algorithms, the search behaviors of employed bees and onlookers are designed differently, such as (Gao *et al.*, 2014; Karaboga & Gorkemli, 2014; Kong *et al.*, 2018; Song *et al.*, 2017). In addition, researchers began to investigate the way to use multiple search equations with different strengths. Kiran *et al.* (2015) employed five solution search equations as well as a selection mechanism. Lin *et al.* (2018) allowed the onlookers to select one between two improved solution search equations by using an adaptive selection mechanism. They also included the best solution in the employed bee phase, which made the search behaviors directional. Chen *et al.* (2019b) embedded multiple differential search equations into ABC. And a self-adaptive mechanism was proposed to adjust the selection probabilities of those search strategies. Yavuz & Aydın (2019) employed a novel method to decide the expression of solution search equation. A pool of possible terms of the search equations was established. Moreover, a local search method and an increasing population size strategy were incorporated into the algorithm.

Furthermore, some scholars were no longer satisfied with a single search procedure, hence they tried to divide the population into subgroups. The population of the proposed ILABC algorithm was divided into several clusters then the search processes were supposed to conduct simultaneously (Gao *et al.*, 2015b). Two new search mechanisms were utilized to facilitate information exchange inside each group as well as between different subpopulations. Harfouchi *et al.* (2018) proposed a cooperative learning ABC via dividing the population into three subgroups. Each of them could evolve independently with multiple search equations.

1.3.2.2 Novel selection mechanisms

Gao *et al.* (2015a) proposed a fitness-based neighborhood mechanism in order to better exploit the hidden information of promising solutions. Cui *et al.* (2017a) adopted the rankings to select the food sources. In other words, candidate solutions with better rankings have more chances of being selected. Moreover, Cui *et al.* (2017b) improved the probability model which paid attention to the success rate as well as the objective function value. Wang *et al.* (2020) believed that the probability selection method of basic ABC was not effective enough, especially when the number of iterations increased. Then the proposed algorithm produced new solutions by selecting the best solution inside neighborhood radius.

1.3.2.3 Hybridization with other algorithms or techniques

There are different ways of hybridization that can combine the strengths of different methods with the ABC algorithm. Li & Yin (2014) incorporated the DE algorithm into the structure of ABC algorithm. More precisely, the mutation, crossover, and selection operators were adopted in ABC for generating new food sources. Jadon *et al.* (2017) proposed another hybrid algorithm based on DE and ABC in order to better balance the exploration and exploitation abilities. The best individual was involved in the employed bee phase. And DE-inspired search strategy was used in the onlooker bee phase. Furthermore, Cui *et al.* (2020) implemented a hybrid differential ABC algorithm to solve the multi-item replenishment-distribution problem.

Kıran & Gündüz (2013) proposed a recombination-based hybridization of PSO and ABC. The global best solutions and the information of neighboring food sources were used to generate novel solutions. A hybrid PS-ABC was proposed by Li *et al.* (2015) which combined the global search process of ABC with the local search phase in PSO. Gao *et al.* (2016) proposed a different structure in their hybrid algorithm. More precisely, the GABC (Zhu & Kwong, 2010) was combined with the evolutionary operators of DE to learn from the previous experiences accurately. For each individual, the probability of selecting GABC to update the position was determined based on the performances of both approaches during the last generation. Chen *et al.* (2018) combined the teaching-learning-based optimization (TLBO) algorithm with ABC for the solar PV parameter estimation

problems. The teaching phase and learning phase were incorporated into the employed bee phase and onlooker bee phase, respectively.

In addition, Sharma *et al.* (2016) employed the Lévy flights in the solution search equation to improve the local search capability of ABC. Badem *et al.* (2018) proposed a hybrid algorithm based on ABC and limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm to solve functions with a large number of local minima. The Boltzmann selection method was introduced to the onlooker bees in the novel ABC variants (Chen *et al.*, 2019a). And the extremal optimization algorithm was incorporated into the search strategies.

Furthermore, plenty of techniques have been applied to ABC with the aim of enhancing the optimization performance. It can be found that the chaotic-based method and OBL were widely used in the initialization phase to enhance the population diversity and convergence rate (Gao *et al.*, 2012, 2016; Hu *et al.*, 2015; Kuang *et al.*, 2014). Xiang & An (2013) utilized chaotic initialization based on logistic equation as well as chaotic search in the scout bee phase. Gao *et al.* (2013) not only modified the solution search equation but also introduced orthogonal experimental design (OED) in an attempt to create a learning strategy for discovering more useful information. Moreover, Rosenbrock's rotational direction method was combined with ABC to improve the exploitation ability (Kang *et al.*, 2011). Ji *et al.* (2019) incorporated a scale-free network into the ABC algorithm to enhance the search competence. Furthermore, Feng *et al.* (2022) combined the random forest method with ABC so that crucial hyper-parameters could be optimized. As a result, the computational efficiency was augmented.

1.3.3 Application prospects

Like the other meta-heuristics, ABC and its improved versions can be used to solve almost any numerical global optimization problem. In this context, solving a group of benchmark functions using the proposed algorithm and comparing the statistical results with other well-known algorithms is a common way of validating the performance of proposed algorithm. For solving the constrained optimization problems, certain additional constraint-handling techniques are required.

Moreover, numerous well-known optimization problems can be resolved effectively by the improved ABC algorithms, such as shop scheduling problem (Li &

Pan, 2015), traveling salesman problem (Rekaby *et al.*, 2013), hybrid flowshop problem (Tao *et al.*, 2022), and so on.

The enhanced ABC algorithms have been applied in numerous fields until now, applications have been proposed such as image processing (Banharnsakun *et al.*, 2011; Gao *et al.*, 2019; Öztürk *et al.*, 2020; Su *et al.*, 2022), feature selection (Zhang *et al.*, 2019; Zorarpacı & Özel, 2016), robot path planning (Contreras-Cruz *et al.*, 2015; Liang & Lee, 2015; Lu *et al.*, 2019; Xu *et al.*, 2020), vehicle routing (Lei *et al.*, 2022), parameter identification (Chen *et al.*, 2019a; Hu *et al.*, 2015, 2018), data clustering (Zabihi & Nasiri, 2018), etc. This indicates that there will certainly be more and more practical problems in the future that these effective ABC algorithms can handle.

1.4 Overview of robot path planning (RPP) problem

Robotics has undergone a huge revolution over the past few decades, and nowadays, related technologies are still being innovated. Numerous robotic systems have been developed and produced, and they have demonstrated their abilities to handle various missions in different situations like industrial manufacturing (Meike & Ribickis, 2011; Park *et al.*, 2012; Xing, 2021; Yin *et al.*, 2019), smart home environments (Khan *et al.*, 2021; Wang *et al.*, 2013; Wilson *et al.*, 2019), warehouses (Jiang *et al.*, 2020; Li *et al.*, 2022; Plaksina *et al.*, 2018), etc. It is obvious that intelligence is necessary for effectively accomplishing the tasks. Nevertheless, in order to possess high intelligence in robotic systems, a number of research issues need to be resolved. One of the major challenges for robotic systems is navigation. It could be stated that navigation and guidance are required for applying robots in any situation. Therefore, it is crucial to ensure efficient and successful navigation, especially considering the fact that robots are facing increasingly complex scenarios now.

In this context, the robots need to be aware of their positions in relation to their goals with the purpose of completing the navigation task. Moreover, in order to increase the chances of success, robots must also consider the dangers in their surroundings and adjust the following actions (Koubâa *et al.*, 2018). With the purpose of achieving navigation and guidance, the process can be divided into

three parts: localization, path or motion planning, and mapping. Firstly, localization indicates that a robot should continuously determine its positions. Many types of tools like sensors and cameras have been adopted to help achieve this task. Hence, the robot should find a path according to its current location, target point, and its view of surrounding environment. In other words, path planning refers to the process of determining the pathway across the environment that allows the robot to reach its desired destination without collisions. Subsequently, the robot is expected to act along the path planned. For this purpose, control theory and automation technique are necessary. Last but not the least, the robot needs a map of the environment. This map is crucial for helping the robot to realize its current location and direction. It is worth mentioning that the map can be initially stored or be constructed progressively as the robot explored the workspace. Correspondingly, the path planning can be divided into global path planning and local path planning depending on whether the environment is known or not. Figure 1.3 below summarizes the differences between them (Koubâa *et al.*, 2018).

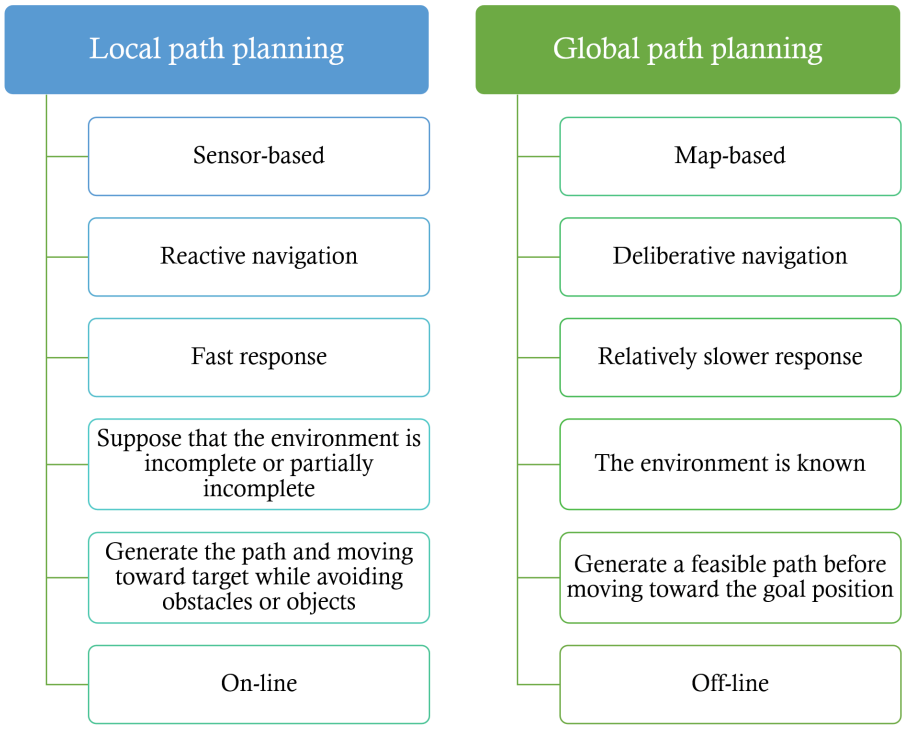


Fig. 1.3. Comparison of global and local path planning

The objective of RPP problem is to find out an optimal collision-free path from start point to target point in an environment with obstacles (Nazarahari *et al.*, 2019). Moreover, path planning for multiple robots has gained increasing attention since these robots can work together to do certain tasks that can not be completed by an individual robot. The multi-robot path planning (MRPP) aims at searching for obstacle-free paths for a group of robots in the same environment. It is also necessary to make sure that there is no collision between any two robots.

With the purpose of accomplishing RPP, a series of optimization problems can be formulated by considering different goals and constraints. There are three main concerns that should be taken into account: efficiency, accuracy, and safety (Châari *et al.*, 2012). In other words, it is essential to determine a feasible path with the least amount of energy and in the shortest amount of time. At the same time, the robot should also be able to safely avoid obstacles nearby.

Path planning approaches can be generally classified into three categories: classical, graph-based, and (meta-)heuristic approaches. Note that because the difference between the heuristics and meta-heuristics is indeed small, there is no clear distinction in the field of robotics. So we organize them into one category in the following discussions related to RPP. First of all, classical approaches like the artificial potential field (APF), roadmap method, and cell decomposition (Chen *et al.*, 2016; Khatib, 1986; Šeda, 2007) were widely used in the period when the path planning problem was just emerging. They were found to be effective in finding feasible paths, however, certain weaknesses were gradually discovered. One obvious drawback is that they are time-consuming, especially in large complex workspaces, since the generated solutions are computationally expensive. Besides, these kinds of methods might fall into local optima (Das & Jena, 2020; Koubâa *et al.*, 2018).

The graph-based search methods, such as the Dijkstra (Dijkstra, 1959) and A* (Hart *et al.*, 1968; Hawa, 2013) algorithms, are also well-known. The feasible paths can be computed in a graph-based environment, such as a grid map. However, the computational complexity may increase greatly in challenging environments. As a result, these traditional methods cannot remain effective when dealing with complex or dynamic environments. Hence, the demand for intelligence has increased when it comes to solving diverse path planning challenges.

In this context, heuristics and meta-heuristics have attracted increasing attention for overcoming the aforementioned limitations. Among numerous heuristic methods, meta-heuristic algorithms are outstanding in solving various optimization problems and are not problem-specific. Hence, the RPP problems can be resolved by transforming them into functional optimization problems. Then, an optimal solution to the problem can be found by adopting meta-heuristic algorithms that are effective and powerful even in complex environments.

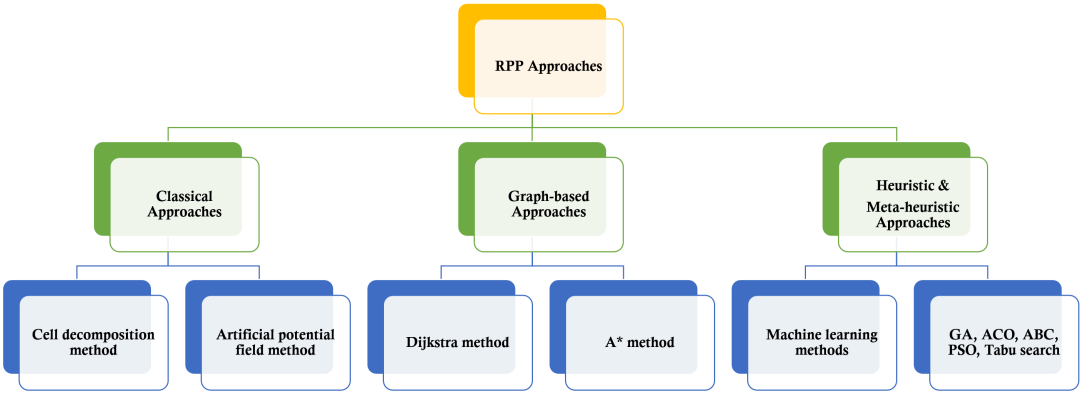


Fig. 1.4. Classification of robot path planning approaches

In recent years, the PSO algorithm (Das *et al.*, 2016a; Das & Jena, 2020; Thabit & Mohades, 2019; Tian *et al.*, 2021), GA Nazarahari *et al.* (2019); Tuncer & Yildirim (2012), ACO (Hasan & Mosa, 2018; Lyridis, 2021; Miao *et al.*, 2021), ABC (Abbas & Ali, 2014; Bhattacharjee *et al.*, 2011; Contreras-Cruz *et al.*, 2015; Liang & Lee, 2015; Wang *et al.*, 2015; Xu *et al.*, 2020), and immune plasma algorithm (IPA) (Aslan, 2022) have all been popular meta-heuristic algorithms in RPP problems.

As mentioned previously, a multi-robot system possesses advantages because of the cooperation and interaction inside the team. And multi-robot collaboration has a stronger ability to resolve complex problems and has higher robustness and reliability (Das & Jena, 2020). In this context, multi-robot systems exist in various fields, for instance, intelligent warehouse (Han & Yu, 2019), oilfield inspection (Li *et al.*, 2020a), etc. As a result of the aforementioned advantages and broad applications, MRPP problems have been investigated by researchers

for decades. The objective of MRPP is to compute collision-free and qualifying paths for a group of robots from an initial configuration to a target configuration via path planners (Han & Yu, 2020).

To name a few, Xu *et al.* (2011) proposed a hierarchical fuzzy control algorithm for achieving MRPP. Moreover, the robots are autonomous and can communicate with each other. Jose & Pratihar (2016) proposed heuristic methods for the task allocation and collision-free path planning for multi-robot system. The GA algorithm was used for minimizing task completion time and the A* algorithm was adopted for path planning. A novel approach for planning paths for centralized and competitive multi-robot was presented (Hasan & Mosa, 2018). The hybrid method combined the pheromone trail updating of the MAX–MIN ACO algorithm with D* algorithm strategies. By locating and displaying the best path for each robot, the robots use tour construction probabilities to select the best way in the dynamic environments. Thabit & Mohades (2019) applied a multi-objective PSO to the MRPP problem considering the path shortness, safety, and smoothness. In addition, a probabilistic window was introduced in order to search for better paths. Nazarahari *et al.* (2019) proposed an enhanced GA for resolving multi-objective MRPP problems. A deterministic algorithm, the APF algorithm, was used to produce feasible initial paths. Then the proposed GA was supposed to optimize those paths. Sahu *et al.* (2022) planned paths for the twin robot via a hybrid algorithm that combined the improved Q-learning and democratic robotics PSO.

1.5 Preliminaries

From the literature above, it can be observed that some famous meta-heuristic algorithms have effective search strategies that can help to further improve other algorithms. So in the following parts, several other well-known meta-heuristics are introduced.

1.5.1 Differential evolution (DE) algorithm

The DE algorithm (Storn & Price, 1997) is a simple yet effective EAs, which attempts to evolve a population of NP individuals to the global optimum. For solving an optimization of D dimensions, the candidate solutions at g^{th} generation

can be represented as $x_i^g = (x_{i,1}^g, x_{i,2}^g, \cdots, x_{i,D}^g)$, where $i = 1, \cdots, NP$. The initial population is supposed to better cover the whole search space as much as possible via uniform randomization (Qin *et al.*, 2009). The same equation (i.e., Eq.(1.1)) is used for generating the initial solutions. Then, DE enters a loop containing three major steps until the termination condition is reached.

1.5.1.1 Mutation operation

In each generation, the individuals first generate their corresponding mutant vectors v_i^g via a mutation operator. Many mutation strategies have been proposed for the DE algorithm. And the most frequently used ones are presented as follows.

1) "DE/rand/1"

$$v_i^g = x_{r_1}^g + F \times (x_{r_2}^g - x_{r_3}^g). \tag{1.6}$$

2) "DE/best/1"

$$v_i^g = x_{best}^g + F \times (x_{r_1}^g - x_{r_2}^g). \tag{1.7}$$

3) "DE/rand/2"

$$v_i^g = x_{r_1}^g + F \times (x_{r_2}^g - x_{r_3}^g) + F \times (x_{r_4}^g - x_{r_5}^g). \tag{1.8}$$

4) "DE/rand-to-best/1"

$$v_i^g = x_i^g + F \times (x_{best}^g - x_i^g) + F \times (x_{r_1}^g - x_{r_2}^g). \tag{1.9}$$

In the above mutation operators, the neighbors $r_1 \neq r_2 \neq r_3 \neq r_4 \neq r_5 \neq i$ are randomly selected among $\{1, \cdots, NP\}$ while x_{best} is the best individual. F is the scale factor which is defined by the users.

1.5.1.2 Crossover operation

Each individual has a pair of vectors, namely the target vector x_i and its mutant vector v_i. Hence, for producing the next generation, individuals need to decide how much information to inherit from their parents. For each dimension, the crossover operation is used to define its value. As a result, a trial vector u_i^g is generated.

$$u_{i,j}^{g} = \begin{cases} v_{i,j}^{g}, \ if\ rand \leqslant CR\ or\ j = j_{rand}, \\ x_{i,j}^{g}, \ otherwise, \end{cases} \tag{1.10}$$

where j_{rand} is randomly selected from $\{1, \cdots, D\}$ while CR is the crossover rate. For each dimension, if the condition of $(rand \leqslant CR\ or\ j = j_{rand})$ is satisfied, the new solution will get information from the mutating vector. Otherwise, it will keep the same as the original x_i.

1.5.1.3 Selection operation

In the next step, the objective function value of the newly generated trial vector u_i^g is compared to that of x_i^g. And the better one will become the offspring. Then the selection operation is performed.

$$x_i^{g+1} = \begin{cases} u_i^g & if\ f(u_i^g) \leqslant f(x_i^g), \\ x_i^g & otherwise. \end{cases} \tag{1.11}$$

In addition, the pseudo-code of basic DE algorithm can be found in Algorithm 2.

Algorithm 2 Pseudo-code of the standard DE algorithm

```
Initialize a population of NP individuals
repeat
    for i = 1 → NP do
      % The mutation operation %
        Generate mutant vector v_i^g with one mutation strategy
      % The crossover operation %
        Randomly select j_rand ∈ {1, ··· , D}
        for j = 1 → D do
            Define the value of u_{i,j}^g with Eq.(1.10)
        end for
      % The selection operation %
        Generate x_i^{g+1} with Eq.(1.11)
    end for
    Update the generation number g = g + 1
until the termination condition is reached.
```

1.5.2 Particle swarm optimization (PSO) algorithm

Kennedy & Eberhart (1995) proposed the PSO algorithm by deriving inspiration from the social behaviors of bird flocks and insect swarms. Each bird in a flock is abstracted as a particle in the algorithm. The flock's flight area corresponds to the search space of concerned optimization problem. And the candidate solutions stand for the food source positions. Similar to the other meta-heuristic algorithms, a parameter of the problem that needs to be optimized is represented by each dimension of this space (Genovesi *et al.*, 2006). The PSO algorithm is able to approach the global optimum by continuously updating the velocity and position information of the swarms.

Each particle flies in the $D-$dimensional search space, and the position of i^{th} particle can be identified by a $x_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,D})$, $i = 1, \cdots, NP$, where NP is the number of particles. Meanwhile, its velocity is represented as $v_i = (v_{i,1}, v_{i,2}, \cdots, v_{i,D})$.

At the beginning, each particle starts the process at an arbitrary location and moves with a random velocity that varies in amplitude as well as direction. In the next step, the PSO starts iterating until the termination condition is satisfied within the set bounds. The strategy for updating the velocity and position of each particle in the $(t+1)^{th}$ iteration is as follows.

$$v_{i,j}^{t+1} = w \times v_{i,j}^{t} + c_1 \times r_1 \times (pbest_{i,j}^{t} - x_{i,j}^{t}) + c_2 \times r_2 \times (gbest_{j}^{t} - x_{i,j}^{t}), \quad (1.12)$$

$$x_{i,j}^{t+1} = x_{i,j}^{t} + v_{i,j}^{t+1}, \quad (1.13)$$

where $v_{i,j}$ is the velocity along the j^{th} dimension of the i^{th} particle. w is the inertia while c_1 and c_2 indicate the cognitive and the social rate, respectively. r_1 and r_2 are random numbers uniformly distributed in the range of $[0, 1]$. $pbest_{i,j}^{t}$ is the value of the j^{th} dimension of the best position that the i^{th} particle has ever visited so far. And $gbest$ represents the global best position discovered by the entire population. Hence, the location can be updated by the Eq.(1.13). Obviously, if the newly produced position is better than the $pbest_i$, then the current best associated to the i^{th} particle will be replaced.

It can be found that the update equation of velocity includes three terms: the actual velocity, the cognitive part based on $pbest_i$, and the social part based on

gbest. The first term provides energy for the particles to fly in the search space. The second term guides a particle towards its own best position, which is related to its independent memory. Moreover, the third term reflects the collaborative behavior among the swarm, so that each particle can move closer to the global historical optimal position. Besides, the pseudo-code of the PSO algorithm can be found in Algorithm 3.

Algorithm 3 Pseudo-code of the standard PSO algorithm

```
Initialize the positions and velocities for the swarm
Evaluate the objective function values of the initial population
Initialize the pbest_i with the particles' current positions
Set gbest as the particle position with the best objective function value
repeat
    for i = 1 → NP do
        for j = 1 → D do
            Update the velocity v_i^t and position x_i^t with Eq.(1.12) and Eq.(1.13)
        end for
        Evaluate the updated x_i^t
        if f(x_i^t) < f(pbest_i^{t-1}) then
            Update pbest_i^t with x_i^t
        end if
        if f(pbest_i^t) < f(gbest^{t-1}) then
            Update gbest^t with pbest_i^t
        end if
    end for
    Update the iteration number t = t + 1
until the termination condition is reached.
```

1.5.3 Cuckoo search (CS) algorithm

Later in the year 2009, being inspired by some cuckoo species' brood parasitism, the CS algorithm was proposed by Yang & Deb (2009). Additionally, the so-called Lévy fight (Lévy, 1938; Shlesinger, 1989) was adopted instead of using the traditional randomization. It is worth mentioning that numerous investigations have revealed that many animal and insect flight patterns exhibit the typical characteristics of Lévy flights. This kind of search manner contains a series of straight flight pathways punctuated by a sudden change in direction.

In nature, the cuckoos search for nests to lay and brood their eggs in a random or random-like manner. In order to clearly characterize the principle of the CS, the following three idealized rules are set (Yang, 2020):

- Each cuckoo lays one egg each time and lays its egg in a randomly chosen nest;
- The best nests with high-quality eggs will be carried over to the next generations;
- The number of available host nests is fixed, and the host bird has a chance of finding a cuckoo egg when it is placed. The probability is defined as $p_a \in [0, 1]$. In this situation, the host bird has two options, either get rid of the egg or just leave and build a new nest.

Note that the last assumption has been approximated by the fraction p_a of the NP nests being replaced by new nests (i.e., new random solutions) for simplicity. Moreover, in the basic CS algorithm, each nest has only one egg. In this case, each nest corresponds to one egg which also indicates one cuckoo.

The initialization method Eq.(1.1), is also used in the CS algorithm to generate the initial positions of NP nests. And for an optimization problem of D dimensions, the positions are also in the form of $x_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,D})$.

In the main loop, each cuckoo performs a Lévy fight with the Eq.(1.14) for producing a new candidate solutions.

$$x_i^{t+1} = x_i^t + \alpha \oplus L\acute{e}vy(s), \tag{1.14}$$

where $\alpha > 0$ is the step size. The product $\oplus$ indicates the entry-wise multiplications. Note that the value of α is defined considering the problem scale. It is usually set as $\alpha = O(L/10)$, where L is the characteristic scale of the problem of interest.

In fact, the Lévy flight is a special case of random walk whose steps obey the Lévy distribution which can be defined as below

$$L\acute{e}vy(\lambda) \sim u = t^{-\lambda}, \tag{1.15}$$

where the stability index $1 < \lambda \leq 3$, while t is a random variable. And the above distribution has an infinite mean with infinite variance. The Lévy distribution

is a kind of heavy-tailed distribution with a quite high probability of generating large steps. Therefore, Lévy flights enable the algorithm to explore the search space efficiently.

Moreover, the CS algorithm generates new solutions via Lévy flights around the best solution obtained so far, which can accelerate the local search. At the same time, to prevent the algorithm from becoming stuck in a local optimum, part of the new solutions should be produced by randomization at locations distant from the current best solution. Correspondingly, the principle steps of the basic CS algorithm are presented in Algorithm 4.

Algorithm 4 Pseudo-code of the CS via Lévy flights

Initialize the positions of host nests x_i, $i = 1, \cdots, NP$
Evaluate the objective function values of x_i
repeat
    for $i = 1 \rightarrow NP$ **do**
        Generate new solution via Eq.(1.14)
        Evaluate its objective function value $f(x_i^{t+1})$
        Randomly select another nest x_k
        if $f(x_i^{t+1}) \leqslant f(x_k^t)$ **then**
            Replace the k^{th} nest with x_i^{t+1}
        end if
    end for
    Abandon a fraction p_a of worst solutions
    Generate $p_a \times NP$ new solutions
    Keep the best solutions
    Rank the solutions and update the current best solution
    Update the iteration number $t = t + 1$
until the termination condition is reached.

1.6 Contributions and outline of dissertation

Briefly, meta-heuristic algorithms are a type of stochastic algorithm through a trade-off between randomization and local search. Such algorithms have been found to be effective and simple to understand. Thus, they have been utilized to solve numerous optimization problems in a variety of fields. Nevertheless, there is still room for improvement, such as easy to be trapped in local optimums or slow convergence speed. And precisely handling the trade-off between exploration and exploitation for all optimization tasks is always challenging. In this context, this thesis focused on a class of meta-heuristic algorithms represented by the Artificial Bee Colony (ABC) algorithm and proposed a series of improved variants by analyzing the characteristics and weaknesses of the ABC algorithm. Furthermore, problems with a higher application value are also taken into account. The proposed enhanced ABC versions have successfully solved different types of optimization problems, including robot path planning tasks for single and multiple robots in various environments. The main contributions and outline of dissertation are summarized as follows.

Chapter 2: In ABC, it can be found that the mission of exploration is mainly accomplished by employed bees whereas the onlookers are responsible for exploiting within certain regions. In addition to those widely mentioned improvement strategies, the impact of population composition is studied in this chapter. Actually, the invariable population composition of a bee colony cannot satisfy the needs of different search stages. In this context, improving the effectiveness of ABC by adjusting the population composition is developed. So, an ABC algorithm with dynamic population composition, namely ABCDC is proposed.

Therefore, the main contributions are as follows: firstly, the Symmetric Latin Hypercube Design (SLHD) is adopted in initialization to improve the population diversity. Secondly, a novel mechanism is proposed to adjust the colony population's composition according to the searching experiences. The number of employed bees decreases periodically while the size of onlooker bees increases to bring more energy for exploiting the global optimum in the mid-late stage of the whole process. In ABCDC, the division of labor between bees with different functions is clearer, so that global optimum can be obtained more efficiently under their cooperation. Moreover, ABCDC keeps a nice balance between diversification and intensification. And experimental studies on functional optimization

problems are done to verify the performance of ABCDC. The comparisons show that ABCDC has better solution precision and a faster convergence rate.

Chapter 3: In fact, it is difficult to define control parameter values appropriately for all types of problems. Thus, these control parameters are usually held constant or updated with predetermined adaptation methods, such as that adopted in ABCDC. However, adaptation approaches still rely heavily on the experience of the designer. In this context, different from the existing literature, a new way of defining parameter values is proposed in this part. An ABC algorithm based on reinforcement learning (RL) is proposed, named ABC_RL. The RL method is used to vary the number of dimensions to be updated in the solution search equation. The reward value of RL is defined based on the update results. In this case, more information can be learned appropriately from the previous update experience.

The main contributions can be summarized as follows: firstly, RL is adopted to enlarge and adjust the frequency of perturbation of employed bee phase intelligently considering the immediate reward from solution update results. Secondly, two enhanced solution search equations are utilized in order to achieve a nice balance between exploration and exploitation. Thirdly, a type of heavy-tailed distribution, the Mittag-Leffler distribution, is used to generate the scale factors of search equations. Finally, the proposed ABC_RL is compared with other improved ABC algorithms on a group of benchmark functions.

Chapter 4: Since one of the most essential goals of improving such algorithms is to solve more practical problems, then its practicality and complexity must be considered. Although many modification strategies are effective in solving functional optimization problems, they do not always assist us in obtaining the optimal solution rapidly in practical applications. Therefore, it is also meaningful to improve the algorithm's performance without overcomplicating it. Therefore, in Chapter 4, enhancing the performance of ABC while avoiding it becoming too complex is investigated. In this context, a learning-based ABC (ABCL) algorithm is proposed. Hence, more energy and time can be saved when solving problems like local path planning.

The main contributions are as follows: firstly, the global best solution is adopted in the employed bee phase and scout bee phase to guide the swarm in a promising search direction. Secondly, learning phase of the TLBO algorithm

is embedded in the onlooker bee phase to improve the exploitation ability and simplify the computational complexity.

Chapter 5: In the proposed ABC variants in Chapters 2-4, the solution search equations are enhanced by enlarging the number of dimensions to be updated and increasing the amount of information that can be gained from the colony. However, this kind of improvement actually ignores some useful information about the individuals' previous search experience. It is worth pointing out that compared to the integer-order derivative, the fractional-order derivative contains entire memory of its previous events. As a result, different from existing results, the fractional-order calculus (FOC) is incorporated into the ABC algorithm considering the memory properties of FOC. In the proposed FOABC algorithm, each time generating a new candidate solution, the previous foraging behaviors stored in memory are considered.

The main contributions of this chapter are as following. The FOC is incorporated into the onlooker bee phase to make full use of the historical experiences. Meanwhile, a differential search strategy is utilized in the employed bee phase to reinforce the exploration ability. And the scale factors of these search equations are generated via Lévy distribution to increase the randomness. In order to validate the performance of FOABC, several groups of comparisons are carried out on a set of benchmark problems.

Chapter 6: After investigating different improvement strategies to improve the effectiveness of the ABC algorithm, we wanted to apply them to some more meaningful problems. Therefore, we attempted to apply these improved ABC algorithms to solve different types of path planning problems. The proposed algorithms are adopted to find better solutions in a limited time after transforming these practical tasks into optimization problems.

First, we used these methods to complete the global path planning for a single robot. Different environments with arbitrary obstacles are considered. Secondly, since multi-robot systems are demonstrating their advantages in more and more fields, we considered this meaningful problem of multi-robot path planning. For all the path planning challenges, the proposed ABC algorithms are compared to other well-known path planners in terms of path length and execution time.

Conclusions and perspectives: The results and findings are summarized and several potential directions for our future research are discussed.

Papers accepted and submitted:

- **Cui, Y.**, Hu, W., & Rahmani, A. (2022). Improved artificial bee colony algorithm with dynamic population composition for optimization problems. *Nonlinear Dynamics*, 107(1), 743-760.

- **Cui, Y.**, Hu, W., & Rahmani, A. (2022). A reinforcement learning based artificial bee colony algorithm with application in robot path planning. *Expert Systems with Applications*, 117389.

- **Cui, Y.**, Hu, W., & Rahmani, A. (2022). Fractional-order artificial bee colony algorithm with application in robot path planning. *European Journal of Operational Research*, https://doi.org/10.1016/j.ejor.2022.11.007.

- **Cui, Y.**, Hu, W., & Rahmani, A.. Multi-robot path planning using learning-based Artificial Bee Colony algorithm. *Engineering Applications of Artificial Intelligence* (Under 1st Review)

Chapter 2

Improved ABC algorithm with dynamic population composition (ABCDC)

Contents

2.1 Introduction

As mentioned in the previous chapter, ABC has been found to be less effective in searching locally when it is evaluated in terms of exploration and exploitation. In this context, numerous possible reasons have been investigated, and many improvement strategies have been proposed.

First of all, one frequently recognized reason for ABC's weakness is the inefficiency of its solution search equations. It can be found that there are two equations to generate and update candidate solutions. The initialization and scout bee phases use the same equation to randomly produce new food source positions in the search space. The employed bee and onlooker bee phases are responsible for updating the candidate solutions based on information about the colony and randomness. In fact, effective solution search equations are supposed to gather useful data. However, the updating equation in ABC only updates one dimension at a time, and there is only one reference neighbor being selected. As a result, one typical enhancement strategy for ABC is to improve its update solution search equation. Notice that the control parameters used in meta-heuristic variants are usually pre-defined by users. To reduce reliance on prior experiences, self-adaptive methods have been proposed which allow the algorithms to set and adjust parameter values during the searching process (Awad *et al.*, 2015; Qin & Suganthan, 2005; Xue *et al.*, 2018; Zhang & Sanderson, 2009). Thus, two enhanced solution search equations with self-adaptive parameters are implemented in the employed bee phase and onlooker bee phase of our proposed ABC variant.

There is another reason which has not been mentioned a lot but is worth investigating, the invariable population composition of the bee colony cannot satisfy the needs of different search stages. Being inspired by the honeybees, the employed bees search over the whole space and then unload the obtained information inside the hive. After that, each onlooker bee chooses one of those food sources and tries to discover better food sources around the chosen one. It can be found that, the mission of exploration is mainly accomplished by employed bees whereas the onlooker bees are responsible for exploiting within certain regions. And in the basic ABC algorithm, the employed bees and onlooker bees have identical sizes. Alizadegan *et al.* (2013) demonstrated that when the number of onlookers is larger than that of employed bees, the algorithm's performance becomes better

in some situations. Hence, it can be concluded that a lack of onlookers is one of the reasons ABC seems unable to effectively reach the global optimum.

To the best of our knowledge, those enhanced ABCs that modified the ratio between employed and onlooker bees use a consistent ratio throughout the searching process (Alizadegan *et al.*, 2013; Hu *et al.*, 2015). And the ratio's setting is heavily influenced by the experiences of users. Therefore, a strategy is provided for dynamically adjusting the sizes of employed and onlooker bees during the search phase. The balance between diversification and intensification can be maintained at the same time.

For the purpose of enhancing the performance of ABC, in this chapter, an improved ABC algorithm with dynamic population composition (ABCDC) is proposed. The division of labor between bees with different roles is more obvious in ABCDC, so that global optimum can be obtained more efficiently under their cooperation. Furthermore, two enhanced solutions search equations are adopted in employed bee and onlooker bee phases according to their responsibilities. Hence, experiments are carried out to assess the performance of the proposed algorithm. ABCDC is first compared to different ABC variations using 22 commonly used benchmark functions (Cui *et al.*, 2017b, 2018; Formica & Milicchio, 2020; Wang *et al.*, 2020; Zhu & Kwong, 2010) with different dimensions. Moreover, comparisons with other non-ABC algorithms are also carried out.

The remaining part of this chapter is organized as follows. The proposed algorithm is introduced in section 2.2. Then, section 2.3 presents experiments and the results. The conclusion is given in the last section 2.4.

2.2 Proposed ABCDC algorithm

The proposed algorithm is introduced in details in this part. More precisely, an initialization method named Symmetric Latin Hypercube Design (SLHD) and two enhanced differential search equations are adopted. Moreover, a novel method for tuning the population composition is proposed to reinforce the intensification as well as diversification.

2.2.1 Improved initialization method

In most meta-heuristic algorithms, the initial candidate solutions are always generated randomly. It is obvious that, the higher the diversity of initial population is, the more efficient the algorithm is in searching the space. Therefore, the initialization is an essential task that can affect the quality of solutions and the convergence rate. In this context, different initialization methods have been proposed in order to enhance the diversity of population. Hence, instead of using the uniform random initialization, approaches like chaotic maps and opposition-based learning (OBL) were utilized in some ABC variants (Gao *et al.*, 2016; Hu *et al.*, 2015).

A random SLHD was employed for generating the initial population in other meta-heuristic algorithms (Regis & Shoemaker, 2004; Zhao *et al.*, 2016). According to the literature, sample points dispersed via SLHD are more uniformly than uniform distributed sample points and the Latin Hypercube Design (LHD) sample points. Moreover, Kenny *et al.* (2000) illuminated how SLHD outperformed LHD considering minimum intersite distance and the entropy.

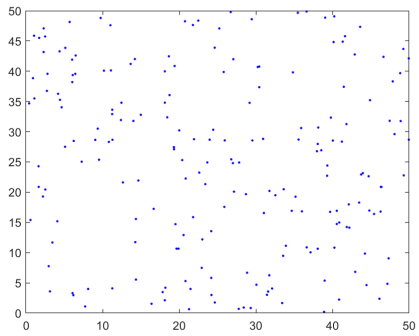
A comparison of the uniform random method and the SLHD in 2D space was also performed to demonstrate the advantage of SLHD. These two methods were used to produce sets of solutions, which are presented in Figure 2.1. The number of sample points are set as 200 and 400 in the two group of tests.

It can be observed that the points generated by the original method (i.e., the uniform random sampling) have some points clustered together while other places have relatively large gaps. Compared with the original method, the SLHD is able to generate more uniformly distributed points so that the space can be explored better. This is critical for the algorithms, particularly at the beginning stage when there is no prior experience.

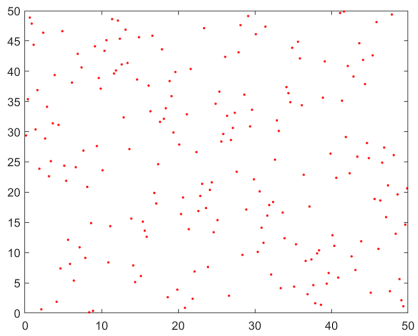
Therefore, the SLHD is incorporated into the initialization phase of ABC algorithm. The pseudo-code of initialization phase via SLHD is presented in Algorithm 5.

2.2.2 Method of dynamic population composition

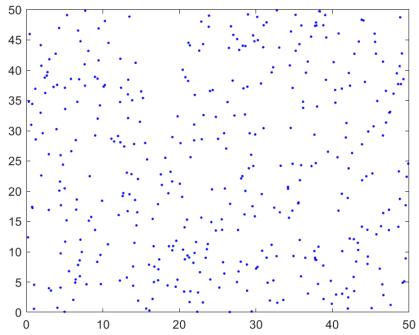
ABC algorithm manages to find the best solution thanks to the cooperation of the three groups of honey bees. The employed bees accomplish the mission of ex-



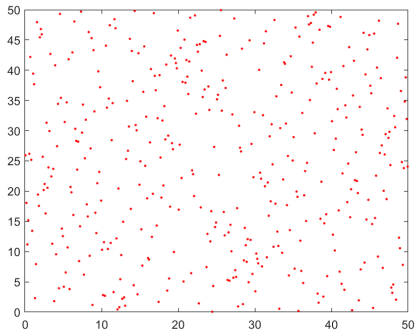
(a) 200 uniform random sample points



(b) 200 SLHD sample points



(c) 400 uniform random sample points



(d) 400 SLHD sample points

Fig. 2.1. Comparison between uniform random sampling and SLHD sampling in 2D.

2. IMPROVED ABC ALGORITHM WITH DYNAMIC POPULATION COMPOSITION (ABCDC)

Algorithm 5 Initialization via SLHD

Initialize an array M of size $SN \times D$
if SN is odd **then** $M(\frac{(SN+1)}{2}, j) = \frac{SN+1}{2}$ for $j = 1, \cdots, D$
end if
$k = \lceil (SN-1)/2 \rceil$ ▷ $\lceil \cdot \rceil$ *is the ceiling function.*
for $j = 1 \to D$ **do**
    Randomly select a permutation of $1, \cdots, k$ and denote it by φ_j
end for
for $i = 1 \to l$ **do**
    for $j = 1 \to D$ **do**
        if $rand(0,1) \leq 0.5$ **then**
            $M(i,j) = \varphi_j(i)$
            $M(SN+1-i,j) = SN+1-\varphi_j(i)$
        else
            $M(i,j) = SN+1-\varphi_j(i)$
            $M(SN+1-i,j) = \varphi_j(i)$
        end if
    end for
end for
$\pi_j = M(:,j)$
Divide $[lower_j, upper_j]$ into SN equal subintervals and $c_j^{(i)}$ denotes the midpoint of the ith subinterval of the jth dimension.
for $i = 1 \to SN$ **do**
    $x_{SLHD}(i) = (c_1^{(\pi_1(i))}, c_2^{(\pi_2(i))}, \cdots, c_D^{(\pi_D(i))})$
end for

ploration while the onlooker bees focus on seeking in the promising regions. And scout bees work for preventing the stagnation of search procedure (Babaoglu, 2015). It is noted that their sizes greatly influence the performance and convergence speed.

Based on the responsibilities of employed and onlooker bees, in the early stage of search process, we hope the size of employed bees is larger than before to search over the whole sample space as much as possible. Hence, when certain promising areas are circled, the principle mission can be shifted to exploitation which is shouldered by onlooker bees. In this way, both of the exploration and exploitation can be accomplished effectively if the composition of population is adjusted timely.

Accordingly, a dynamic population composition mechanism is proposed. With this method, the number of employed bees and onlooker bees can be adjusted so

that they can better perform their respective roles in different search periods. The pseudo-code of the mechanism is given in Algorithm 6.

Algorithm 6 Method of dynamic population composition

Initialize $ratio_E = 0.9$
if $nb\_fail > T\_fail$ **then** ▷ *the situation needs to be adjusted*
  Update the $ratio_E$ with Equation 2.1
  $N_{employed} = \lceil 2 \times SN \times ratio_E \rceil$
  $N_{onlooker} = 2SN - N_{employed}$ ▷ *update the population composition*
  Sort the candidate solutions according to their objective function values
  Take the first $N_{employed}$ candidate solutions
  for $k = 1 \rightarrow N_{employed}$ **do**
    Generate the opposition-based position of x_k via Equation 2.2
    if $f(x_k) <= f(x\_oppo_k)$ **then**
      $x_k \rightarrow x\_new$
    else
      $x\_oppo_k \rightarrow x\_new$
    end if
  end for
  Reset the $nb\_fail = 0$
  $FES = FES + N_{employed}$
end if

The parameter $ratio_E$ is introduced to define and adjust the proportion of employed bees in total population. According to definition in Equation 2.1, $ratio_E$ is initialized to 0.9 in the beginning. Then its value decreases gradually along with the augmentation of function evaluations FES. Note that the max_FES represents the maximal number of function evaluations. The coefficient $a \in (0, 1)$ is important because it decides the change rate of the population composition.

$$ratio_E = 0.9 - a \times \frac{FES}{max\_FES}. \tag{2.1}$$

Remark 2.1 *If a is too large, the number of employed bees quickly reduces, and eventually only few employed bees remain. Whereas, if a is set to a small value, the number of onlooker bees increases too slowly. So, in order to select an appropriate value for a, different values are tested and analyzed in subsection 2.3.2.*

Moreover, a counter nb_fail is introduced to determine when the algorithm needs to adjust the population composition. Firstly, at the end of each iteration,

the best solution in current population is compared to the best solution so far. If the current best solution has smaller objective function value, then the current iteration is considered as one successful experience, otherwise, the counter of fails nb_fail will plus one (lines 26-28 in Algorithm 7). Secondly, this cumulative failed iteration number is compared to a predefined threshold T_fail. In this way, the value of $ratio_E$ can be updated periodically based on the comparison result (line 2 in Algorithm 6). Moreover, if the historical best solution hasn't been updated more than T_fail times, the situation is considered to be improved by switching to another population composition (lines 3-5 in Algorithm 6).

Once the composition is altered, a portion of the employed bees will become onlooker bees, resulting in the abandonment of several food source positions. In order to select the most hopeful candidate solutions, all the solutions are sorted according to their associated objective function values. The top $N_{employed}$ positions will be remained (lines 6-7 in Algorithm 6). Moreover, to avoid losing the population diversity too rapidly, the opposition-based learning (OBL) method is employed on all these remaining solutions. The opposite position of solution can be found by Equation 2.2.

$$x\_oppo_{i,j} = x_{l,j} + x_{u,j} - x_{i,j}, \tag{2.2}$$

where the $x_{l,j}$ and $x_{u,j}$ denote the lower bound and upper bound of the j^{th} variable respectively.

Then, for each left employed bee, it will go to the fitter position by challenging its previous position with the opposite one (lines 10-14 in Algorithm 6). While the more qualified food sources are preserved, some employed bees become onlooker bees and help to exploit around those subsistent nectars. In contrast, if the counter of fails doesn't reach the sill, the bee colony will remain the same to continue searching. Furthermore, the impact of varying the value of parameter T_fail is discussed in subsection 2.3.2.

In addition, the method for dynamic population composition above is able to ensure that the food source positions with disappointing fitness values are removed. And new solutions can still be found by employing OBL. Therefore, ABCDC do not keep the scout bee phase because its work is accomplished by the proposed mechanism.

2.2.3 Two enhanced solution search equations

As mentioned previously, the original ABC algorithm works well in exploration but poorly in the exploitation. One major reason is that the solution search equation Equation 1.2 can only get information from one neighbor at a time, and this information may also be useless. Meanwhile, only one variable of the candidate solution can be updated via this equation. On the other hand, the search strategy of DE is able to learn more information from the population and can vary the number of dimensions to be updated each time. Actually, Equation 1.2 has similar role as the mutation operator in DE. Nonetheless, DE is able to vary the number of updating dimensions through the crossover and selection operators. Thus, being inspired by several enhanced DEs (Brest *et al.*, 2006; Qin & Suganthan, 2005; Zhang & Sanderson, 2009), certain effective differential strategies are incorporated into the proposed ABCDC algorithm.

Qin & Suganthan (2005) proposed an algorithm named SaDE which employs two different mutation operators "*DE/rand/1*" and "*DE/current-to-best/1*" simultaneously. The authors made the choice because "*DE/rand/1*" has shown good diversity while "*DE/current-to-best/1*" has proved outstanding convergence property. Hence, given the responsibilities of employed bees and onlooker bees, applying two different search equations in these phases is naturally regarded as an effective enhancement strategy. Therefore, two search equations based on "*DE/rand/1*" and "*DE/current-to-best/1*" are proposed as follows.

Employed bee phase:

$$v_i = x_{k1} + F_{employed\_i} \times (x_{k2} - x_{k3}), \tag{2.3}$$

Onlooker bee phase:

$$\begin{aligned} v_{i,j} =& x_{i,j} + F_{onlooker\_i} \times (x_{best,j} - x_{i,j}) \\ &+ F_{onlooker\_i} \times (x_{k1,j} - x_{k2,j}), \end{aligned} \tag{2.4}$$

where $k1 \neq k2 \neq k3 \neq i$ are randomly selected from $\{1, \cdots, N_{employed}\}$ and $j \in \{1, \cdots, D\}$ is randomly chosen. $x_{best,j}$ is the jth dimension of the current best individual. The parameters $F_{employed\_i}$ and $F_{onlooker\_i}$ are associated with each employed bee and each onlooker bee, respectively. Different from other

solution search equations, the control parameters $F_{employed\_i}$ and $F_{onlooker\_i}$ are adaptively adjusted based on the successful experiences.

The parameter adaptation method proposed by Zhang & Sanderson (2009) in algorithm JADE is adopted in ABCDC. The parameters $F_{employed\_i}$ and $F_{onlooker\_i}$ are independently created by Cauchy distribution of mean ($\mu_{F\_employed}$ or $\mu_{F\_onlooker}$) and standard deviation 0.1. Each time, if the new position v_i is better than the previous position x_i (i.e., $f(v_i) < f(x_i)$), then the production of new solution will be regarded as a successful search. And the utilized value of F will be recorded into a list $S_{F\_employed}$ or $S_{F\_onlooker}$. The values in lists will be used when re-calculating the parameters $\mu_{F\_employed}$ and $\mu_{F\_onlooker}$ via Equation 2.5 at the end of each iteration. Thus, those $F_{employed\_i}$ and $F_{onlooker\_i}$ who have helped to achieve successful searches will be used to lead the algorithm searching in a promising direction.

$$\mu_F := (1-c) \times \mu_F + c \times mean_L(S_F), \tag{2.5}$$

where $c \in (0,1)$ is a constant. According the suggestions of JADE, $c = 0.1$ while $\mu_{F\_employed}$ and $\mu_{F\_onlooker}$ are initialized to be 0.5. $mean_L(\cdot)$ is the Lehmer mean which is expressed as

$$mean_L(S_F) = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F}. \tag{2.6}$$

Remark 2.2 *Although the parameters $F_{employed\_i}$ and $F_{onlooker\_i}$ are generated and updated in the same manner, they are independent from each other.*

2.2.4 The framework of ABCDC algorithm

In this part, the complete proposed algorithm is presented. To explain the algorithm clearly, the pseudo-code is represented in Algorithm 7 and the corresponding flowchart is given in Figure 2.2.

Remark 2.3 *Note that when a generated position is outside the space, it will be replaced by a new candidate solution which is randomly generated by Equation 1.1.*

Algorithm 7 Pseudo-code of ABCDC algorithm

Initialize $\mu_{F\_employed} = 0.5; \mu_{F\_onlooker} = 0.5$; $N_{employed} = 2SN \times 0.9$; $nb\_fail = 0$
Create $N_{employed}$ initial food sources with Algorithm 5.
Evaluate objective function values of the population, $FES = N_{employed}$
while $FES \leq max\_FES$ **do**
    $S_{F\_employed} = \emptyset$; $S_{F\_onlooker} = \emptyset$
    % `Enhanced employed bee phase` %
    for $i = 1 \rightarrow N_{employed}$ **do**
        $F_{employed\_i} = randCauchy(\mu_{F\_employed}, 0.1)$
        Randomly select $k1 \neq k2 \neq k3 \neq i$ from the colony
        Generate v_i with Eq.(2.3)
        if $f(v_i) \leq f(x_i)$ **then**
            Replace x_i with v_i
            Add $F_{employed\_i}$ into $S_{F\_employed}$
        end if
    end for
    Evaluate the probability values $prob_i$ with Eq.(1.5)
    % `Enhanced onlooker bee phase` %
    while $t < N_{onlooker}$ **do**
        Select x_s by roulette wheel method according $prob$
        $F_{onlooker\_s} = randCauchy(\mu_{F\_onlooker}, 0.1)$
        Randomly select $j \in \{1, \cdots, D\}$ and $k1 \neq k2 \neq i$ from the colony
        Generate v_s with Eq.(2.4)
        if $f(v_s) \leq f(x_s)$ **then**
            Replace x_s with v_s
            Add $F_{onlooker\_s}$ into $S_{F\_onlooker}$
        end if
    end while
    $FES = FES + 2SN$
    if $best_{current} \geq best_{history}$ **then**
        $nb\_fail = nb\_fail + 1$
    end if
    Update the $\mu_{F\_employed}$ and $\mu_{F\_onlooker}$ with Eq.(2.5)
    Update the number of employed bees and onlooker bees with Algorithm 6.
end while

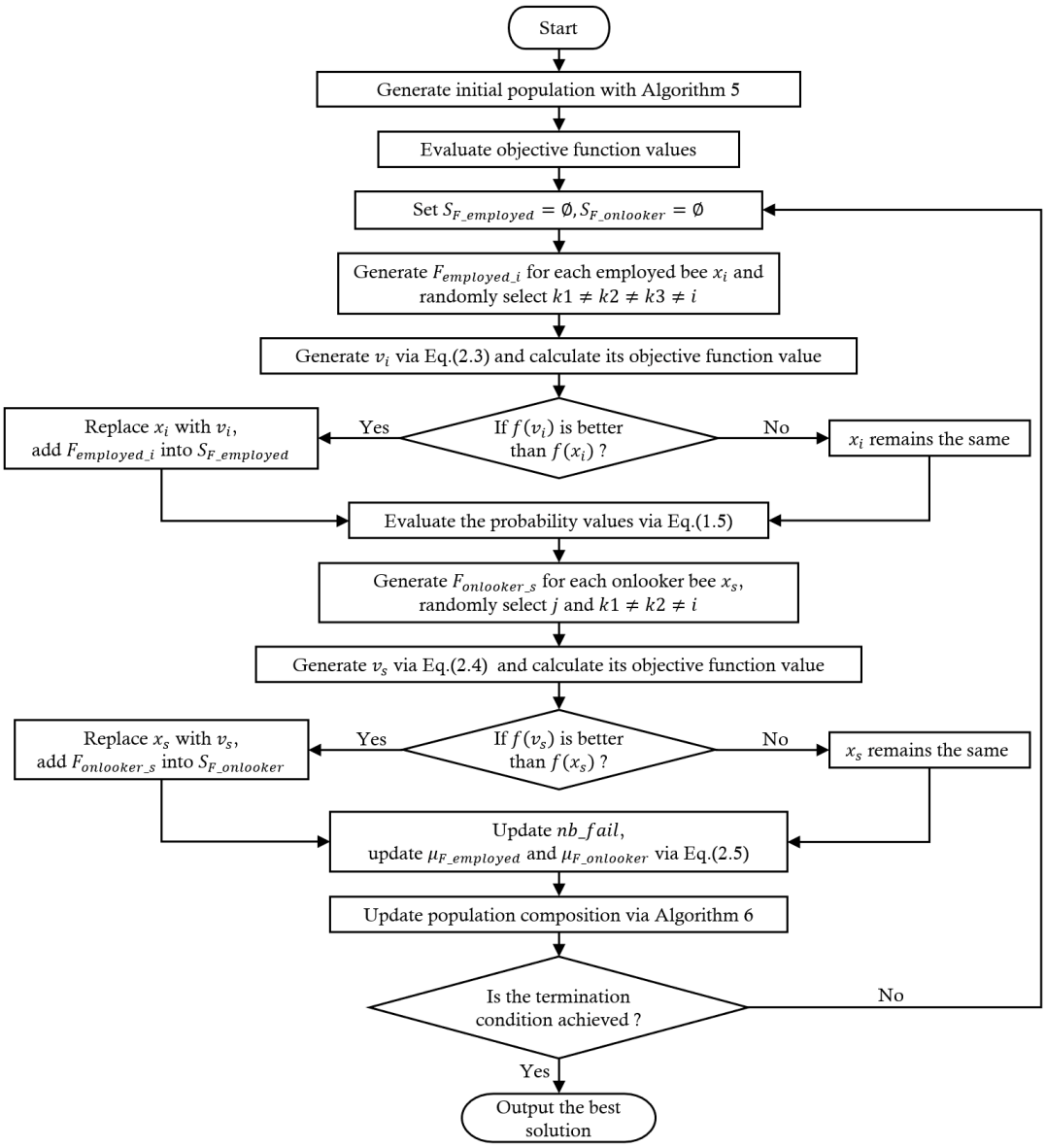


Fig. 2.2. The flowchart of ABCDC algorithm

2.3 Experiments on function optimization problems

In this section, experiments have been done to validate and demonstrate the performance of ABCDC. Firstly, the sensitivity tests and analysis of control parameters are presented in subsection 2.3.2. Then two series of experiments are applied on 22 benchmark functions in different dimensions. The results of ABCDC are compared with several ABC variants and other effective meta-heuristic algorithms, respectively.

In order to do the comparisons fairly, for all the involved algorithms, the determination conditions are respect to the maximum number of function evaluations $max\_FES = 5000 \times D$. Furthermore, all the experimental studies are based on statical results of 25 independent runs. The mean and standard deviation (Std) of errors $f(X_{best}) - f(X^*)$ are calculated and presented in the comparison tables. The $f(X_{best})$ is the best solution found by an algorithm and $f(X^*)$ is the global optimum.

2.3.1 Benchmark functions

22 well-known functions which are widely utilized in the comparisons of optimization methods are chosen as benchmark problems (Cui *et al.*, 2017b, 2018; Farah & Belazi, 2018; Wang *et al.*, 2020; Zhu & Kwong, 2010). The function definitions, corresponding global minimum and the search range are presented in Table 2.1.

Among these optimization problems, $f_1 - f_6$ and $f_8 - f_9$ are uni-modal functions while f_7 is the discontinuous step function and f_{10} is the noisy quartic function. Notice that f_{11} is the Rosenbrock function which is uni-modal problem when D=2 and 3, however, it probably has multiple optima in the higher dimensional cases (Kang *et al.*, 2011). As for the remaining part, $f_{12} - f_{22}$ are multi-modal functions and their local minima augments exponentially when the problem dimension increases.

2.3.2 Sensitivity analysis of the parameters a and T_fail

In the proposed algorithm, parameter a is the change rate of the Equation 2.1 which is able to adjust the population's composition. The $ratio_E$ is the proportion

2. IMPROVED ABC ALGORITHM WITH DYNAMIC POPULATION COMPOSITION (ABCDC)

Table 2.1: 22 benchmark optimization functions

Function	Range	Min
$f_1(\mathbf{x}) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$	0
$f_2(\mathbf{x}) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} x_i^2$	$[-100, 100]^D$	0
$f_3(\mathbf{x}) = \sum_{i=1}^{D} i x_i^2$	$[-10, 10]^D$	0
$f_4(\mathbf{x}) = \sum_{i=1}^{D} \lvert x_i \rvert^{(i+1)}$	$[-1, 1]^D$	0
$f_5(\mathbf{x}) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	$[-10, 10]^D$	0
$f_6(\mathbf{x}) = \max_{i=1,\ldots,n} \lvert x_i \rvert$	$[-100, 100]^D$	0
$f_7(\mathbf{x}) = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	$[-100, 100]^D$	0
$f_8(\mathbf{x}) = \exp(0.5 \sum_{i=1}^{D} x_i)$	$[-10, 10]^D$	0
$f_9(\mathbf{x}) = \sum_{i=1}^{D} i x_i^4$	$[-1.28, 1.28]^D$	0
$f_{10}(\mathbf{x}) = \sum_{i=1}^{D} i x_i^4 + \text{random}[0, 1)$	$[-1.28, 1.28]^D$	0
$f_{11}(\mathbf{x}) = \sum_{i=1}^{D} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-5, 10]^D$	0
$f_{12}(\mathbf{x}) = \sum_{i=1}^{D} [x_i^2 - 10cos(2\pi x_i) + 10]$	$[-5.12, 5.12]^D$	0
$f_{13}(\mathbf{x}) = \sum_{i=1}^{D} [y_i^2 - 10cos(2\pi y_i) + 10]$ $y_i = \left\{ \begin{array}{ll} x_i & \lvert x_i \rvert < \frac{1}{2} \\ \frac{round(2x_i)}{2} & \lvert x_i \rvert \geq \frac{1}{2} \end{array} \right\}$	$[-5.12, 5.12]^D$	0
$f_{14}(\mathbf{x}) = 1 + \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} cos(\frac{x_i}{\sqrt{i}})$	$[-600, 600]^D$	0
$f_{15}(\mathbf{x}) = 418.98288727243380 \times D - \sum_{i=1}^{D} x_i \sin(\sqrt{\lvert x_i \rvert})$	$[-500, 500]^D$	0
$f_{16}(\mathbf{x}) = -20 \exp(-0.2\sqrt{\frac{1}{D} \sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D} \sum_{i=1}^{D} cos(2\pi x_i)) + 20 + e$	$[-50, 50]^D$	0
$f_{17}(\mathbf{x}) = \frac{\pi}{D} \{10 \sin^2(\pi y_1) + \sum_{i=1}^{D-1} (y_1 - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})]$ $+(y_D - 1)^2\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{1}{4}(x_i + 1),\ u_{x_i,a,k,m} = \left\{ \begin{array}{ll} k(x_i - a)^m & x_i > a \\ 0 & -a \leq x_i \leq a \\ k(-x_i - a)^m & x_i < -a \end{array} \right\}$	$[-100, 100]^D$	0
$f_{18}(\mathbf{x}) = \frac{1}{10} \{\sin^2(\pi x_1) + \sum_{i=1}^{D-1} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})] +$ $(x_D - 1)^2 [1 + \sin^2(s\pi x_{i+1})]\} + \sum_{i=1}^{D} u(xi, 5, 100, 4)$	$[-100, 100]^D$	0
$f_{19}(\mathbf{x}) = \sum i = 1^D \lvert x_i sin(x_i) + 0.1 x_i \rvert$	$[-10, 10]^D$	0
$f_{20}(\mathbf{x}) = \sum_{i=1}^{D-1} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})] + \sin^2(3\pi x_1) + \lvert x_D - 1 \rvert [1 + \sin^2(3\pi x_D)]$	$[-10, 10]^D$	0
$f_{21}(\mathbf{x}) = \sum_{i=1}^{D} \left(\sum_{k=0}^{k_{max}} [a^k \cos(2\pi b^k (x_i + 0.5))] \right) - D \sum_{k=0}^{k_{max}} [a^k \cos(2\pi b^k 0.5)]$ $a = 0.5, b = 3, k_{max} = 20$	$[-0.5, 0.5]^D$	0
$f_{22}(\mathbf{x}) = \frac{1}{D} \sum_{i=1}^{D} (x_i^4 - 16 x_i^2 + 5 x_i)$	$[-5, 5]^D$	-78.33236

of employed bees in the total colony and its value reduces linearly along with the increase of iteration. And coefficient a decides the decline rate of employed bees' size. Meanwhile, in ABCDC the number of failures to update the global best solution are counted cumulatively. And T_fail stands for the threshold of adjusting the population's composition, in other words, the frequency of tuning the number of employed bee and onlooker bee depends on this parameter. In this context, it is essential to determine proper values for these two crucial control parameters.

In the following, experiments are developed to investigate the performances of ABCDC with different combinations of a and T_fail. The tests are conducted on 30-dimensional benchmark problems. And the Friedman test is employed to evaluate the mean rankings of concerned ABCDC versions. The results are listed in Table 2.2 and the best ranking is marked in **boldface**. For each function, the average of the errors between the exact minima and the best solutions which are searched in 25 independent runs is calculated.

Table 2.2: Comparison of ABCDC variants with different values of a and T_fail

		T_fail			
		5	10	15	20
a	0.2	16.17	18.74	17.88	17.40
	0.3	15.22	15.43	16.38	16.71
	0.4	13.81	14.43	13.43	14.67
	0.5	28.24	18.84	13.00	13.76
	0.6	11.90	13.05	11.90	19.48
	0.7	10.76	11.95	10.76	13.10
	0.8	**10.00**	11.57	12.19	13.81

According to the mean ranking of different ABCDC versions, it can be found that the best version is the one with $T\_fail = 5$ and $a = 0.8$. If we observe the table in columns (i.e., value of T_fail is fixed), better results are obtained when a is set to be relatively large. Moreover, when we analyze the results in rows, the ABCDC variants who have smaller T_fail can usually possess better outcomes. In other words, on the chosen optimization problems, tuning the population's composition more frequently and using a larger change rate can help to obtain better performance. As a result, the setting $a = 0.8$ and $T\_fail = 5$ will be used in the following studies.

2.3.3 Comparison with ABC variants

In this part, the proposed algorithm is compared to five improved ABC variants and the standard ABC algorithm in order to evaluate its performance. The tests are conducted on the 22 benchmark problems with D=30, 50, and 100.

The reason of choosing DEABC (Li & Yin, 2014) is that DE's mutation and crossover operators are combined in this DE-inspired hybrid algorithm. APABC (Cui *et al.*, 2017b) is an improved ABC with adaptive population size, however, different from our proposed algorithm, the total size is adjusted according to the success rate of honey bees. The ILTD_ABC (Gao *et al.*, 2019) and NSABC (Wang *et al.*, 2020) are proposed recently which can represent the latest direction of improving ABC. Notice that the ILTD_ABC utilizes novel solution search equations which is powerful in converging to the optimum and its code is open to public. To start the comparison fairly, their control parameters are set the same as those of their original papers. The utilized parameter settings are listed in Table 2.3.

Table 2.3: Parameter settings of ABCDC and compared ABC algorithms

Algorithm	Parameter setting
ABC (Karaboga, 2005)	$SN = 50$, $limit = SN \times D$
DEABC (Li & Yin, 2014)	$SN = 50$, $limit = SN \times D$
APABC (Cui *et al.*, 2017b)	$SN = 35$, $SN_{min} = 20$, $SN_{max} = 35$, $T = 20$
ILTD_ABC (Gao *et al.*, 2019)	$SN = 50$, $limit = 100$
NSABC (Wang *et al.*, 2020)	$SN = 50$, $limit = 100$, $k = 10$, $C = 1.5$
ABCDC	$SN = 50$, $\mu_{F\_employed} = 0.5$, $\mu_{F\_onlooker} = 0.5$, $a = 0.8$, $T\_fail = 5$

Table 2.4 - Table 2.6 present the comparison results in terms of the mean and standard deviation (Std) of the errors $f(X_{best}) - f(X^*)$. And for each test function, the results of involved ABC algorithms are compared to that of ABCDC via the Wilcoxon rank sum test at 0.05 significant level. The symbols "+", "=", and "-" denote that ABCDC is better than, similar to, and worse than the compared algorithm, respectively. In addition, the Friedman test is also applied on the results and Figure 2.3 summarizes the average rankings of involved algorithms.

Table 2.4: Comparison between ABCDC and other ABC variants with $D = 30$

Function		ABC		DEABC		APABC		ILTD_ABC		NSABC		ABCDC
f_1	Mean	0.00E+00	-	1.37E-15	+	6.63E-70	+	0.00E+00	-	1.67E-25	+	1.84E-197
	Std	0.00E+00		2.27E-15		1.79E-69		0.00E+00		1.15E-25		0.00E+00
f_2	Mean	5.75E-09	+	1.70E-12	+	1.35E-64	+	0.00E+00	-	2.70E-22	+	4.22E-145
	Std	8.70E-09		1.97E-12		6.51E-64		0.00E+00		2.98E-22		2.11E-144
f_3	Mean	0.00E+00	-	9.77E-17	+	4.35E-69	+	0.00E+00	-	2.02E-26	+	2.91E-188
	Std	0.00E+00		1.38E-16		2.10E-68		0.00E+00		1.52E-26		0.00E+00
f_4	Mean	0.00E+00	=	6.39E-35	+	1.21E-78	+	0.00E+00	=	3.16E-79	+	0.00E+00
	Std	0.00E+00		3.19E-34		3.22E-78		0.00E+00		9.96E-79		0.00E+00
f_5	Mean	0.00E+00	-	1.14E-05	+	2.61E-35	+	0.00E+00	-	3.23E-14	+	1.40E-103
	Std	0.00E+00		2.99E-05		9.99E-35		0.00E+00		1.08E-14		5.89E-103
f_6	Mean	1.17E+01	+	2.11E+00	+	6.11E-01	+	0.00E+00	-	8.64E+00	+	1.12E-299
	Std	2.63E+00		2.39E+00		1.16E-01		0.00E+00		1.75E+00		0.00E+00
f_7	Mean	0.00E+00	=	0.00E+00	=	0.00E+00	=	0.00E+00	=	0.00E+00	=	0.00E+00
	Std	0.00E+00		0.00E+00		0.00E+00		0.00E+00		0.00E+00		0.00E+00
f_8	Mean	0.00E+00	-	7.18E-66	=	7.18E-66	=	7.18E-66	=	0.00E+00	-	7.18E-66
	Std	0.00E+00		3.23E-81		3.23E-81		3.23E-81		0.00E+00		3.23E-81
f_9	Mean	0.00E+00	=	7.35E-31	+	8.75E-140	+	0.00E+00	=	1.51E-55	+	0.00E+00
	Std	0.00E+00		2.08E-30		3.33E-139		0.00E+00		4.18E-55		0.00E+00
f_{10}	Mean	5.78E-02	+	6.26E-02	+	1.92E-02	+	5.68E-06	-	3.24E-02	+	1.99E-05
	Std	1.24E-02		9.65E-03		3.88E-03		5.42E-06		9.96E-03		1.36E-05
f_{11}	Mean	1.34E-01	+	1.14E+01	+	3.52E-01	+	2.64E+01	+	1.19E+01	+	0.00E+00
	Std	9.68E-02		1.71E+00		5.12E-01		1.60E-01		2.06E+01		0.00E+00
f_{12}	Mean	0.00E+00	=	2.29E+02	+	0.00E+00	=	0.00E+00	=	7.96E-02	+	0.00E+00
	Std	0.00E+00		1.08E+01		0.00E+00		0.00E+00		2.75E-01		0.00E+00
f_{13}	Mean	3.17E-09	+	2.08E+02	+	0.00E+00	=	0.00E+00	=	0.00E+00	=	0.00E+00
	Std	1.36E-08		1.55E+01		0.00E+00		0.00E+00		0.00E+00		0.00E+00
f_{14}	Mean	7.36E-10	+	3.18E-02	+	9.06E-13	+	0.00E+00	=	3.33E-13	+	0.00E+00
	Std	3.68E-09		1.30E-01		3.36E-12		0.00E+00		1.66E-12		0.00E+00
f_{15}	Mean	2.67E+01	+	5.70E+01	+	9.48E+00	+	2.13E+02	+	1.41E+47	+	0.00E+00
	Std	4.91E+01		6.03E+01		3.28E+01		1.51E+02		6.89E+47		0.00E+00
f_{16}	Mean	8.67E-04	+	2.69E-08	+	9.76E-02	+	8.88E-16	=	6.41E-12	+	8.88E-16
	Std	1.52E-03		1.99E-08		3.61E-01		0.00E+00		9.28E-12		0.00E+00
f_{17}	Mean	0.00E+00	-	4.15E-03	+	1.57E-32	=	1.57E-32	=	2.41E-27	+	1.57E-32
	Std	0.00E+00		2.07E-02		5.59E-48		5.59E-48		1.61E-27		5.59E-48
f_{18}	Mean	0.00E+00	-	4.15E-03	+	1.57E-32	=	1.57E-32	=	3.86E-27	+	1.57E-32
	Std	0.00E+00		2.07E-02		5.59E-48		5.59E-48		3.23E-27		5.59E-48
f_{19}	Mean	2.09E-05	+	1.97E-02	+	4.39E-29	+	0.00E+00	-	2.58E-14	+	2.38E-92
	Std	1.17E-05		8.31E-03		2.18E-28		0.00E+00		1.38E-14		1.19E-91
f_{20}	Mean	0.00E+00	-	1.10E+01	+	3.91E-30	-	2.45E+01	+	1.10E-23	-	4.96E-11
	Std	0.00E+00		6.79E-01		2.15E-45		1.52E+01		1.24E-23		2.13E-10
f_{21}	Mean	5.33E-07	+	9.29E-04	+	0.00E+00	=	0.00E+00	=	0.00E+00	=	0.00E+00
	Std	4.22E-07		4.68E-04		0.00E+00		0.00E+00		0.00E+00		0.00E+00
f_{22}	Mean	0.00E+00	-	3.39E-01	+	2.86E-05	+	9.00E-05	+	2.86E-05	+	2.86E-05
	Std	0.00E+00		5.36E-01		1.23E-14		1.92E-05		1.15E-14		0.00E+00
Total	+/=/-	10/4/8		20/2/0		14/7/1		4/11/7		17/3/2		

The statical results for 30-dimensional problems can be found in Table 2.4. In fact, the results of solving uni-modal functions ($f_1 - f_9$) can reveal the exploitation ability of concerned algorithms because each problem has only one global optimum. In this case, the proposed ABCDC algorithm has competitive exploitation capability considering its values of mean and standard deviation. For $f_1 - f_3$ and $f_5 - f_6$, ABCDC surpass most of the other ABC variants.

Unlike the uni-modal functions, multi-modal functions have more than one local optima. So it is worth to point out that multi-modal functions are adequate for evaluating the exploration ability of an algorithm. It can be found that, the diversification of ABCDC is excellent in this comparison. For functions $f_{11} - f_{16}$ and f_{21}, ABCDC achieves the corresponding global optimum. For f_{17}, f_{18}, f_{20} and f_{22}, the basic ABC obtains the optimal solutions, however, the results of ABCDC are competitive.

In addition, according to the results of the Wilcoxon tests, the advantage of ABCDC is remarkable compared to DEABC. Meanwhile, ABCDC performs better than the original ABC on 10 functions, especially on $f_2, f_6, f_{13} - f_{15}$ and f_{21}. Compared with the NSABC algorithm, ABCDC performs better than it on 17 out of 22 problems. As for APABC, there are 7 functions where their corresponding errors are similar and 15 functions where ABCDC has significant advantages. Moreover, the proposed algorithm performs as good as ILTD_ABC algorithm on 11 functions.

Besides, it is essential to consider the overall performances of 22 benchmark functions together. In this context, it is worth pointing out that ABCDC is outstanding in term of solution precision for various kinds of problems. Compared to basic ABC and ILTD_ABC, ABCDC has very small errors when the other two algorithms get better solutions. Nevertheless, when the solutions of ABCDC are better, their errors are relatively large. So in Figure 2.3, ABCDC is in the first place under Friedman test. We can conclude that the proposed algorithm achieves excellent results considering solution quality in the case of low-dimension.

The comparison results for middle dimensional problems are shown in Table 2.5 and similar phenomena can be observed. For uni-modal functions f_4 and f_9, ABCDC, ILTD_ABC and original ABC manage to find out the optimal solutions. For f_6, ABCDC and ILTD_ABC attain the global optimum whereas other competitors are far from the optimal solution. And for $f_1 - f_3$ and f_5, ABCDC

Table 2.5: Comparison between ABCDC and other ABC variants with $D = 50$

Function		ABC		DEABC		APABC		ILTD_ABC		NSABC		ABCDC
f_1	Mean	0.00E+00	-	1.85E-15	+	3.22E-68	+	0.00E+00	-	1.23E-23	+	4.10E-241
	Std	0.00E+00		2.09E-15		5.97E-68		0.00E+00		7.88E-24		0.00E+00
f_2	Mean	2.90E-08	+	2.63E-12	+	2.32E-63	+	0.00E+00	-	2.77E-20	+	4.43E-211
	Std	1.95E-08		2.86E-12		6.70E-63		0.00E+00		2.12E-20		0.00E+00
f_3	Mean	0.00E+00	-	3.07E-16	+	1.35E-67	+	0.00E+00	-	2.84E-24	+	1.95E-246
	Std	0.00E+00		3.34E-16		4.44E-67		0.00E+00		1.99E-24		0.00E+00
f_4	Mean	0.00E+00	=	1.20E-21	+	6.45E-74	+	0.00E+00	=	2.79E-77	+	0.00E+00
	Std	0.00E+00		5.97E-21		2.40E-73		0.00E+00		1.39E-76		0.00E+00
f_5	Mean	0.00E+00	-	7.76E+00	+	6.56E-36	+	0.00E+00	-	5.06E-13	+	4.53E-135
	Std	0.00E+00		2.29E+01		1.21E-35		0.00E+00		2.07E-13		2.04E-134
f_6	Mean	2.95E+01	+	1.02E+01	+	3.89E+00	+	0.00E+00	=	2.44E+01	+	0.00E+00
	Std	2.83E+00		5.92E+00		3.90E-01		0.00E+00		2.56E+00		0.00E+00
f_7	Mean	0.00E+00	=	0.00E+00	=	0.00E+00	=	0.00E+00	=	0.00E+00	=	0.00E+00
	Std	0.00E+00		0.00E+00		0.00E+00		0.00E+00		0.00E+00		0.00E+00
f_8	Mean	0.00E+00	-	2.67E-109	+	2.67E-109	=	2.67E-109	=	0.00E+00	-	2.67E-109
	Std	0.00E+00		5.15E-123		9.65E-125		9.65E-125		0.00E+00		9.65E-125
f_9	Mean	0.00E+00	=	1.69E-26	+	4.74E-135	+	0.00E+00	=	4.71E-52	+	0.00E+00
	Std	0.00E+00		4.79E-26		1.24E-134		0.00E+00		7.71E-52		0.00E+00
f_{10}	Mean	1.12E-01	+	1.25E-01	+	3.52E-02	+	3.60E-06	-	7.52E-02	+	1.31E-05
	Std	1.99E-02		2.28E-02		6.53E-03		2.90E-06		1.52E-02		9.41E-06
f_{11}	Mean	1.66E-01	+	4.75E+01	+	4.77E-01	+	4.62E+01	+	1.28E+01	+	0.00E+00
	Std	1.28E-01		1.88E+01		3.82E-01		1.36E-01		2.63E+01		0.00E+00
f_{12}	Mean	3.24E-04	+	4.38E+02	+	0.00E+00	=	0.00E+00	=	3.98E-02	+	0.00E+00
	Std	1.62E-03		1.49E+01		0.00E+00		0.00E+00		1.99E-01		0.00E+00
f_{13}	Mean	8.00E-02	+	4.17E+02	+	0.00E+00	=	0.00E+00	=	0.00E+00	=	0.00E+00
	Std	2.77E-01		1.77E+01		0.00E+00		0.00E+00		0.00E+00		0.00E+00
f_{14}	Mean	0.00E+00	=	6.90E-04	+	1.06E-11	+	0.00E+00	=	0.00E+00	=	0.00E+00
	Std	0.00E+00		2.41E-03		5.30E-11		0.00E+00		0.00E+00		0.00E+00
f_{15}	Mean	2.33E+02	+	5.56E+02	+	9.48E+00	+	4.26E+02	+	3.53E+47	+	1.82E-11
	Std	9.35E+01		1.55E+03		3.28E+01		2.42E+02		1.45E+48		0.00E+00
f_{16}	Mean	1.35E-03	+	2.73E-08	+	1.46E-01	+	8.88E-16	=	9.14E-11	+	8.88E-16
	Std	1.80E-03		2.51E-08		3.56E-01		0.00E+00		2.26E-10		0.00E+00
f_{17}	Mean	0.00E+00	-	5.65E+00	+	9.42E-33	=	1.15E-32	+	1.49E-25	+	9.42E-33
	Std	0.00E+00		1.54E+00		1.40E-48		1.05E-32		7.53E-26		1.40E-48
f_{18}	Mean	0.00E+00	-	5.80E+00	+	9.42E-33	=	1.17E-32	+	1.50E-25	+	9.42E-33
	Std	0.00E+00		1.89E+00		1.40E-48		1.05E-32		1.39E-25		1.40E-48
f_{19}	Mean	1.20E-04	+	7.69E-04	+	8.33E-32	+	0.00E+00	-	5.41E-13	+	2.60E-117
	Std	9.04E-05		2.08E-03		3.68E-31		0.00E+00		3.12E-13		1.29E-116
f_{20}	Mean	0.00E+00	-	2.03E+01	+	6.61E-30	-	6.18E+01	+	1.08E-21	-	8.06E-11
	Std	0.00E+00		1.18E+00		1.43E-45		2.50E+01		1.18E-21		4.02E-10
f_{21}	Mean	1.98E-06	+	8.42E-04	+	5.68E-15	+	0.00E+00	=	5.97E-14	+	0.00E+00
	Std	1.30E-06		8.32E-04		8.20E-15		0.00E+00		2.36E-14		0.00E+00
f_{22}	Mean	0.00E+00	-	7.46E-01	+	2.86E-05	+	1.89E-04	+	2.86E-05	+	2.86E-05
	Std	0.00E+00		5.34E-01		2.58E-14		4.67E-05		1.57E-14		0.00E+00
Total	+/=/-	10/4/8		21/1/0		15/6/1		6/10/6		17/3/2		

is competitive with ILTD_ABC and basic ABC. Meanwhile, the advantage of ABCDC is significant on most muti-modal problems.

According to the results of the Wilcoxon tests, there are 10 out of 22 functions that ABCDC performs better than the original ABC. And they have similar results on 4 functions. At the same time, ABCDC obtains better solutions in 21 problems compared to DEABC. Meanwhile, ABCDC performs better than the APABC on 15 functions. Compared with the ILTD_ABC algorithm, the numbers that ABCDC performs better than it and worse than it are both 6. There are 10 out of 22 functions that they have similar results. As for NSABC, there are 17 functions where ABCDC has significant advantages.

In Table 2.6, the proposed ABCDC, basic ABC and ILTD_ABC are competitive with each other in solving uni-modal functions. For $f_1 - f_3$, the ILTD_ABC algorithm attains the global optima. For f_4 ABCDC and the basic ABC achieve the optimum while ABCDC and ILTD_ABC find the optimum of f_6. Notice that, on the functions where ABCDC doesn't attain the global optimum, the corresponding errors are much smaller than those of other competitors. At the same time, the advantages of ABCDC, basic ABC and ILTD_ABC algorithms are also remarkable on multi-modal functions. For $f_{12} - f_{14}$, ABCDC and ILTD_ABC attain the global optima. For f_{15} and f_{16}, all competitors fail to reach the optimum, and the ABCDC algorithm generates the best solutions. The original ABC perform the best on f_{17}, f_{18} and f_{22} whereas ABCDC attain the best results for f_{20} and f_{21}.

Considering the Wilcoxon test results, ABCDC performs better than the original ABC on 11 functions. DEABC fails to surpass the proposed algorithm on all the benchmarks. ABCDC obtains smaller errors than APABC does on 15 out of 22 functions. Meanwhile, ABCDC is competitive to ILTD_ABC as their results do not have significant differences on 10 functions. And these two methods each achieves better results on 6 functions.

In addition, Friedman tests are conducted on all the three comparisons as it is widely used to evaluate the overall performance of more than two algorithms. Figure 2.3 illustrates the average rankings of involved ABC algorithms based on the mean values for each dimensions case.

It can be seen that ABCDC obtains the best rankings in all the three cases.

Table 2.6: Comparison between ABCDC and other ABC variants with $D = 100$

Function		ABC		DEABC		APABC		ILTD_ABC		NSABC		ABCDC
f_1	Mean	0.00E+00	-	2.04E-13	+	2.45E-63	+	0.00E+00	-	1.04E-21	+	5.46E-67
	Std	0.00E+00		5.67E-13		4.78E-63		0.00E+00		4.43E-22		2.73E-66
f_2	Mean	1.47E-07	+	1.19E-10	+	4.00E-59	+	0.00E+00	-	2.72E-18	+	2.56E-251
	Std	8.09E-08		1.69E-10		1.22E-58		0.00E+00		1.80E-18		0.00E+00
f_3	Mean	0.00E+00	-	1.65E-14	+	3.35E-63	+	0.00E+00	-	4.99E-22	+	9.31E-170
	Std	0.00E+00		2.19E-14		9.17E-63		0.00E+00		3.70E-22		0.00E+00
f_4	Mean	0.00E+00	=	3.16E-15	+	1.26E-74	+	0.00E+00	=	3.92E-77	+	0.00E+00
	Std	0.00E+00		1.54E-14		6.27E-74		0.00E+00		1.93E-76		0.00E+00
f_5	Mean	5.55E-09	+	1.54E+02	+	1.62E-33	+	0.00E+00	-	7.52E-12	+	1.33E-106
	Std	5.94E-09		6.58E+01		2.01E-33		0.00E+00		1.77E-12		6.67E-106
f_6	Mean	5.28E+01	+	7.53E+01	+	1.84E+01	+	0.00E+00	=	5.00E+01	+	0.00E+00
	Std	2.85E+00		3.97E+00		1.22E+00		0.00E+00		2.22E+00		0.00E+00
f_7	Mean	0.00E+00	=	2.00E-01	+	0.00E+00	=	0.00E+00	=	0.00E+00	=	0.00E+00
	Std	0.00E+00		4.08E-01		0.00E+00		0.00E+00		0.00E+00		0.00E+00
f_8	Mean	0.00E+00	-	7.12E-218	+	7.12E-218	=	7.12E-218	=	0.00E+00	-	7.12E-218
	Std	0.00E+00		0.00E+00		0.00E+00		0.00E+00		0.00E+00		0.00E+00
f_9	Mean	0.00E+00	=	7.43E-20	+	3.46E-122	+	0.00E+00	=	3.85E-48	+	0.00E+00
	Std	0.00E+00		1.83E-19		7.46E-122		0.00E+00		9.00E-48		0.00E+00
f_{10}	Mean	2.70E-01	+	2.71E-01	+	7.57E-02	+	2.84E-06	-	2.19E-01	+	6.61E-06
	Std	3.35E-02		3.47E-02		1.01E-02		2.25E-06		2.29E-02		4.17E-06
f_{11}	Mean	2.81E-01	-	1.61E+02	+	1.17E+00	-	9.56E+01	+	1.84E+01	+	3.92E+00
	Std	1.59E-01		4.47E+01		1.28E+00		1.53E-01		2.57E+01		1.96E+01
f_{12}	Mean	4.88E-02	+	9.76E+02	+	0.00E+00	=	0.00E+00	=	2.39E-01	+	0.00E+00
	Std	2.02E-01		2.35E+01		0.00E+00		0.00E+00		4.34E-01		0.00E+00
f_{13}	Mean	8.05E-01	+	9.67E+02	+	0.00E+00	=	0.00E+00	=	0.00E+00	=	0.00E+00
	Std	6.82E-01		3.22E+01		0.00E+00		0.00E+00		0.00E+00		0.00E+00
f_{14}	Mean	0.00E+00	=	7.88E-04	+	2.12E-13	+	0.00E+00	=	0.00E+00	=	0.00E+00
	Std	0.00E+00		2.82E-03		6.60E-13		0.00E+00		0.00E+00		0.00E+00
f_{15}	Mean	7.15E+02	+	1.16E+04	+	4.26E+01	+	1.14E+03	+	5.18E+51	+	1.09E-10
	Std	1.85E+02		1.03E+04		6.73E+01		3.33E+02		2.59E+52		0.00E+00
f_{16}	Mean	1.54E-03	+	1.99E-01	+	3.51E-01	+	8.88E-16	=	7.19E-10	+	8.88E-16
	Std	1.01E-03		4.08E-01		3.98E-01		0.00E+00		6.31E-10		0.00E+00
f_{17}	Mean	0.00E+00	-	2.83E+03	+	4.71E-33	=	1.06E-31	+	5.61E-24	+	4.71E-33
	Std	0.00E+00		2.94E+03		6.98E-49		4.97E-31		2.24E-24		6.98E-49
f_{18}	Mean	0.00E+00	-	2.29E+03	+	4.71E-33	=	1.14E-31	+	5.82E-24	+	4.71E-33
	Std	0.00E+00		2.75E+03		6.98E-49		4.97E-31		2.66E-24		6.98E-49
f_{19}	Mean	3.85E-03	+	1.24E-07	+	1.52E-21	+	0.00E+00	-	3.21E-06	+	2.95E-125
	Std	4.96E-03		1.67E-07		7.60E-21		0.00E+00		1.60E-05		1.35E-124
f_{20}	Mean	5.87E-10	+	3.29E+01	+	8.79E-03	+	1.56E+02	+	4.39E-03	+	1.04E-15
	Std	2.94E-09		5.18E+00		3.04E-02		4.52E+01		2.20E-02		3.54E-15
f_{21}	Mean	1.07E-05	+	7.54E-01	+	6.59E-14	+	0.00E+00	=	2.25E-05	+	0.00E+00
	Std	3.54E-06		7.80E-01		2.28E-14		0.00E+00		1.13E-04		0.00E+00
f_{22}	Mean	0.00E+00	-	2.26E+00	+	2.86E-05	+	4.58E-04	+	2.86E-05	+	2.86E-05
	Std	0.00E+00		6.78E-01		2.29E-14		1.02E-04		1.72E-14		0.00E+00
Total	+/=/=	11/4/7		22/0/0		15/6/1		6/10/6		18/3/1		

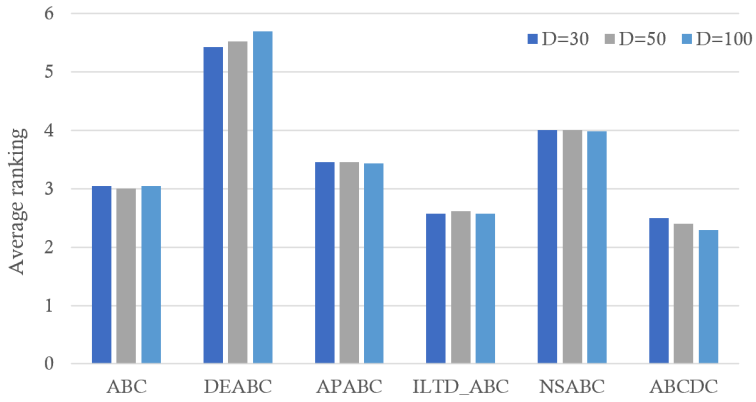


Fig. 2.3. Average rankings of ABC algorithms by Friedman test with $D = 30$, 50, and 100

And ILTD_ABC is the second-best algorithm followed by the original ABC algorithm. Therefore, the proposed algorithm outperforms the other ABC algorithms.

In addition to comparing with other ABC variants, we also observe and compare the results of ABCDC itself in different dimensions. It can be found that the results of $D = 30$, 50, and 100 calculated by ABCDC are similar. That is to say, the proposed algorithm is not sensitive to the increase of dimensions which means that it possesses superior robustness at least in solving the chosen benchmark functions.

2.3.4 Comparison with non-ABC algorithms

In this subsection, ABCDC is compared to four effective non-ABC meta-heuristic algorithms. The DE with mutation "DE/rand/1" is first concerned in the comparison. PSO and CS are also involved as they are also effective and famous. Last, the firefly algorithm (FA) (Yang, 2009) is also an effective method inspired by the behavior of the firefly.

The tests are carried out on those benchmarks in low, middle and high dimensions as well. The statical results of are presented in Table 2.7 - Table 2.9. In addition, results of Wilcoxon rank sum test are shown. And Figure 2.4 presents the average rankings given by Friedman tests.

In Table 2.7, the results of uni-modal functions $f_1 - f_9$ show that ABCDC contains significant advantages compared to other methods. Moreover, ABCDC is able to find the global optimum of f_2, f_4 f_5, f_7 and f_9. Meanwhile, DE and

Table 2.7: Comparison between ABCDC and other meta-heuristic algorithms with $D = 30$

Function		PSO		DE		CS		FA		ABCDC
f_1	Mean	1.33E-18	+	7.77E-39	+	6.65E-05	+	1.72E-09	+	1.84E-197
	Std	3.75E-18		9.06E-39		3.62E-05		3.46E-10		0.00E+00
f_2	Mean	5.02E-16	+	4.85E-36	+	2.74E-01	+	9.83E-05	+	4.22E-145
	Std	1.01E-15		7.98E-36		1.18E-01		2.51E-05		2.11E-144
f_3	Mean	8.93E-20	+	5.16E-40	+	7.84E-06	+	2.22E-10	+	2.91E-188
	Std	1.60E-19		5.32E-40		2.81E-06		3.94E-11		0.00E+00
f_4	Mean	6.27E-40	+	6.10E-82	+	4.53E-24	+	2.48E-17	+	0.00E+00
	Std	2.47E-39		3.05E-81		1.15E-23		2.65E-17		0.00E+00
f_5	Mean	2.80E-13	+	2.76E-21	+	1.09E-01	+	1.44E-05	+	1.40E-103
	Std	4.60E-13		1.98E-21		4.39E-02		1.07E-06		5.89E-103
f_6	Mean	2.81E+00	+	2.32E-01	+	6.84E-01	+	1.05E+01	+	1.12E-299
	Std	1.23E+00		7.18E-01		3.09E-01		4.39E+00		0.00E+00
f_7	Mean	4.00E-02	+	0.00E+00	=	0.00E+00	=	8.00E-02	+	0.00E+00
	Std	2.00E-01		0.00E+00		0.00E+00		2.77E-01		0.00E+00
f_8	Mean	7.86E-59	+	7.18E-66	=	7.18E-66	+	7.18E-66	+	7.18E-66
	Std	2.13E-58		3.23E-81		3.45E-70		1.23E-72		3.23E-81
f_9	Mean	4.41E-28	+	1.38E-62	+	2.70E-14	+	3.21E-20	+	0.00E+00
	Std	1.13E-27		3.13E-62		2.23E-14		4.26E-20		0.00E+00
f_{10}	Mean	1.99E-02	+	6.24E-03	+	2.32E-02	+	6.50E-03	+	1.99E-05
	Std	7.71E-03		1.92E-03		6.74E-03		1.85E-03		1.36E-05
f_{11}	Mean	3.50E+01	+	1.91E+01	+	2.31E+01	+	4.63E+01	+	0.00E+00
	Std	2.43E+01		1.02E+00		4.09E+00		2.83E+01		0.00E+00
f_{12}	Mean	3.50E+01	+	1.41E+02	+	7.10E+01	+	4.99E+01	+	0.00E+00
	Std	1.22E+01		1.49E+01		1.14E+01		1.74E+01		0.00E+00
f_{13}	Mean	3.72E+01	+	1.17E+02	+	6.24E+01	+	6.21E+01	+	0.00E+00
	Std	1.31E+01		1.01E+01		1.27E+01		2.38E+01		0.00E+00
f_{14}	Mean	1.89E-02	+	7.89E-04	+	4.14E-03	+	6.01E-03	+	0.00E+00
	Std	1.85E-02		2.82E-03		3.79E-03		8.16E-03		0.00E+00
f_{15}	Mean	1.21E+03	+	5.41E+03	+	3.58E+03	+	3.36E+03	+	0.00E+00
	Std	3.48E+02		4.15E+02		2.64E+02		5.86E+02		0.00E+00
f_{16}	Mean	1.12E+01	+	2.00E+01	+	1.86E+01	+	1.54E-05	+	8.88E-16
	Std	1.01E+01		7.87E-03		3.42E+00		2.11E-06		0.00E+00
f_{17}	Mean	4.15E-03	+	4.15E-03	+	3.56E-03	+	5.81E-02	+	1.57E-32
	Std	2.07E-02		2.07E-02		5.89E-03		1.16E-01		5.59E-48
f_{18}	Mean	4.15E-03	+	1.57E-32	=	1.66E-03	+	4.56E-02	+	1.57E-32
	Std	2.07E-02		5.59E-48		2.11E-03		9.02E-02		5.59E-48
f_{19}	Mean	2.73E-12	+	1.67E-02	+	5.45E+00	+	7.31E-06	+	2.38E-92
	Std	3.60E-12		4.42E-03		1.85E+00		4.88E-06		1.19E-91
f_{20}	Mean	1.78E+00	+	5.32E+00	+	6.51E+00	+	2.56E+00	+	4.96E-11
	Std	5.16E-01		6.51E-01		8.29E-01		7.63E-01		2.13E-10
f_{21}	Mean	6.52E-02	+	0.00E+00	=	2.99E+00	+	2.73E-03	+	0.00E+00
	Std	3.06E-01		0.00E+00		7.47E-01		1.51E-04		0.00E+00
f_{22}	Mean	5.24E+00	+	7.54E-02	+	5.77E+00	+	6.97E+00	+	2.86E-05
	Std	1.87E+00		2.61E-01		1.17E+00		3.02E+00		0.00E+00
Total	+/=/-	22/0/0		18/4/0		21/1/0		22/0/0		

CS also find the global optimal solution of f_7. As mentioned above, the uni-modal problem can detect the exploitation ability, so the exploitation capability of proposed ABCDC algorithm is outstanding among the concerned algorithms. As for the multi-modal functions $f_{11} - f_{22}$, ABCDC outperforms the other non-ABC methods. DE algorithm obtains the competitive results as ABCDC does on f_{18} and f_{21}. Regarding to the Wilcoxon test, the PSO and FA algorithms fail to surpass ABCDC on all the benchmark functions. Meanwhile, solutions found by DE are comparable to ABCDC on 4 functions. And ABCDC performs better than DE on 18 functions. As for CS, there is only one problem where it performs as good as the proposed algorithm. Thus, the proposed algorithm has excellent performance for low-dimensional problems considering the solution accuracy.

Similar conclusions can be derived from the comparison results of $D = 50$ and 100. Table 2.8 presents the results of 50-dimensional problems, for solving the uni-modal problems, ABCDC achieves the exact optimal solutions of f_3, f_6 and f_9 whereas PSO and CS only manage to reach the optimum of f_7. Moreover, the superiority of ABCDC is obvious on f_1 to f_6. As for multi-modal functions, ABCDC manages to obtain the optima of five functions. Considering the results of Wilcoxon test, the mean values of ABCDC surpass all the compared algorithms on most problems. It can be concluded that ABCDC achieves the best results in the comparison with $D = 50$.

In Table 2.9, ABCDC attains the best results on all the benchmark functions whereas DE algorithm obtains similar solutions on f_8 and f_{14}. Note that, the PSO, DE and CS algorithms have competitive results on uni-modal functions. Nevertheless, their solution qualities are not comparable to that of ABCDC when solving multi-modal functions. Therefore, the proposed algorithm has excellent diversification and intensification abilities compared to non-ABC algorithms.

Furthermore, Figure 2.4 presents the results of Friedman tests. The advantages of ABCDC is evident in this figure and it is followed by the DE algorithm. It can be concluded that ABCDC is outstanding in all the comparisons with other meta-heuristic algorithms.

Table 2.8: Comparison between ABCDC and other meta-heuristic algorithms with $D = 50$

Function		PSO		DE		CS		FA		ABCDC
f_1	Mean	2.12E-14	+	3.07E-34	+	5.90E-05	+	4.55E-09	+	4.10E-241
	Std	3.61E-14		2.59E-34		2.70E-05		2.55E-09		0.00E+00
f_2	Mean	1.79E-11	+	2.12E-31	+	3.84E-01	+	6.06E-06	+	4.43E-211
	Std	4.55E-11		2.49E-31		1.79E-01		1.00E-05		0.00E+00
f_3	Mean	2.15E-15	+	6.01E-35	+	1.28E-05	+	9.18E-10	+	1.95E-246
	Std	2.86E-15		8.96E-35		5.25E-06		7.49E-10		0.00E+00
f_4	Mean	1.37E-30	+	3.50E-40	+	5.44E-24	+	3.36E-23	+	0.00E+00
	Std	4.37E-30		1.75E-39		2.67E-23		1.96E-23		0.00E+00
f_5	Mean	1.47E-10	+	1.67E-18	+	2.40E+09	+	1.19E-06	+	4.53E-135
	Std	2.88E-10		1.65E-18		4.36E+09		3.91E-07		2.04E-134
f_6	Mean	1.98E+01	+	6.42E+00	+	1.81E+00	+	3.94E+01	+	0.00E+00
	Std	3.00E+00		5.71E+00		7.43E-01		6.94E+00		0.00E+00
f_7	Mean	8.80E-01	+	0.00E+00	=	0.00E+00	=	2.72E+00	+	0.00E+00
	Std	9.27E-01		0.00E+00		0.00E+00		3.41E+00		0.00E+00
f_8	Mean	7.35E-94	+	2.67E-109	=	2.68E-109	+	2.67E-109	+	2.67E-109
	Std	3.57E-93		9.65E-125		1.23E-111		2.87E-119		9.65E-125
f_9	Mean	8.44E-20	+	4.51E-53	+	1.68E-12	+	2.66E-16	+	0.00E+00
	Std	1.84E-19		1.19E-52		2.55E-12		5.50E-16		0.00E+00
f_{10}	Mean	5.71E-02	+	1.18E-02	+	3.32E-02	+	1.77E-02	+	1.31E-05
	Std	1.74E-02		2.46E-03		7.91E-03		5.23E-03		9.41E-06
f_{11}	Mean	8.43E+01	+	3.98E+01	+	5.88E+01	+	1.01E+02	+	0.00E+00
	Std	3.52E+01		1.05E+01		1.82E+01		3.15E+01		0.00E+00
f_{12}	Mean	9.04E+01	+	3.05E+02	+	1.38E+02	+	1.19E+02	+	0.00E+00
	Std	2.09E+01		1.94E+01		1.58E+01		3.09E+01		0.00E+00
f_{13}	Mean	1.08E+02	+	2.72E+02	+	1.36E+02	+	1.53E+02	+	0.00E+00
	Std	2.84E+01		1.86E+01		2.57E+01		2.98E+01		0.00E+00
f_{14}	Mean	1.11E-02	+	0.00E+00	=	1.32E-03	+	5.61E-03	+	0.00E+00
	Std	1.38E-02		0.00E+00		3.72E-03		8.17E-03		0.00E+00
f_{15}	Mean	2.28E+03	+	1.11E+04	+	6.90E+03	+	6.35E+03	+	1.82E-11
	Std	3.77E+02		6.57E+02		4.67E+02		6.73E+02		0.00E+00
f_{16}	Mean	1.83E+01	+	2.00E+01	+	1.94E+01	+	3.20E+00	+	8.88E-16
	Std	5.50E+00		1.68E-03		9.15E-01		7.48E+00		0.00E+00
f_{17}	Mean	2.99E-02	+	3.17E-32	+	6.45E+08	+	8.21E-02	+	9.42E-33
	Std	5.42E-02		1.09E-31		2.18E+09		8.93E-02		1.40E-48
f_{18}	Mean	3.49E-02	+	4.98E-03	+	8.60E+08	+	1.25E-01	+	9.42E-33
	Std	5.40E-02		1.72E-02		2.76E+09		1.66E-01		1.40E-48
f_{19}	Mean	1.17E-09	+	3.15E-02	+	1.30E+01	+	4.56E-07	+	2.60E-117
	Std	2.11E-09		7.68E-03		2.84E+00		1.58E-07		1.29E-116
f_{20}	Mean	4.75E+00	+	1.19E+01	+	1.67E+01	+	7.85E+00	+	8.06E-11
	Std	1.20E+00		1.95E+00		2.10E+00		2.44E+00		4.02E-10
f_{21}	Mean	1.73E+00	+	0.00E+00	=	3.08E+00	+	1.18E-02	+	0.00E+00
	Std	1.77E+00		0.00E+00		7.53E-01		2.46E-02		0.00E+00
f_{22}	Mean	7.49E+00	+	2.04E-01	+	6.78E+00	+	8.89E+00	+	2.86E-05
	Std	1.61E+00		3.22E-01		9.49E-01		1.74E+00		0.00E+00
Total	+/=/-	22/0/0		18/4/0		21/1/0		22/0/0		

Table 2.9: Comparison between ABCDC and other meta-heuristic algorithms with $D = 100$

Function		PSO		DE		CS		FA		ABCDC
f_1	Mean	2.39E-08	+	2.84E-32	+	1.27E-04	+	2.05E-06	+	5.46E-67
	Std	4.98E-08		3.45E-32		5.51E-05		1.23E-06		2.73E-66
f_2	Mean	5.94E-06	+	3.68E-29	+	3.24E+00	+	1.90E+04	+	2.56E-251
	Std	1.22E-05		4.55E-29		2.48E+00		4.35E+04		0.00E+00
f_3	Mean	5.08E-09	+	1.94E-32	+	5.21E-05	+	1.87E-06	+	9.31E-170
	Std	7.03E-09		2.01E-32		3.24E-05		4.89E-06		0.00E+00
f_4	Mean	1.59E-20	+	1.03E-13	+	7.89E-26	+	2.84E-22	+	0.00E+00
	Std	6.03E-20		5.15E-13		1.79E-25		8.14E-22		0.00E+00
f_5	Mean	5.85E-05	+	3.05E-17	+	1.00E+10	+	7.90E-01	+	1.33E-106
	Std	1.99E-04		2.84E-17		0.00E+00		2.73E+00		6.67E-106
f_6	Mean	5.54E+01	+	9.22E+01	+	5.31E+00	+	8.45E+01	+	0.00E+00
	Std	3.33E+00		1.36E+01		1.65E+00		5.19E+00		0.00E+00
f_7	Mean	1.36E+01	+	1.20E-01	+	1.96E+00	+	3.91E+01	+	0.00E+00
	Std	8.45E+00		3.32E-01		2.61E+00		2.34E+01		0.00E+00
f_8	Mean	1.79E-182	+	7.12E-218	=	7.48E-218	+	7.12E-218	+	7.12E-218
	Std	0.00E+00		0.00E+00		0.00E+00		0.00E+00		0.00E+00
f_9	Mean	1.01E-11	+	1.07E-45	+	5.43E-10	+	5.87E-11	+	0.00E+00
	Std	1.46E-11		2.10E-45		4.95E-10		4.93E-11		0.00E+00
f_{10}	Mean	2.45E-01	+	2.15E-02	+	9.51E-02	+	1.17E-01	+	6.61E-06
	Std	5.34E-02		4.23E-03		2.28E-02		9.13E-02		4.17E-06
f_{11}	Mean	1.88E+02	+	9.52E+01	+	9.94E+01	+	2.63E+02	+	3.92E+00
	Std	4.38E+01		1.66E+01		5.76E+01		2.03E+02		1.96E+01
f_{12}	Mean	2.96E+02	+	7.21E+02	+	3.41E+02	+	3.38E+02	+	0.00E+00
	Std	3.68E+01		4.05E+01		3.42E+01		6.26E+01		0.00E+00
f_{13}	Mean	3.92E+02	+	7.07E+02	+	3.51E+02	+	4.54E+02	+	0.00E+00
	Std	5.26E+01		4.88E+01		5.68E+01		6.02E+01		0.00E+00
f_{14}	Mean	1.47E-02	+	0.00E+00	=	1.87E-03	+	3.94E-03	+	0.00E+00
	Std	2.66E-02		0.00E+00		4.84E-03		8.18E-03		0.00E+00
f_{15}	Mean	5.47E+03	+	2.60E+04	+	1.64E+04	+	1.44E+04	+	1.09E-10
	Std	7.26E+02		8.52E+02		6.41E+02		1.17E+03		0.00E+00
f_{16}	Mean	1.99E+01	+	2.00E+01	+	1.85E+01	+	1.47E+01	+	8.88E-16
	Std	5.82E-02		1.25E-02		3.77E+00		8.09E+00		0.00E+00
f_{17}	Mean	4.24E-01	+	4.98E-03	+	1.00E+10	+	1.07E-01	+	4.71E-33
	Std	8.03E-01		2.49E-02		0.00E+00		1.00E-01		6.98E-49
f_{18}	Mean	2.40E-01	+	3.15E-02	+	1.00E+10	+	8.98E-02	+	4.71E-33
	Std	2.96E-01		8.39E-02		0.00E+00		1.03E-01		6.98E-49
f_{19}	Mean	4.51E-06	+	1.22E-05	+	2.49E+01	+	1.48E-05	+	2.95E-125
	Std	1.34E-05		4.55E-05		4.43E+00		5.04E-05		1.35E-124
f_{20}	Mean	1.41E+01	+	1.88E+01	+	5.52E+01	+	4.96E+01	+	1.04E-15
	Std	2.96E+00		7.67E+00		6.27E+00		1.24E+01		3.54E-15
f_{21}	Mean	1.15E+01	+	6.00E-02	+	6.46E+00	+	7.55E+00	+	0.00E+00
	Std	4.25E+00		3.00E-01		1.35E+00		3.20E+00		0.00E+00
f_{22}	Mean	9.84E+00	+	1.52E+00	+	8.04E+00	+	1.01E+01	+	2.86E-05
	Std	1.30E+00		6.21E-01		9.65E-01		1.50E+00		0.00E+00
Total	+/=/-	22/0/0		20/2/0		22/0/0		22/0/0		

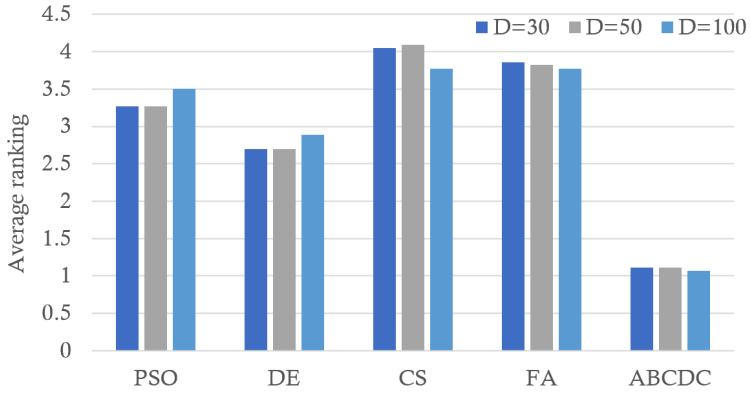


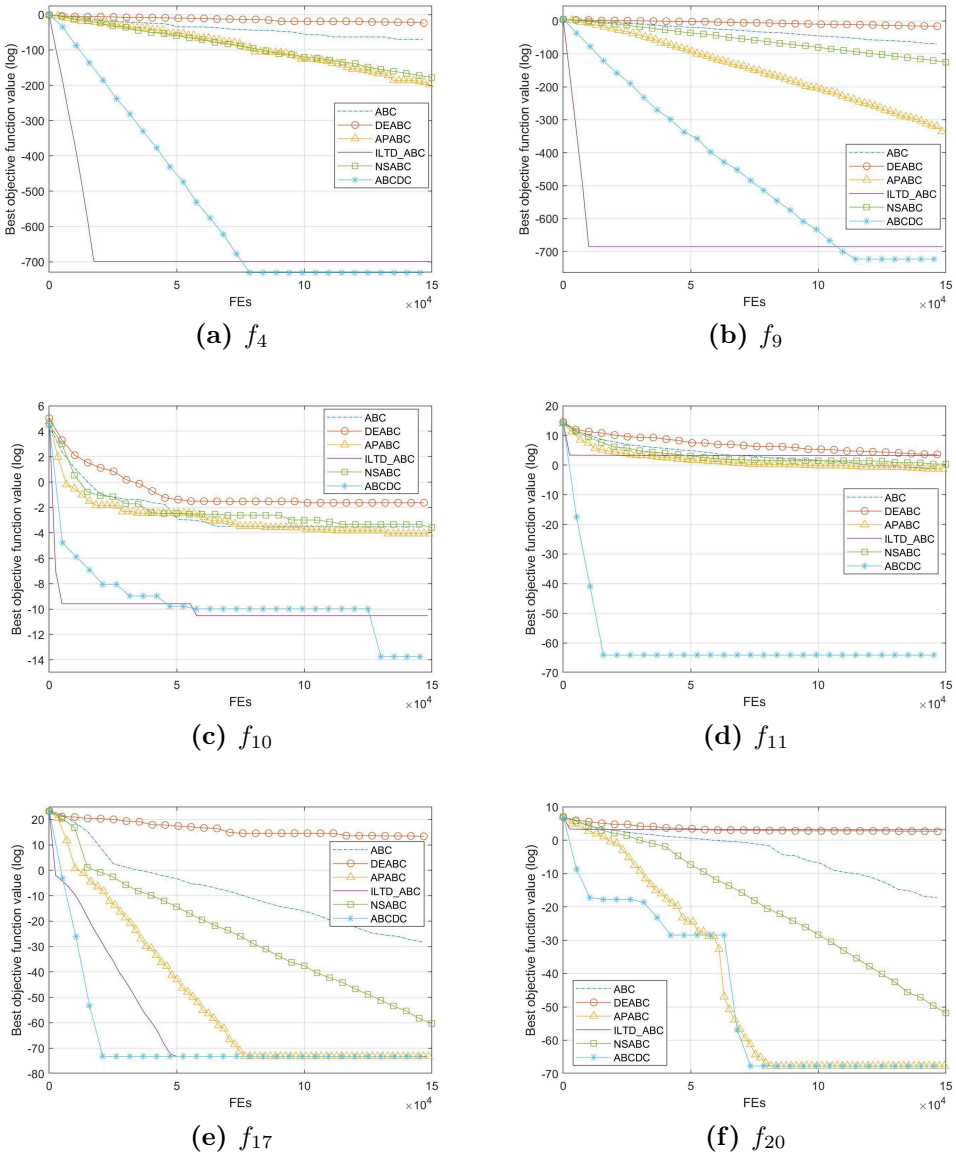
Fig. 2.4. Average rankings of non-ABC algorithms and ABCDC by Friedman test with $D = 30$, 50, and 100

2.3.5 Convergence behavior analysis

The convergence performances of proposed algorithm are analyzed in this subsection. Convergence graphs of compared ABC algorithms as well as the execution time of all the involved methods are presented in the follows.

In fact, convergence curves can vividly show how fast the objective function value decreases along with the increase of function evaluations. And, the convergence processes of compared ABC algorithms to solve six representative benchmark functions are plotted in Figure 2.5. f_4, f_9 and f_{10} are uni-modal while the other three problems are multi-modal.

For uni-modal functions, it is obvious that ABCDC and ILTD_ABC converge much faster than other ABC algorithms, including the original ABC. In addition, the proposed algorithm achieves more accurate results than ILTD_ABC does. For multi-modal functions, the advantages of proposed algorithm is significant which indicates its outstanding exploration ability. Therefore, the convergence curves demonstrate that proposed ABCDC enhances the convergence speed of ABC algorithm effectively.



(a) f_4 **(b)** f_9

(c) f_{10} **(d)** f_{11}

(e) f_{17} **(f)** f_{20}

Fig. 2.5. The convergence performance of ABCDC and compared ABC algorithms with D=30

2.4 Conclusion

With the purpose of enhancing the performance of ABC algorithm, an improved ABC with dynamic population composition (ABCDC) is proposed in this chapter. Firstly, the SLHD is adopted in the initialization phase to ensure the diversity of initial population which can help with the convergence rate. Secondly, we divide the missions of exploration and exploitation more clearly, and distribute them to employed bees and onlooker bees respectively. Accordingly, two DE-inspired solution search strategies are utilized to reinforce the ability of employed bees and onlooker bees. Then, the balance between exploration and exploitation can be improved because the solution search equation used in employed bee phase is good at exploring while the one used in onlooker bee phase shows the strength of exploiting. In addition, a method for adjusting the population's composition is proposed. In order to help the employed bees to explore, its size is set to be very large in the beginning. And as the iteration increases, some promising regions appear, the size of onlooker bees augments gradually. According to the experimental results, ABCDC outperforms the other competitors in terms of solution accuracy.

2. IMPROVED ABC ALGORITHM WITH DYNAMIC POPULATION COMPOSITION (ABCDC)

Chapter 3

Reinforcement Learning based ABC algorithm (ABC_RL)

Contents

3.1 Introduction

As mentioned in the previous chapters, among many possible reasons that limit the performance of ABC, the ineffectiveness of its search equation has been mentioned the most frequently. Due to the fact that only one dimension can be updated at a time, it has been discovered that the solution search equation limits the search efficiency (Akay & Karaboga, 2012; Zhou *et al.*, 2021a). In this case, the evolutionary operators of the DE algorithm have been widely concerned because they are able to update multiple dimensions at a time. Meanwhile, the mutation operators can make individuals learn more information from their neighbors (Chen *et al.*, 2019b; Cui *et al.*, 2020, 2022; Jadon *et al.*, 2017). For instance, Akay & Karaboga (2012) introduced a modification rate (MR) to control the number of variables that can be inherited from the previous solution, which has a similar role as the crossover rate (CR) in DE. Therefore, in this part of work, a DE-based search strategy is utilized in the employed bee phase. Nevertheless, different from the existing literature, we provide a novel way of varying the frequency of perturbation. The parameters CR and MR are pre-defined control parameters that play an essential role in corresponding algorithms. Yet, it is difficult to define the parameter value appropriately for all kinds of problems. These control parameters are usually constant or updated with predetermined adaptation methods (Chen *et al.*, 2019b; Qin *et al.*, 2009; Wu *et al.*, 2016; Zhang & Sanderson, 2009), like the one adopted in the ABCDC algorithm. However, the adaptation approaches still heavily rely on the designer's experience.

With the purpose of setting and adjusting the control parameters more intelligently, reinforcement learning (RL) has attracted our attention. RL is one of the most important machine learning approaches that can solve various problems by learning from the interaction between a decision-making agent and an environment (Sutton & Barto, 2018). More details will be introduced in the next section. Some RL methods have been embedded with meta-heuristic algorithms to improve their performance. For instance, RL was utilized to select a suitable search strategy for the proposed ABC algorithm (Zhao & Zhang, 2020). Nonetheless, there exist few works on tuning the parameters of optimization algorithms via RL. Emary *et al.* (2017) were incorporated RL and neural networks into gray wolf optimization (GWO) to adjust the value of exploration rate intelligently. In order to enhance the performance of the simulated annealing (SA)

algorithm (Samma *et al.*, 2020), two key control parameters were controlled by using Q-learning. A self-learning GA was proposed by Chen *et al.* (2020) to solve a flexible job-shop scheduling problem. SARSA and Q-learning were utilized to adjust the values of mutation probability and crossover probability. Moreover, Hu *et al.* (2021) used RL to adjust the scale factor of the solution search equation in DE.

To the best of our knowledge, the study of adjusting the parameters of ABC via RL has not been sufficient so far. Considering the excellent performance of integrating RL with other meta-heuristic algorithms, one principle objective of this work is to overcome the shortcomings of ABC by incorporating RL appropriately. Therefore, RL is adopted to vary the number of dimensions to be updated (nb_{up}) in solution search equation of ABC. The reward value of RL is defined based on the comparison result between the original solution and the newly generated one. In this case, more information can be learned appropriately from the previous updating experience. And nb_{up} can be adjusted at different stages of the search process. Moreover, RL is adopted to set the nb_{up} for each employed bee independently rather than setting the same value for all the population.

As summarized in the first chapter, numerous ABC variations involved the information about the global best solution to help the colony search in a promising direction (Banharnsakun *et al.*, 2011; Cui *et al.*, 2022; Gao *et al.*, 2012, 2015a; Jadon *et al.*, 2017; Li *et al.*, 2015; Lin *et al.*, 2018; Zhu & Kwong, 2010). Moreover, surprising results have been obtained when phases employed bees and onlooker bees utilize different search strategies (Cui *et al.*, 2022; Gao *et al.*, 2015a; Jadon *et al.*, 2017; Karaboga & Gorkemli, 2014; Song *et al.*, 2017; Wang *et al.*, 2020). It is important to note that the success of these kinds of strategies was also confirmed by the previous ABCDC algorithm's solid performance. Hence, in this chapter, the search behavior of onlooker bees in the ABC_RL algorithm is different from that of the employed bees. More precisely, the local search in the onlooker bee phase is boosted by using the global best solution.

In this chapter, in order to be able to adjust certain important parameters more intelligently while improving the performance of the algorithm, the combination of RL and ABC algorithm is developed and the performance is investigated. As a result, the improvement strategies can be summarized as follows: firstly, RL is adopted to enlarge and adjust the frequency of perturbation of employed

bee phase intelligently considering the immediate reward from solution update results. Secondly, two enhanced solution search equations are utilized to achieve a nice balance between exploration and exploitation. Thirdly, a type of heavy-tailed distribution, the Mittag-Leffler distribution, is used to generate the scale factors of search equations. Finally, with the purpose of validating the performance of ABC_RL, it is compared with five advanced ABC algorithms on the Congress on Evolutionary Computation 2017 (CEC 2017) benchmark functions.

The rest of this chapter is organized as follows. The preliminaries of RL method are explained in section 3.2. In section 3.3, the proposed ABC_RL algorithm is introduced. Experimental studies are presented in section 3.4. Finally, the conclusion is given in the section 3.5.

3.2 Preliminaries

3.2.1 Reinforcement learning (RL)

Machine learning (ML) is one of the most essential components of artificial intelligence which has continued to revolutionize technologies since the last century. Countless challenges have been tackled with increasing accuracy and astonishing results in various fields like supply chain dynamics (Arora & Majumdar, 2022), healthcare (Houssein *et al.*, 2021; Maqsood *et al.*, 2022), etc. ML algorithms are often summarized into three categories: firstly, supervised learning trains a classifier with labeled datasets in order to classify or predict data; secondly, unsupervised learning can analyze hidden patterns in unlabeled data (Hussein *et al.*, 2019); last but not the least, RL allows a decision-making agent to take different actions in an environment and learn good policies with an explicit goal of maximizing the cumulative reward (Chen *et al.*, 2020; Sutton & Barto, 2018).

In RL algorithms, except for the agent and environment, there are other main components: actions, states, and rewards that make up a formal framework of RL. The interaction between the agent and its environment is shown in Figure 3.1. Each time, the agent selects an action a to perform regarding its current state. Then, after taking action, its state is updated and a reward is given by the environment. This reward value will be concerned when selecting the next action. And the loop continues until the termination condition is reached.

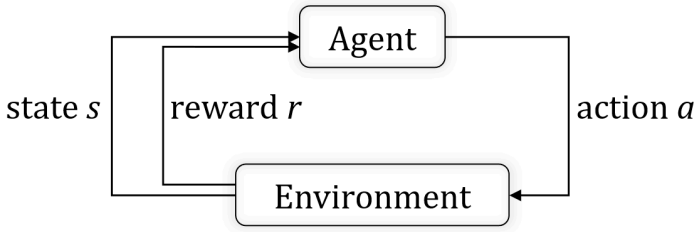


Fig. 3.1. The framework of reinforcement learning

Q-learning (Watkins & Dayan, 1992), one of the most famous model-free RL algorithms, is able to approximate the optimal action-value function by using a learned action-value function Q (Sutton & Barto, 2018). The function values of all possible state-action pairs are recorded in a Q table. Suppose the agent has p possible states and q actions can take, the form of the Q table can be found in Table 3.1.

Table 3.1: Form of Q table in Q-learning method

		Action			
		a_1	a_2	$\cdots$	a_q
State	s_1	$Q(s_1, a_1)$	$Q(s_1, a_2)$	$\cdots$	$Q(s_1, a_q)$
	s_2	$Q(s_2, a_1)$	$Q(s_2, a_2)$	$\cdots$	$Q(s_2, a_q)$
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	s_p	$Q(s_p, a_1)$	$Q(s_p, a_2)$	$\cdots$	$Q(s_p, a_q)$

The values in this table, Q values, indicate the quality of taking certain action a at state s. After taking an action, the corresponding Q value is updated by considering the immediate reward from the environment and the current Q table via Eq.(3.1) (Hu *et al.*, 2021; Watkins & Dayan, 1992). In addition, a policy π is utilized to choose an action each time. The $\epsilon - greedy$ strategy is adopted and is presented in subsection 3.3.2.2.

$$Q_{new}(s_t, a_t) = (1 - \alpha) \times Q(s_t, a_t) + \alpha \times \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) \right], \tag{3.1}$$

where $Q(s_t, a_t)$ is the Q value of acting a_t at current state s_t, r_{t+1} indicates the immediate reward after executing action a_t. $\alpha \in [0, 1]$ denotes the learning rate

and γ is the discount-rate parameter within range $[0, 1]$.

Remark 3.1 *$\gamma = 0$ is responsible for trading off the importance of immediate and future rewards. When $\gamma = 0$, only current rewards are taken into account. And when $\gamma = 1$, Q-learning looks for long-term rewards.*

The framework of Q-learning is described in Algorithm 8. In the initialization part, the Q table is generated as a zero-value matrix, and a state s_t is arbitrarily selected. Then the learning process starts by repeating several steps (lines 4-7) until the stopping condition is reached.

Algorithm 8 Pseudo-code of Q-learning algorithm

1: Initialize $Q(s, a)$
2: Select an initial state s_t randomly
3: **repeat**
4: Choose an action a_t for current state s_t from Q table via policy π (e.g., $\epsilon - greedy$)
5: Perform action a_t and get reward r_{t+1}
6: Evaluate $Q_{new}(s_t, a_t)$ with Eq.(3.1) and update Q table
7: Update current state $s_t \leftarrow s_{t+1}$
8: **until** the termination condition is met

3.3 Proposed ABC_RL algorithm

The proposed RL-based ABC algorithm (ABC_RL for short) is introduced in this section. In ABC_RL, RL is incorporated in the employed bee phase to adjust the number of dimensions to be updated (nb_{up}) each time. Hence, the search strategy of onlooker bees is improved by collecting information from the global best and two neighbors. And random numbers based on heavy-tailed distribution are used as scale factors in the search strategies.

3.3.1 Scale factors based on heavy-tailed distribution

Randomness has a comparatively large impact on both intensification and diversification of a meta-heuristic algorithm. In basic ABC, a uniformly distributed random number in the range of $[-1, 1]$ is utilized. This restrictive range has been found to be quite narrow, which may reduce the search efficiency. Additionally, if

ABC gets caught in local optima, this definition may be useless. Compared with ABC, the CS algorithm adopted a different way to produce its random parameters. Lévy flights are employed in the search equation to enhance the randomness. Lévy flight is a kind of random walk where the step lengths are drawn from a type of heavy-tailed probability distribution, namely the Lévy distribution (Yang, 2020; Yang & Deb, 2009). To tackle the lack of exploitation in ABC, Aydoğdu *et al.* (2016) replaced the production of new solutions in the scout bee phase by Lévy flights. Besides, Lévy flight was utilized to enhance the effectiveness of GWO, FA, and PSO algorithms (Heidari & Pahlavani, 2017; Jensi & Jiji, 2016; Kalantzis *et al.*, 2016).

For the purpose of investigating the optimal randomness in swarm-based algorithms, Wei *et al.* (2019) incorporated different types of heavy-tailed distributions into CS and compared them with the original CS (i.e., CS with Lévy flights). And these randomness-enhanced CS variants have been found to outperform the basic CS algorithm. In (Yousri *et al.*, 2021), other types of heavy-tailed distributions were adopted instead of the Lévy distribution aiming at improving the performance of the proposed FOCS algorithm. As for the ABC algorithm, only Lévy distribution has been utilized in some ABC variants. To the best of our knowledge, other types of heavy-tailed distributions have not yet been employed in ABC and related studies are inadequate. Moreover, the other kinds of heavy-tailed distributions have demonstrated their advantages compared with the Lévy distribution in experimental studies (Wei *et al.*, 2019).

Based on the discussion above, in this work, the heavy-tailed distribution is adopted to help with the randomness. The Mittag-Leffler distribution, one of the most common heavy-tailed distributions, is utilized in the proposed algorithm to generate the scale factors in solution search equations. The definition of Mittag-Leffler distribution is described as follows.

A random variable is said to be subjected to Mittag-Leffler distribution if its distribution function has the following form (Huillet, 2016; Wei *et al.*, 2019):

$$F_\beta(x) = \sum_{k=1}^{\infty} \frac{(-1)^{k-1} x^{k\beta}}{\Gamma(1+k\beta)}, \tag{3.2}$$

where $\Gamma(\cdot)$ is the Gamma function. And $x > 0$, $\beta \in [0, 1]$, $F_\beta = 0$ for $x \leq 0$. The Mittag-Leffler distribution is heavy-tailed when $0 < \beta < 1$ and it is an

exponential distribution if $\beta = 1$.

For the implementation, a convenient expression proposed by Kozubowski & Rachev (1999) is adopted:

$$\tau_\beta = -\gamma \times \ln u \left(\frac{\sin(\beta\pi)}{\tan(\beta\pi v)} - \cos(\beta\pi) \right)^{1/\beta}, \tag{3.3}$$

where τ_β is a Mittag-Leffler random number, γ is the scale parameter, and u, v are two independent uniform random numbers. The parameters are set as $\gamma = 4.5$ and $\beta = 0.8$, based on the suggestions of Wei *et al.* (2019).

Remark 3.2 *The random factor τ produced via Eq.(3.3) is adopted in both employed bee phase and onlooker bee phase. And to fit the problem scale, τ is multiplied with a coefficient. According to the different responsibility of the two phases, the corresponding coefficients are set differently. $0.07 \times \tau$ is utilized in the employed bee phase whereas onlooker bee phase uses $0.05 \times \tau$. The onlooker bee phase is supposed to exploit locally, so its coefficient value is relatively small to avoid jumping over the optimum.*

3.3.2 Employed bee phase with RL

3.3.2.1 Differential search strategy

In fact, the way a search equation gathers useful information can greatly affect its search effectiveness. As mentioned earlier, the search equation of standard ABC was found to collect limited information at a time. In Eq.(1.2), the value in only one dimension of one neighbor is considered when producing a new feasible solution. Note that the search abilities of ABC and DE are compared by analyzing their variations of individuals in the equations (Xu *et al.*, 2020). And DE has demonstrated that it is able to search and converge more efficiently than ABC because DE's solution search equation has more possibility to collect useful information from other members.

In addition, DE has more flexibility in terms of the nb_{up} each time. Its mutation operator updates individuals on total dimensions at first, then the crossover and selection processes allow the individuals to collect messages from both the previous generation and the latest mutation. Eq.(5.7) presents a basic version of DE search strategy, "*rand/1/bin*".

$$u_{i,j} = \begin{cases} x_{r_1,j} + F \times (x_{r_2,j} - x_{r_3,j}), & if \quad rand \leqslant CR \ or \ j = j_{rand} \\ x_{i,j}, & otherwise \end{cases} \tag{3.4}$$

where $r_1 \neq r_2 \neq r_3 \neq i$ are randomly selected from $\{1, \cdots, N\}$, j_{rand} is randomly chosen among all the dimensions. $F \in [0, 1]$ is the scale factor while CR is crossover rate. And u_i is the newly produced solution based on x_i. According to the expression, each time the nb_{up} is possible to be any integer between 1 and D.

Furthermore, a parameter M was adopted to determine the amount of variables to be updated in order to improve information sharing among the colony (Xiang *et al.*, 2017). And Xiang *et al.* (2021) defined a range to randomly generate the number of components to be modified. Nevertheless, most of these enhancements depend on the parameter settings, and it is nearly impossible to determine a fixed configuration that solves every issue (Hu *et al.*, 2021).

In this work, the search strategy of employed bees phase will be enhanced from two aspects, i.e., the amount of information learned from the swarm as well as the nb_{up}. DE-based search equation (3.5) is utilized while the RL is introduced to tune the nb_{up}.

$$v_{i,j} = x_{k_1,j} + c_1 \times \tau \times (x_{k_2,j} - x_{k_3,j}), \tag{3.5}$$

where $k_1 \neq k_2 \neq k_3 \neq i$ are randomly selected from $\{1, \cdots, SN\}$ and $j \in \{1, \cdots, D\}$ is the chosen dimension to be updated. τ is the heavy-tailed random scale factor produced by Eq.(3.3) and $c_1 = 0.07$.

3.3.2.2 Adjusting parameter nb_{up} with Q-learning

The method of utilizing Q-learning to define the nb_{up} during the search process is described in the following. The nb_{up} is adjusted by altering the proportion to the total dimension (d_{ratio}) rather than adding or subtracting a fixed number of dimensions each time in order to tackle problems of various scales. And the range of possible values for the parameter d_{ratio} is $\{0.1, 0.2, \cdots, 0.9\}$.

Three actions are defined to adjust the value of d_{ratio}: stays the same, add 0.1 and subtract 0.1. As mentioned before, the reward of taking action a at state s is recorded as the value of $Q(s, a)$ in the Q table. In the proposed algorithm, each

employed bee selects an action according to its associated Q table. After taking the action on its d_{ratio}, the nb_{up} is calculated as below:

$$nb_{up} = ceil(d_{ratio} \times D), \tag{3.6}$$

where function $ceil(\cdot)$ is used to compute the smallest integer that is greater than or equal to $d_{ratio} \times D$.

And then nb_{up} variables are randomly selected among the D variables to be modified. Thereafter, the new candidate solution is generated via Eq.(3.5). According to the comparison results between the new solution and the previous one, a reward is given and the state is updated. The last step of implanting Q-learning is to update the corresponding Q value via Eq.(3.1). Notice that, each bee has its corresponding Q table in order to avoid affecting each other.

State and reward set

Since the objective is to minimize the objective functions value, so two states are defined considering the result of the updates.

s_1: $f(v_i) < f(x_i)$, the new generated solution is better than the previous one. And the corresponding reward value is set as 1;

s_2: $f(v_i) \geqslant f(x_i)$, the new solution fails to outperform the original solution, in this case the reward value is 0.

Action set

There are three possible actions that can be selected and executed by the agent:

a_1	a_2	a_3
d_{ratio} stays same	$d_{ratio} + 0.1$	$d_{ratio} - 0.1$

Action selection strategy

In fact, always selecting the action with the highest estimated Q value is the simplest selection strategy. However, greedy selection may ignore certain actions with better potential. To get over this weakness, $\epsilon - greedy$ is proposed to occasionally select the actions with smaller values, while most of the time still select the actions greedily (Sutton & Barto, 2018). And ϵ is named the greedy rate. This mechanism allows the Q-learning method to balance exploration and

exploitation well (Chen *et al.*, 2020; Shahrabi *et al.*, 2017). The expression of $\epsilon - greedy$ is as below:

$$\pi(s_t, a_t) = \begin{cases} \max\limits_{a} Q(s_t, a), & if \quad 1 - \epsilon \geqslant rand, \\ a_{rand}, & otherwise, \end{cases} \tag{3.7}$$

where $rand$ is a random number within $[0, 1]$. And a_{rand} is randomly selected among the action set $\{a_1, \cdots, a_q\}$ with probability ϵ.

Then, the proposed algorithm is able to alter the nb_{up} considering the historical updating experiences. The pseudo-code of modified employed bee phase is shown in Algorithm 9 in order to present it more clearly. For each candidate solution, an action is firstly selected based on its $Q-$table (line 4). Hence, those dimensions that will be updated can be determined through lines 5-6. After updating the candidate solution, the state and reward value corresponding to the action are given based on the result of greedy selection (lines 10-17). At the end, the $Q-$table is updated as expressed in line 18. Note that, the initialization of parameter d_{ratio} will be discussed in subsection 3.4.2.

Algorithm 9 Pseudo-code of RL-based employed bee phase

1: Initialize $Q-$tables for all the employed bees and initialize state s_t randomly
2: Initialize $d^i_{ratio} = 0.2$, $i = 1, \cdots, SN$
3: **for** each employed bee **do**
4: Choose an action a_t for current state s_t from $Q-$table via $\epsilon - greedy$ Eq.(3.7)
5: Perform action a_t on adjusting d^i_{ratio}
6: Calculate nb_{up} and randomly select nb_{up} dimensions to update
7: **for** j $\in$ selected dimensions **do**
8: Update the $j-$th dimension of candidate solution via Eq.(3.5)
9: **end for**
10: **if** $f(v_i) < f(x_i)$ **then**
11: Replace x_i with v_i; $trial_i = 0$
12: $reward = 1$
13: $state = s_1$
14: **else** $trial_i = trial_i + 1$
15: $reward = 0$
16: $state = s_2$
17: **end if**
18: Evaluate $Q_{new}(s_t, a_t)$ with Eq.(3.1) and update corresponding $Q-$table
19: **end for**

Remark 3.3 *By using RL, the proposed algorithm can not only enlarge the nb_{up}, but also adjust parameter's value considering on-line learning rather than requiring a predefinition by users.*

3.3.3 Improved onlooker bee phase

Considering the weakness of the basic ABC algorithm, an enhanced solution search equation is used in the onlooker bee phase of our algorithm. In the proposed search strategy Eq.(3.8), the information of the global best solution as well as two neighbors are taken into account. Moreover, the scale factor of the second term is also generated with the Mittag-Leffler distribution in order to increase the randomness. Meanwhile, the factor of the third term is set as a random number in a range of $[0, 1]$ to help the newly produced solution learn from the global best solution stably.

$$v_{i,j} = x_{i,j} + c_2 \times \tau \times (x_{k_1,j} - x_{k_2,j}) + \theta \times (x_{best,j} - x_{i,j}) \tag{3.8}$$

where $\theta \in [0, 1]$ is random number drawn from uniform distribution whereas τ is generated by Eq.(3.3) and $c_2 = 0.05$. $k_1 \neq k_2 \neq i$ are randomly chosen from the colony and x_{best} is the global best solution found so far.

3.3.4 The framework of ABC_RL algorithm

In order to explain the entire process of the proposed algorithm, its framework and flowchart of ABC_RL are shown in Algorithm 10 and Figure 3.2, respectively. At the beginning, the SN candidate solutions and parameters of Q-learning are initialized. Then in each iteration, the employed bee phase with RL is firstly executed to update the candidate solutions. For each candidate solution, Q-learning is adopted to define and adjust the number of dimensions being updated. And similar to the standard ABC procedure, onlooker bee phase and scout bee phase are executed afterwards.

Algorithm 10 Pseudo-code of ABC_RL algorithm

Generate initial population x_i, $i = 1, \cdots, SN$ with Eq.(1.1)
Evaluate objective function values $f(x_i)$, $FES = SN$
Initialize Q-tables, $d^i_{ratio} = 0.2$ for $i = 1, \cdots, SN$
while $FES \leq max\_FES$ **do**
    % `RL-based employed bee phase` %
    for $i = 1 \rightarrow SN$ **do**
        Randomly select $k_1 \neq k_2 \neq k_3 \neq i$ from $\{1, \cdots, SN\}$
        Choose an action via $Q-\text{table}_i$ via Eq.(3.7)
        Take action by adjusting d^i_{ratio} and calculate nb_{up}
        Randomly select nb_{up} dimensions from $\{1, \cdots, D\}$
        for $j \in$ selected dimensions **do**
            Generate τ via Eq.(3.3)
            Produce $v_{i,j}$ via Eq.(3.5)
        end for
        if $f(v_i) < f(x_i)$ **then**
            Replace x_i with v_i; $trial_i = 0$
            $reward = 1$
            $state = s_1$
        else $trial_i = trial_i + 1$
            $reward = 0$
            $state = s_2$
        end if
        Update $Q-\text{table}_i$ via Eq.(3.1)
    end for
    Evaluate the probability values P_i with Eq.(1.5)
    % `Improved onlooker bee phase` %
    for $t = 1 \rightarrow SN$ **do**
        Select x_i based on P_i by roulette wheel selection method
        Generate τ via Eq.(3.3)
        Randomly select $k_1 \neq k_2 \neq i$ and j_{rand} from $\{1, \cdots, D\}$
        Generate the $v_{i,j_{rand}}$ via Eq.(3.8)
        if $f(v_i) < f(x_i)$ **then**
            Replace x_i with v_i; $trial_i = 0$
        else $trial_i = trial_i + 1$
        end if
    end for
    % `Scout bee phase` %
    for $i = 1 \rightarrow SN$ **do**
        if $trial_i > limit$ **then**
            Generate new position with Eq.(1.1), $trial_i = 0$
        end if
    end for
end while

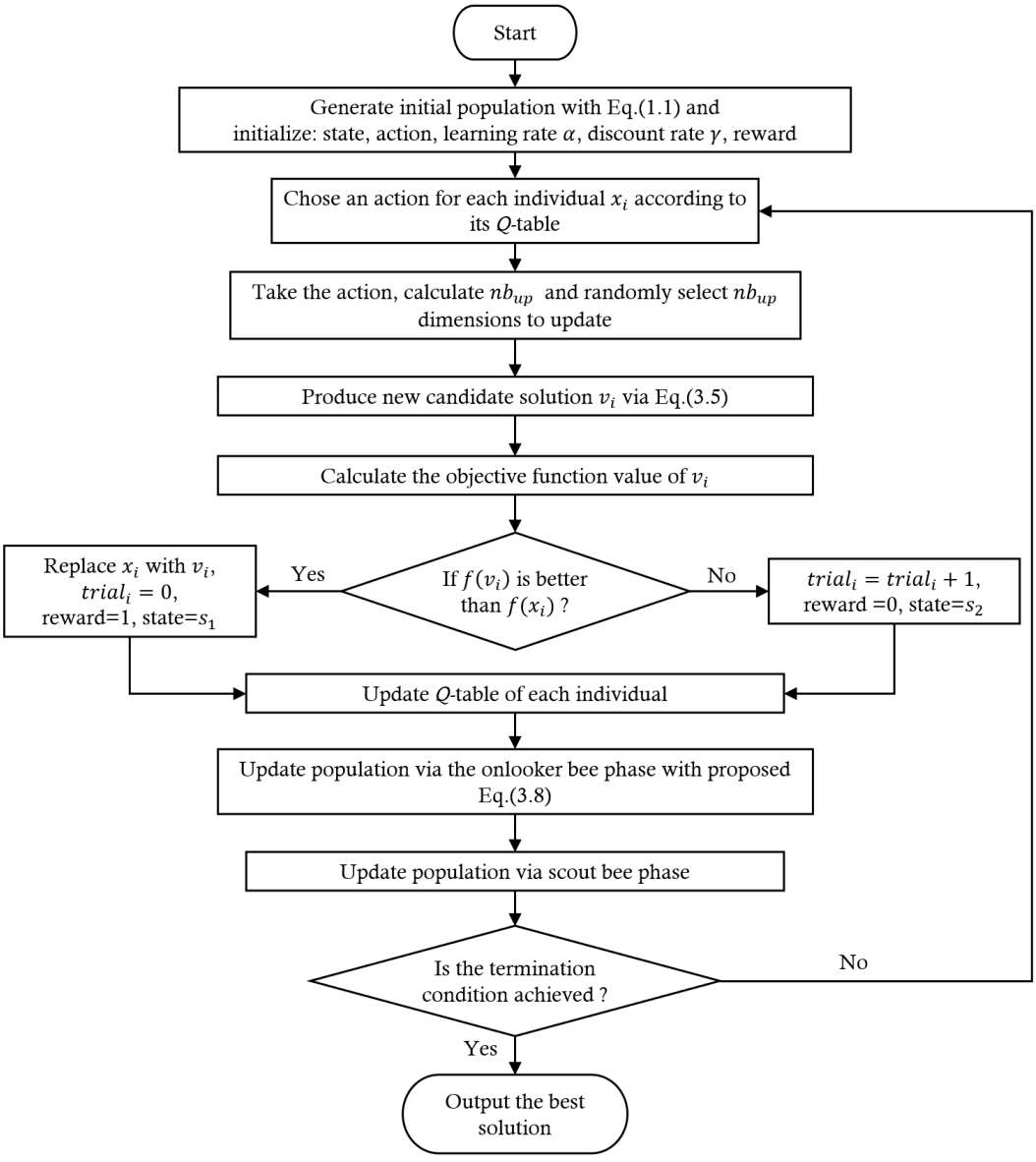


Fig. 3.2. The flowchart of ABC_RL algorithm

3.4 Experiments on function optimization problems

In this part, the initialization method of parameter d_{ratio} was studied at first. Hence, the experimental verification was done by comparing the proposed algorithm with five state-of-art ABC variants on CEC 2017 benchmark problems in different dimension cases.

3.4.1 CEC 2017 benchmark problems

In the experiments, 29 CEC 2017 benchmark problems (Awad *et al.*, 2017) were chosen to be solved as various kinds of single-objective optimization problems were contained. There are 2 unimodal functions (f_1, f_3), 7 simple multimodal functions ($f_4 - f_{10}$), 10 hybrid functions ($f_{11} - f_{20}$) and 10 composition functions ($f_{21} - f_{30}$). In fact, it is hard to distinguish the algorithms' performance on certain classical benchmarks as most of the improved ABC variants can find the optima effectively. In this case, the CEC 2017 benchmarks are selected to better test the compared methods.

According to the evaluation criteria of CEC 2017, the search space for all the benchmarks is defined as $[-100, 100]^D$. The determination condition is set in terms of the maximum number of function evaluations and problem dimension, i.e., $max\_FES = 10^4 \times D$. Moreover, the official code is accessible online*. Notice that the function f_2 is not included in the following experiments because it has been deleted in the latest official code.

3.4.2 Effects of the initial value of parameter d_{ratio}

The parameter d_{ratio} is used to control the frequency of perturbation in ABC_RL. For each employed bee, each time $d_{ratio} \times D$ dimensions of the previous candidate solution will be updated. Note that d_{ratio} can take value from the set $\{0.1, 0.2, \cdots, 0.9\}$ and is defined by Q-learning algorithm. In this section, the setting of the initial value of parameter d_{ratio} is studied since the initialization method can impact the following search process.

*https://github.com/P-N-Suganthan/CEC2017-BoundContrained

3. REINFORCEMENT LEARNING BASED ABC ALGORITHM (ABC_RL)

Hu *et al.* (2021) randomly initialized the parameter F that tuned by RL in their algorithm. Nevertheless, besides setting different initial values for different individuals, it is also possible to define a uniform initial value for the swarm. It is worth investigating which initialization method can provide more help to the algorithm.

In this context, different methods to define the initial values of d_{ratio} are implemented and compared with standard ABC together. Firstly, variant ABC_RL$_{rand}$ is constructed with the random initialization method. In other words, the d_{ratio} of each employed bee is initialized randomly and independently. Secondly, variants that use predefined d_{ratio} are built as well. In this uniform initialization manner, all the employed bees have the same d_{ratio} value in the beginning stage. And three values (0.2, 0.5, and 0.8) are selected as initial values in the comparison. ABC_RL$_{0.2}$, ABC_RL$_{0.5}$, and ABC_RL$_{0.8}$ stand for the cases of defining d_{ratio} with low, middle, and high value.

Table 3.2 presents the comparison results of these ABC_RL variants on benchmark problems with $D = 10$. In the comparison, the mean of function error values $f(X_{best}) - f(X^*)$ over 25 independent runs are calculated, where X_{best} is the best solution found by algorithm and X^* is the exact global optimum. For each function, the best results are marked in **boldface**. In order to better analyze the results, the Friedman test and Wilcoxon tests are conducted based on the average errors obtained by algorithms. The rankings in the last row are evaluated by Friedman test. The best ranking is marked in **boldface**. The Wilcoxon test is able to compare the difference between the two methods, so each ABC_RL variant is compared to the original ABC algorithm. The symbol "+/=/-" indicates the number of functions that ABC_RL is better, similar, or worse than ABC.

According to the comparison, the ABC_RL$_{0.2}$ perform the best and is followed by the version with random initialization ABC_RL$_{rand}$. It can be found that, when the parameter d_{ratio} is defined relatively large in the beginning, the final results are less satisfying. Likewise, the same conclusion can be drawn from Wilcoxon test results. ABC_RL$_{0.2}$ outperforms the original ABC on 19 functions while ABC_RL$_{0.8}$ surpasses ABC on 16 functions. Therefore, when the initial value of parameter d_{ratio} is defined relatively small, the proposed algorithm is able to show better performance. And the initial value of d_{ratio} is set as 0.2 in the following part of experiments. Furthermore, all the concerned ABC_RL versions

Table 3.2: Comparison of initialization methods for parameter d_{ratio}

Function	ABC	ABC_RL$_{rand}$	ABC_RL$_{0.2}$	ABC_RL$_{0.5}$	ABC_RL$_{0.8}$
f_1	**2.88E+02**	1.60E+03	1.10E+03	9.29E+02	3.04E+03
f_3	6.66E+02	2.99E+00	7.97E+00	4.24E+00	**3.23E-01**
f_4	**2.28E-01**	3.26E+00	3.24E+00	3.17E+00	3.62E+00
f_5	7.71E+00	4.08E+00	3.55E+00	3.56E+00	**3.43E+00**
f_6	3.35E-09	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_7	1.75E+01	**1.35E+01**	1.42E+01	1.39E+01	1.39E+01
f_8	8.00E+00	3.77E+00	3.40E+00	3.87E+00	**3.34E+00**
f_9	6.04E-03	**0.00E+00**	4.55E-15	**0.00E+00**	**0.00E+00**
f_{10}	2.50E+02	1.20E+02	1.67E+02	1.23E+02	**9.11E+01**
f_{11}	4.05E+00	2.08E+00	**1.91E+00**	2.04E+00	1.97E+00
f_{12}	4.29E+04	**1.10E+04**	1.14E+04	1.66E+04	2.38E+04
f_{13}	**5.98E+02**	2.76E+03	1.34E+03	1.74E+03	4.92E+03
f_{14}	1.72E+02	5.93E+01	**2.19E+01**	2.79E+01	2.47E+01
f_{15}	9.76E+01	6.25E+01	**3.72E+01**	4.13E+01	1.34E+02
f_{16}	8.06E+00	2.70E+00	**2.49E+00**	1.20E+01	1.06E+01
f_{17}	2.35E+00	1.22E+00	**1.11E+00**	1.84E+00	2.07E+00
f_{18}	1.53E+03	2.07E+03	1.61E+03	**1.20E+03**	2.98E+03
f_{19}	9.75E+01	**1.96E+01**	5.98E+01	2.77E+01	6.20E+01
f_{20}	2.84E-01	4.99E-02	**1.25E-02**	1.25E-02	8.74E-02
f_{21}	1.10E+02	1.40E+02	**1.02E+02**	1.53E+02	1.71E+02
f_{22}	**7.54E+01**	8.27E+01	7.75E+01	8.02E+01	9.03E+01
f_{23}	**3.02E+02**	3.08E+02	3.07E+02	3.07E+02	3.07E+02
f_{24}	**1.04E+02**	2.23E+02	1.58E+02	2.82E+02	2.82E+02
f_{25}	**2.19E+02**	4.06E+02	3.82E+02	4.06E+02	4.08E+02
f_{26}	**1.07E+02**	2.64E+02	2.18E+02	2.84E+02	3.04E+02
f_{27}	3.95E+02	3.90E+02	3.90E+02	3.90E+02	**3.90E+02**
f_{28}	**2.77E+02**	3.35E+02	2.87E+02	3.62E+02	3.96E+02
f_{29}	2.53E+02	2.48E+02	2.44E+02	2.43E+02	**2.43E+02**
f_{30}	1.87E+04	**3.88E+03**	4.68E+03	5.70E+03	7.23E+03
Wilcoxon +/=/-		18/0/11	19/0/10	18/0/11	16/0/13
Mean ranking	3.45	2.88	**2.36**	2.95	3.36

attained better rankings than the standard ABC algorithm did, which can verify the effectiveness of improvement strategies.

3.4.3 Comparison with ABC variants

To evaluate the performance of ABC_RL, four effective ABC variants and the standard ABC are involved in the following experiments. In order to compare fairly, the control parameters of the competitors are set the same as those of their original papers which are presented in Table 3.3 in order to compare fairly. Considering that the swarm size may also affect the results, SN values are consistent with the values defined in the corresponding papers. And for all the cases of dimension, SN stays the same.

The six algorithms are executed to solve the 29 CEC 2017 benchmark functions with $D = 10$, 30, and 50. Each method is run 51 independent times on all the problems. And the comparisons are carried out by comparing the mean and standard deviation (Std) of function error values.

Table 3.3: Parameter settings of ABC_RL and compared ABC algorithms

Algorithm	Parameter setting
ABC (Karaboga, 2005)	$SN = 50$, $limit = SN \times D$
sdABC (Chen *et al.*, 2019b)	$SN = 50$, $limit = SN \times D$, $pa_{min} = 0.2$
ARABC (Cui *et al.*, 2017a)	$SN = 50$, $limit = SN \times D$, $\Delta = 0.01$, $\alpha_{min} = 0$, $\alpha_{max} = 5$
ILTD_ABC (Gao *et al.*, 2019)	$SN = 40$, $limit = 100$
MGABC (Zhou *et al.*, 2021a)	$SN = 75$, $limit = 100$, $MR = 0.5$, $q = 0.1$, $p = 0.1$
ABC_RL	$SN = 50$, $limit = SN \times D$, $\alpha = 0.6$, $\gamma = 0.4$, $\epsilon = 0.3$

Tables 3.4 - 3.6 present the comparison results in terms of the mean and Std of the errors calculated. For each problem, the Wilcoxon tests are used to display the significance between ABC_RL and other competitors. The symbol "+" means that ABC_RL outperforms the compared algorithm whereas symbol "-" indicates that compared algorithm is better than ABC_RL. And "=" denotes that the two algorithms have similar results. The total numbers of each symbol are listed in the last row of each table. Furthermore, Friedman tests are conducted to evaluate the overall performance of concerned methods. The average rankings are presented in Figure 3.3.

Table 3.4 shows the performance of the compared algorithms on 10-dimensional problems. The advantages of the proposed algorithm are significant, it obtains the best results on 18 functions (f_3, $f_5 - f_{12}$, $f_{17} - f_{20}$, f_{22}, f_{23}, f_{27}, f_{29}, and f_{30}).

Table 3.4: Comparison between ABC_RL and other ABC variants with $D = 10$

Function		ABC	sdABC	ARABC	ILTD_ABC	MGABC	ABC_RL
f_1	Mean	2.88E+02 $^-$	1.66E+09 $^+$	1.88E+03 $^+$	1.66E+04 $^+$	2.41E+02 $^-$	9.80E+02
	Std	2.39E+02	2.89E+09	2.79E+03	4.06E+04	3.46E+02	1.41E+03
f_3	Mean	6.66E+02 $^+$	6.03E+03 $^+$	1.59E+04 $^+$	1.89E+03 $^+$	1.49E+03 $^+$	5.40E+00
	Std	3.85E+02	5.63E+03	5.55E+03	6.74E+02	8.26E+02	1.31E+01
f_4	Mean	2.28E-01 $^-$	1.96E+01 $^+$	3.70E+00 $^+$	4.97E+00 $^+$	6.09E+00 $^+$	2.91E+00
	Std	2.38E-01	5.11E+01	2.53E+00	1.41E+00	8.39E-01	1.70E+00
f_5	Mean	7.71E+00 $^+$	4.25E+01 $^+$	5.02E+00 $^+$	5.22E+00 $^+$	4.76E+00 $^+$	3.42E+00
	Std	1.96E+00	1.76E+01	1.61E+00	1.80E+00	1.35E+00	1.10E+00
f_6	Mean	3.35E-09 $^+$	4.57E+01 $^+$	2.22E-07 $^+$	3.70E-04 $^+$	0.00E+00 $^=$	0.00E+00
	Std	7.72E-09	9.48E+00	1.59E-06	2.01E-04	0.00E+00	0.00E+00
f_7	Mean	1.75E+01 $^+$	1.09E+02 $^+$	1.53E+01 $^+$	1.66E+01 $^+$	1.58E+01 $^+$	1.42E+01
	Std	2.33E+00	1.72E+01	1.83E+00	2.32E+00	1.39E+00	1.40E+00
f_8	Mean	8.00E+00 $^+$	4.42E+01 $^+$	5.74E+00 $^+$	6.59E+00 $^+$	4.26E+00 $^+$	3.78E+00
	Std	2.51E+00	1.17E+01	1.93E+00	3.35E+00	1.04E+00	1.37E+00
f_9	Mean	6.04E-03 $^+$	7.12E+01 $^+$	6.15E-06 $^+$	7.72E-05 $^+$	1.17E-05 $^+$	4.46E-15
	Std	1.83E-02	1.36E+02	2.13E-05	1.06E-04	1.46E-05	2.23E-14
f_{10}	Mean	2.50E+02 $^+$	1.24E+03 $^+$	2.05E+02 $^+$	1.86E+02 $^+$	2.00E+02 $^+$	1.38E+02
	Std	8.62E+01	4.14E+02	1.21E+02	1.26E+02	1.00E+02	9.46E+01
f_{11}	Mean	4.05E+00 $^+$	1.49E+01 $^+$	6.03E+00 $^+$	3.94E+00 $^+$	2.13E+00 $^+$	1.95E+00
	Std	1.54E+00	2.80E+01	6.74E+00	2.38E+00	1.08E+00	1.21E+00
f_{12}	Mean	4.29E+04 $^+$	4.65E+07 $^+$	1.51E+05 $^+$	1.67E+04 $^+$	2.27E+04 $^+$	1.66E+04
	Std	2.41E+04	9.75E+07	2.40E+05	2.02E+04	2.85E+04	1.45E+04
f_{13}	Mean	5.98E+02 $^-$	1.12E+03 $^-$	2.89E+03 $^+$	7.19E+03 $^+$	6.38E+03 $^+$	1.23E+03
	Std	4.90E+02	4.23E+03	2.95E+03	5.57E+03	3.09E+03	2.19E+03
f_{14}	Mean	1.72E+02 $^+$	7.79E+00 $^-$	1.80E+03 $^+$	2.99E+03 $^+$	2.04E+03 $^+$	2.82E+01
	Std	1.99E+02	1.16E+01	3.05E+03	1.92E+03	1.81E+03	4.50E+01
f_{15}	Mean	9.76E+01 $^+$	1.15E+01 $^-$	1.05E+03 $^+$	1.45E+03 $^+$	3.65E+02 $^+$	8.30E+01
	Std	2.03E+02	3.10E+01	1.43E+03	1.11E+03	6.00E+02	1.62E+02
f_{16}	Mean	8.06E+00 $^+$	1.83E+02 $^+$	1.24E+01 $^+$	4.33E+01 $^+$	2.59E+00 $^-$	3.17E+00
	Std	1.89E+01	1.25E+02	2.53E+01	5.93E+01	3.39E+00	9.08E+00
f_{17}	Mean	2.35E+00 $^+$	1.12E+02 $^+$	1.64E+00 $^+$	1.75E+01 $^+$	2.80E+00 $^+$	6.08E-01
	Std	1.29E+00	3.77E+01	1.15E+00	1.93E+01	3.94E+00	4.83E-01
f_{18}	Mean	1.53E+03 $^+$	4.31E+03 $^+$	4.84E+03 $^+$	7.77E+03 $^+$	1.34E+03 $^+$	9.88E+02
	Std	9.88E+02	1.55E+04	4.28E+03	6.61E+03	9.26E+02	1.11E+03
f_{19}	Mean	9.75E+01 $^+$	2.95E+02 $^+$	8.41E+02 $^+$	4.58E+03 $^+$	6.59E+02 $^+$	4.97E+01
	Std	1.18E+02	1.93E+03	1.21E+03	4.29E+03	8.10E+02	1.13E+02
f_{20}	Mean	2.84E-01 $^+$	1.20E+02 $^+$	4.76E+00 $^+$	5.24E+00 $^+$	3.58E-02 $^+$	3.06E-02
	Std	4.01E-01	1.05E+02	1.86E+01	2.33E+01	1.09E-01	9.38E-02
f_{21}	Mean	1.10E+02 $^-$	1.04E+02 $^-$	1.55E+02 $^+$	1.85E+02 $^+$	1.08E+02 $^-$	1.20E+02
	Std	3.36E+00	1.02E+01	4.11E+01	4.37E+01	2.55E+01	3.86E+01
f_{22}	Mean	7.54E+01 $^+$	4.05E+02 $^+$	8.08E+01 $^+$	9.43E+01 $^+$	9.24E+01 $^+$	7.39E+01
	Std	2.86E+01	2.63E+02	2.94E+01	2.38E+01	2.46E+01	3.41E+01
f_{23}	Mean	3.02E+02 $^+$	3.14E+02 $^+$	3.10E+02 $^+$	3.10E+02 $^+$	3.03E+02 $^+$	3.01E+02
	Std	5.95E+01	5.45E+00	2.65E+00	2.82E+00	4.29E+01	4.31E+01
f_{24}	Mean	1.04E+02 $^-$	1.86E+02 $^+$	2.56E+02 $^+$	3.23E+02 $^+$	1.60E+02 $^+$	1.42E+02
	Std	1.81E+01	1.17E+02	7.85E+01	6.57E+01	9.44E+01	8.29E+01
f_{25}	Mean	2.19E+02 $^-$	4.71E+02 $^+$	3.91E+02 $^-$	4.16E+02 $^+$	4.16E+02 $^+$	4.05E+02
	Std	1.10E+02	1.21E+02	6.67E+01	2.23E+01	2.00E+01	1.68E+01
f_{26}	Mean	1.07E+02 $^-$	4.80E+02 $^+$	2.34E+02 $^-$	2.76E+02 $^+$	2.42E+02 $^-$	2.47E+02
	Std	9.92E+01	3.70E+02	1.03E+02	5.51E+01	1.08E+02	1.06E+02
f_{27}	Mean	3.95E+02 $^+$	3.93E+02 $^+$	3.95E+02 $^+$	3.91E+02 $^+$	3.92E+02 $^+$	3.90E+02
	Std	2.30E+00	9.45E+00	3.27E+00	2.02E+00	1.87E+00	1.16E+00
f_{28}	Mean	2.77E+02 $^-$	5.20E+02 $^+$	4.46E+02 $^+$	4.44E+02 $^+$	2.99E+02 $^-$	3.06E+02
	Std	8.66E+01	1.36E+02	1.11E+02	1.54E+02	9.22E+01	7.97E+01
f_{29}	Mean	2.53E+02 $^+$	2.73E+02 $^+$	2.58E+02 $^+$	2.57E+02 $^+$	2.61E+02 $^+$	2.44E+02
	Std	2.60E+01	5.37E+01	2.30E+01	9.39E+00	7.25E+00	5.56E+00
f_{30}	Mean	1.87E+04 $^+$	1.47E+06 $^+$	2.09E+04 $^+$	1.74E+04 $^+$	2.04E+04 $^+$	3.85E+03
	Std	2.03E+04	3.33E+06	2.86E+04	1.88E+04	1.85E+04	3.13E+03
Wilcoxon	+/=/-	21/0/8	25/0/4	27/0/2	29/0/0	23/1/5	

In this comparison, the search effectiveness of ABC_RL can be proved by solving different types of problems rather than only being good at solving one kind of problem. It is worth pointing out that, the ABC_RL algorithm's exploitation and exploration abilities have been well balanced during the search process compared to other competitors.

Considering the results of Wilcoxon tests, ABC_RL surpasses the original ABC on 21 out of 29 problems, and its advantages on $f_3, f_6, f_9, f_{12}, f_{17}$, and f_{20} are remarkable. sdABC is slightly better than ABC_RL on 4 problems. And ABC_RL outperforms sdABC on the rest 25 problems. Meanwhile, ABC_RL has more accurate solutions than ARABC on 27 problems while ARABC outperforms ABC_RL on 2 functions. ILTD_ABC fails to surpass ABC_RL on all the benchmarks in the case of $D = 10$. In fact, only MGABC and ABC_RL achieve the exact optimum on f_6. MGABC gets smaller errors than ABC_RL on 5 functions while ABC_RL performs better on the remaining 23 functions.

The comparison results with $D = 30$ are listed in Table 3.5. Notice that the search difficulty will increase as the problem scale augments, but the population size remains the same. In this context, the advantages of ABC_RL compared to the other effective ABC algorithms haven't been affected greatly. Moreover, its superiorities are obvious on f_6, f_9, and f_{11}. The Wilcoxon tests show that ABC_RL outperforms ABC on 19 functions and ABC has better results on 10 functions. The number of functions where ABC_RL has more accurate solutions than sdABC is 28 problems and sdABC achieves a smaller error on only 1 function, f_{14}. And ABC_RL gains better results than ARABC on 21 problems and fails on 8 problems. ILTD_ABC and MGABC become more competitive in the case of $D = 30$. ILTD_ABC obtains smaller errors than ABC_RL on 9 functions, while ABC_RL achieves more accurate solutions on the rest 20 problems. MGABC performs better than ABC_RL on 10 functions and fails to outperform the proposed algorithm on the rest 19 functions.

Table 3.6 presents the comparison with $D = 50$. According to the Wilcoxon test results, similar conclusions can be derived from this comparison. The original ABC surpasses ABC_RL on 9 functions while ABC_RL performs better on the rest 20 functions. Compared with sdABC, ABC_RL obtains smaller errors on 26 functions and larger errors on 3 functions. Although ARABC has outstanding performance in this case, ABC_RL is still competitive to it.

Table 3.5: Comparison between ABC_RL and other ABC variants with $D = 30$

Function		ABC	sdABC	ARABC	ILTD_ABC	MGABC	ABC_RL
f_1	Mean	2.15E+02 $^-$	8.81E+09 $^+$	2.77E+03 $^+$	4.23E+03 $^+$	1.70E+03 $^-$	1.74E+03
	Std	2.66E+02	7.91E+09	4.95E+03	4.05E+03	1.06E+03	3.06E+03
f_3	Mean	7.29E+04 $^+$	5.35E+04 $^+$	1.44E+05 $^+$	5.63E+04 $^+$	4.20E+04 $^-$	5.25E+04
	Std	2.35E+04	2.25E+04	2.74E+04	5.27E+03	6.85E+03	1.33E+04
f_4	Mean	3.54E+01 $^-$	2.59E+03 $^+$	7.62E+01 $^-$	7.20E+01 $^-$	1.07E+02 $^+$	7.86E+01
	Std	2.75E+01	3.80E+03	1.80E+01	1.70E+01	1.19E+01	2.13E+01
f_5	Mean	8.21E+01 $^+$	1.81E+02 $^+$	4.22E+01 $^-$	7.62E+01 $^+$	6.21E+01 $^+$	4.63E+01
	Std	1.12E+01	5.13E+01	7.57E+00	1.98E+01	7.81E+00	7.32E+00
f_6	Mean	7.22E-10 $^+$	2.58E+01 $^+$	1.43E-07 $^+$	2.29E-03 $^+$	1.14E-13 $^+$	0.00E+00
	Std	8.03E-10	1.87E+01	6.82E-07	8.03E-04	0.00E+00	0.00E+00
f_7	Mean	9.97E+01 $^+$	3.50E+02 $^+$	6.73E+01 $^-$	1.24E+02 $^+$	8.72E+01 $^+$	7.37E+01
	Std	1.01E+01	1.11E+02	5.95E+00	4.70E+01	9.36E+00	7.60E+00
f_8	Mean	9.17E+01 $^+$	1.75E+02 $^+$	4.47E+01 $^-$	7.70E+01 $^+$	5.58E+01 $^+$	5.28E+01
	Std	1.37E+01	5.46E+01	7.67E+00	1.63E+01	9.13E+00	9.06E+00
f_9	Mean	7.16E+02 $^+$	6.62E+03 $^+$	4.61E+01 $^+$	3.22E+02 $^+$	3.16E+01 $^+$	4.44E-01
	Std	3.95E+02	2.93E+03	3.90E+01	4.46E+02	3.30E+01	6.48E-01
f_{10}	Mean	2.22E+03 $^+$	5.98E+03 $^+$	1.98E+03 $^-$	1.97E+03 $^-$	2.35E+03 $^+$	2.08E+03
	Std	2.66E+02	8.29E+02	2.80E+02	4.36E+02	2.99E+02	3.22E+02
f_{11}	Mean	2.09E+02 $^+$	8.76E+02 $^+$	1.58E+03 $^+$	9.59E+01 $^+$	8.48E+01 $^+$	4.67E+01
	Std	1.11E+02	1.87E+03	1.24E+03	2.01E+01	6.03E+01	2.98E+01
f_{12}	Mean	5.46E+05 $^+$	1.22E+08 $^+$	1.29E+06 $^+$	9.52E+05 $^+$	9.98E+05 $^+$	2.47E+05
	Std	2.99E+05	3.56E+08	9.02E+05	5.25E+05	6.54E+05	2.24E+05
f_{13}	Mean	9.18E+03 $^-$	3.14E+05 $^+$	2.64E+04 $^+$	1.72E+04 $^+$	1.43E+04 $^-$	1.54E+04
	Std	6.39E+03	2.07E+06	2.42E+04	1.37E+04	8.36E+03	1.68E+04
f_{14}	Mean	4.92E+04 $^+$	4.81E+03 $^-$	2.60E+05 $^+$	2.29E+05 $^+$	2.50E+05 $^+$	3.03E+04
	Std	3.77E+04	3.38E+04	1.89E+05	1.92E+05	2.10E+05	2.74E+04
f_{15}	Mean	1.84E+03 $^-$	9.46E+06 $^+$	1.11E+04 $^+$	2.24E+03 $^-$	1.10E+03 $^-$	9.25E+03
	Std	1.86E+03	6.28E+07	1.06E+04	1.62E+03	1.14E+03	1.13E+04
f_{16}	Mean	6.14E+02 $^+$	8.73E+02 $^+$	6.84E+02 $^+$	6.24E+02 $^+$	6.49E+02 $^+$	4.65E+02
	Std	1.54E+02	4.03E+02	1.68E+02	2.40E+02	1.50E+02	1.87E+02
f_{17}	Mean	2.02E+02 $^+$	2.78E+02 $^+$	1.62E+02 $^+$	1.94E+02 $^+$	1.34E+02 $^+$	1.22E+02
	Std	8.70E+01	2.63E+02	9.67E+01	1.37E+02	7.26E+01	9.67E+01
f_{18}	Mean	1.77E+05 $^-$	2.91E+06 $^+$	3.90E+05 $^+$	3.09E+05 $^+$	1.23E+05 $^-$	1.84E+05
	Std	1.00E+05	5.20E+06	2.27E+05	1.92E+05	8.14E+04	1.22E+05
f_{19}	Mean	1.58E+03 $^-$	5.02E+07 $^+$	1.29E+04 $^-$	3.30E+03 $^-$	3.60E+03 $^-$	1.29E+04
	Std	1.38E+03	2.12E+08	1.10E+04	2.33E+03	2.52E+03	1.66E+04
f_{20}	Mean	2.70E+02 $^+$	5.20E+02 $^+$	2.23E+02 $^+$	2.92E+02 $^+$	1.57E+02 $^+$	1.21E+02
	Std	8.96E+01	2.11E+02	1.02E+02	1.50E+02	4.65E+01	7.26E+01
f_{21}	Mean	2.65E+02 $^+$	3.29E+02 $^+$	2.47E+02 $^+$	2.51E+02 $^+$	2.47E+02 $^+$	2.39E+02
	Std	6.31E+01	3.63E+01	9.95E+00	1.03E+01	7.29E+00	4.28E+01
f_{22}	Mean	6.07E+02 $^+$	1.70E+03 $^+$	6.67E+02 $^+$	1.48E+02 $^-$	1.00E+02 $^-$	2.39E+02
	Std	1.09E+03	1.35E+03	9.91E+02	3.40E+02	2.43E-13	5.65E+02
f_{23}	Mean	4.21E+02 $^+$	5.05E+02 $^+$	4.03E+02 $^+$	4.07E+02 $^+$	3.96E+02 $^-$	4.02E+02
	Std	2.66E+01	3.38E+01	7.96E+00	1.07E+01	8.70E+00	1.06E+01
f_{24}	Mean	4.80E+02 $^-$	7.66E+02 $^+$	5.29E+02 $^+$	5.34E+02 $^+$	5.00E+02 $^+$	4.85E+02
	Std	1.88E+02	2.87E+02	2.90E+01	1.94E+01	1.91E+01	7.05E+01
f_{25}	Mean	3.85E+02 $^-$	5.97E+02 $^+$	3.86E+02 $^-$	3.86E+02 $^-$	3.97E+02 $^+$	3.87E+02
	Std	1.08E+00	1.13E+02	1.28E+00	3.03E+00	1.46E+01	7.52E-01
f_{26}	Mean	4.31E+02 $^-$	2.88E+03 $^+$	1.56E+03 $^+$	1.25E+03 $^-$	5.84E+02 $^-$	1.36E+03
	Std	5.35E+02	2.01E+03	2.24E+02	5.84E+02	6.32E+02	5.47E+02
f_{27}	Mean	5.13E+02 $^+$	5.45E+02 $^+$	5.11E+02 $^+$	5.04E+02 $^+$	5.11E+02 $^+$	5.02E+02
	Std	4.51E+00	1.99E+01	5.43E+00	5.32E+00	5.50E+00	6.76E+00
f_{28}	Mean	4.02E+02 $^-$	1.39E+03 $^+$	4.09E+02 $^-$	4.00E+02 $^-$	4.29E+02 $^+$	4.09E+02
	Std	2.80E+00	9.33E+02	9.84E+00	4.73E+00	1.37E+01	6.08E+00
f_{29}	Mean	6.14E+02 $^+$	9.08E+02 $^+$	6.00E+02 $^+$	5.45E+02 $^+$	5.12E+02 $^+$	5.11E+02
	Std	7.83E+01	3.31E+02	8.17E+01	6.93E+01	6.28E+01	6.51E+01
f_{30}	Mean	7.17E+03 $^+$	1.11E+05 $^+$	1.14E+04 $^+$	5.27E+03 $^-$	5.65E+03 $^-$	7.05E+03
	Std	1.84E+03	4.36E+05	5.36E+03	2.21E+03	1.84E+03	4.17E+03
Wilcoxon	+/=/-	19/0/10	28/0/1	21/0/8	20/0/9	19/0/10	

Table 3.6: Comparison between ABC_RL and other ABC variants with $D = 50$

Function		ABC	sdABC	ARABC	ILTD_ABC	MGABC	ABC_RL
f_1	Mean	1.76E+03 $^-$	3.18E+10 $^+$	5.11E+03 $^+$	6.13E+03 $^+$	4.55E+02 $^-$	3.87E+03
	Std	1.45E+03	1.75E+10	5.75E+03	9.25E+03	5.47E+02	6.14E+03
f_3	Mean	2.14E+05 $^+$	1.48E+05 $^-$	2.94E+05 $^+$	1.23E+05 $^-$	1.16E+05 $^-$	1.83E+05
	Std	3.78E+04	3.33E+04	3.04E+04	1.20E+04	1.19E+04	2.62E+04
f_4	Mean	3.87E+01 $^-$	4.62E+03 $^+$	7.03E+01 $^-$	4.56E+01 $^-$	9.41E+01 $^+$	8.00E+01
	Std	1.51E+01	3.72E+03	3.27E+01	2.47E+01	3.60E+01	3.98E+01
f_5	Mean	1.94E+02 $^+$	4.24E+02 $^+$	9.20E+01 $^-$	2.27E+02 $^+$	1.52E+02 $^+$	1.13E+02
	Std	2.29E+01	9.59E+01	1.05E+01	3.94E+01	2.05E+01	1.57E+01
f_6	Mean	6.72E-10 $^+$	4.23E+01 $^+$	7.02E-07 $^+$	8.34E-03 $^+$	2.07E-13 $^+$	3.34E-14
	Std	7.81E-10	1.81E+01	4.16E-06	2.51E-02	4.38E-14	5.23E-14
f_7	Mean	2.12E+02 $^+$	6.82E+02 $^+$	1.30E+02 $^-$	3.79E+02 $^+$	2.02E+02 $^+$	1.53E+02
	Std	1.91E+01	1.40E+02	1.07E+01	1.60E+02	1.75E+01	1.41E+01
f_8	Mean	2.06E+02 $^+$	4.35E+02 $^+$	9.19E+01 $^-$	2.28E+02 $^+$	1.47E+02 $^+$	1.16E+02
	Std	2.06E+01	8.31E+01	1.37E+01	5.03E+01	1.98E+01	1.61E+01
f_9	Mean	5.41E+03 $^+$	2.39E+04 $^+$	2.88E+02 $^+$	3.77E+03 $^+$	5.28E+02 $^+$	9.05E+01
	Std	1.55E+03	9.37E+03	1.75E+02	2.34E+03	3.53E+02	9.58E+01
f_{10}	Mean	4.21E+03 $^+$	1.07E+04 $^+$	3.77E+03 $^-$	4.15E+03 $^+$	4.36E+03 $^+$	3.95E+03
	Std	3.30E+02	2.79E+03	2.99E+02	6.14E+02	3.54E+02	4.42E+02
f_{11}	Mean	7.13E+02 $^+$	4.87E+03 $^+$	4.39E+03 $^+$	1.62E+03 $^+$	8.84E+02 $^+$	9.06E+01
	Std	4.87E+02	4.58E+03	2.57E+03	5.44E+02	5.39E+02	2.70E+01
f_{12}	Mean	3.33E+06 $^+$	5.90E+09 $^+$	6.35E+06 $^+$	1.75E+06 $^-$	3.14E+06 $^+$	1.88E+06
	Std	1.43E+06	1.21E+10	2.55E+06	7.12E+05	1.36E+06	1.17E+06
f_{13}	Mean	5.06E+03 $^-$	4.28E+08 $^+$	1.97E+04 $^+$	2.85E+03 $^-$	1.41E+03 $^-$	9.26E+03
	Std	3.35E+03	2.20E+09	1.66E+04	3.59E+03	1.60E+03	1.16E+04
f_{14}	Mean	7.10E+05 $^+$	6.00E+04 $^-$	1.51E+06 $^+$	1.13E+06 $^+$	3.06E+06 $^+$	1.18E+05
	Std	5.36E+05	2.86E+05	1.15E+06	7.38E+05	1.55E+06	9.11E+04
f_{15}	Mean	9.19E+03 $^+$	8.20E+08 $^+$	1.14E+04 $^+$	1.46E+04 $^+$	1.59E+04 $^+$	8.14E+03
	Std	5.51E+03	1.68E+09	6.14E+03	6.50E+03	3.68E+03	7.87E+03
f_{16}	Mean	1.27E+03 $^+$	1.81E+03 $^+$	1.26E+03 $^+$	8.99E+02 $^-$	9.32E+02 $^-$	1.14E+03
	Std	2.40E+02	4.13E+02	1.93E+02	2.85E+02	1.96E+02	2.78E+02
f_{17}	Mean	9.36E+02 $^+$	1.66E+03 $^+$	8.09E+02 $^+$	9.92E+02 $^+$	8.20E+02 $^+$	7.37E+02
	Std	1.64E+02	1.14E+03	1.86E+02	2.71E+02	2.18E+02	1.81E+02
f_{18}	Mean	9.38E+05 $^+$	2.48E+05 $^-$	2.88E+06 $^+$	1.45E+06 $^+$	2.21E+06 $^+$	6.41E+05
	Std	4.90E+05	2.36E+05	1.47E+06	5.63E+05	9.69E+05	5.11E+05
f_{19}	Mean	7.47E+03 $^-$	1.46E+05 $^+$	1.62E+04 $^+$	1.82E+04 $^+$	1.61E+04 $^+$	1.46E+04
	Std	2.99E+03	4.90E+05	8.10E+03	1.13E+04	4.28E+03	1.31E+04
f_{20}	Mean	7.12E+02 $^+$	1.35E+03 $^+$	6.89E+02 $^+$	5.63E+02 $^+$	4.44E+02 $^-$	5.23E+02
	Std	1.81E+02	4.14E+02	1.49E+02	2.14E+02	1.63E+02	1.71E+02
f_{21}	Mean	4.12E+02 $^+$	5.23E+02 $^+$	3.05E+02 $^-$	3.52E+02 $^+$	3.23E+02 $^-$	3.28E+02
	Std	2.23E+01	6.04E+01	1.32E+01	2.09E+01	1.47E+01	1.39E+01
f_{22}	Mean	4.65E+03 $^-$	8.55E+03 $^+$	4.09E+03 $^-$	4.88E+03 $^+$	2.65E+03 $^-$	4.75E+03
	Std	1.54E+03	4.26E+03	1.06E+03	1.77E+03	2.45E+03	7.99E+02
f_{23}	Mean	6.48E+02 $^+$	8.53E+02 $^+$	5.51E+02 $^-$	5.76E+02 $^+$	5.46E+02 $^-$	5.68E+02
	Std	4.82E+01	7.45E+01	1.49E+01	2.43E+01	1.70E+01	1.97E+01
f_{24}	Mean	1.01E+03 $^+$	1.12E+03 $^+$	7.97E+02 $^+$	8.21E+02 $^+$	7.48E+02 $^+$	6.85E+02
	Std	6.34E+01	3.64E+02	3.97E+01	4.57E+01	3.35E+01	3.11E+01
f_{25}	Mean	5.08E+02 $^-$	4.22E+03 $^+$	5.12E+02 $^-$	5.42E+02 $^+$	5.97E+02 $^+$	5.16E+02
	Std	1.99E+01	2.66E+03	1.78E+01	3.18E+01	1.02E+01	2.24E+01
f_{26}	Mean	1.92E+03 $^-$	6.32E+03 $^+$	2.38E+03 $^-$	1.65E+03 $^-$	6.20E+02 $^-$	2.59E+03
	Std	1.54E+03	2.32E+03	1.65E+02	1.54E+03	1.13E+03	1.76E+02
f_{27}	Mean	6.49E+02 $^+$	8.92E+02 $^+$	6.41E+02 $^+$	5.92E+02 $^+$	6.20E+02 $^+$	5.80E+02
	Std	3.03E+01	1.34E+02	3.42E+01	1.69E+01	1.96E+01	3.75E+01
f_{28}	Mean	4.85E+02 $^-$	2.54E+03 $^+$	4.84E+02 $^-$	4.95E+02 $^+$	5.44E+02 $^+$	4.92E+02
	Std	1.19E+01	1.64E+03	1.68E+01	1.99E+01	1.56E+01	1.90E+01
f_{29}	Mean	1.03E+03 $^+$	1.96E+03 $^+$	8.59E+02 $^+$	6.60E+02 $^-$	8.18E+02 $^+$	7.62E+02
	Std	1.54E+02	3.19E+03	1.64E+02	1.62E+02	1.78E+02	1.34E+02
f_{30}	Mean	7.23E+05 $^-$	1.13E+08 $^+$	7.97E+05 $^-$	8.52E+05 $^-$	8.76E+05 $^-$	9.74E+05
	Std	4.99E+04	4.22E+08	9.45E+04	1.05E+05	7.15E+04	2.70E+05
Wilcoxon	+/=/-	20/0/9	26/0/3	17/0/12	21/0/8	19/0/10	

ABC_RL gains better results than ARABC on 17 problems. And comparing to ILTD_ABC, ABC_RL achieves smaller errors on 21 problems while ABC_RL surpasses MGABC on 19 problems.

Furthermore, the Friedman tests are carried out, and the average rankings of each method with $D = 10, 30$, and 50 are shown in Figure 3.3. It has been demonstrated that the proposed ABC_RL algorithm achieves the best average ranking in all the comparisons. For 10-dimensional problems, the original ABC is the second-best followed by MGABC. As for $D = 30$ and 50, MGABC obtains smaller rankings than other competitors except for ABC_RL. Therefore, it can be concluded that ABC_RL has the best overall performance among the six ABC algorithms.

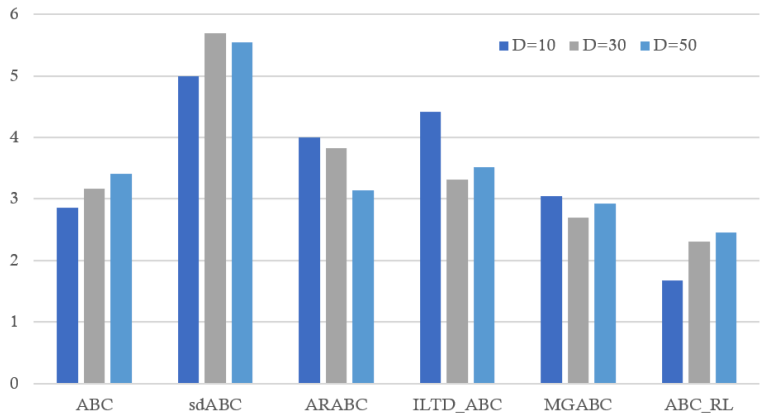


Fig. 3.3. Average rankings of ABC algorithms by Friedman tests with $D = 10, 30$ and 50

3.4.4 Effectiveness of the proposed strategies

In order to further validate the proposed ABC_RL algorithm, the effectiveness of each improvement is tested and analyzed in this part. Hence, two variants of ABC_RL are constructed. ABC_RL$_{eq}$ adopted only the modified search equations, i.e., (3.5) and (3.8), with scale factors following heavy-tailed distribution. Note that these equations are one-dimensional because RL was not introduced to vary the nb_{up}. Meanwhile, ABC_RL$_{QL}$ utilized the Q-learning-based strategy to enlarge and adjust the nb_{up} in employed bee phase. Its solution search equation is the same as the original Eq.(1.2).

3. REINFORCEMENT LEARNING BASED ABC ALGORITHM (ABC_RL)

These variants are compared with the standard ABC aiming at verifying the effectiveness of the proposed strategies. Experimental results on 29 benchmarks over 51 independent executions are shown in Table 3.7. The best results of each function are highlighted in **boldface**. And the last row shows the Wilcoxon test results of comparing these two variants with basic ABC algorithm. The average rankings given by Friedman tests are drawn in Figure 3.4.

Table 3.7: Effectiveness of each modification of ABC_RL on benchmarks with D= 10, 30 and 50

	D=10			D=30			D=50		
Function	ABC	ABC_RL$_{eq}$	ABC_RL$_{QL}$	ABC	ABC_RL$_{eq}$	ABC_RL$_{QL}$	ABC	ABC_RL$_{eq}$	ABC_RL$_{QL}$
f_1	**2.88E+02**	6.87E+02	3.21E+02	**2.15E+02**	5.10E+02	5.03E+02	1.76E+03	3.65E+03	**1.45E+03**
f_3	6.66E+02	**1.03E+02**	2.16E+02	7.29E+04	6.04E+04	**6.01E+04**	2.14E+05	2.04E+05	**1.86E+05**
f_4	**2.28E-01**	2.26E+00	4.91E-01	**3.54E+01**	5.66E+01	5.03E+01	**3.87E+01**	4.04E+01	6.05E+01
f_5	7.71E+00	**4.51E+00**	7.34E+00	8.21E+01	**4.58E+01**	7.75E+01	1.94E+02	**1.11E+02**	1.89E+02
f_6	3.35E-09	**0.00E+00**	5.32E-11	7.22E-10	**0.00E+00**	7.74E-07	6.72E-10	**2.01E-14**	1.30E-05
f_7	1.75E+01	**1.44E+01**	1.74E+01	9.97E+01	**7.22E+01**	1.01E+02	2.12E+02	**1.47E+02**	2.10E+02
f_8	8.00E+00	**5.10E+00**	8.06E+00	9.17E+01	**5.20E+01**	8.27E+01	2.06E+02	**1.11E+02**	1.81E+02
f_9	6.04E-03	**5.21E-08**	9.10E-08	7.16E+02	**5.16E+01**	5.34E+02	5.41E+03	**5.19E+02**	4.42E+03
f_{10}	2.50E+02	**1.37E+02**	2.83E+02	2.22E+03	**1.90E+03**	2.42E+03	4.21E+03	**3.79E+03**	4.39E+03
f_{11}	4.05E+00	**2.36E+00**	3.26E+00	2.09E+02	8.16E+01	**5.10E+01**	7.13E+02	4.49E+02	**1.33E+02**
f_{12}	4.29E+04	**2.88E+04**	4.45E+04	5.46E+05	**4.90E+05**	5.37E+05	3.33E+06	**2.78E+06**	3.82E+06
f_{13}	5.98E+02	5.14E+02	**4.13E+02**	9.18E+03	1.63E+04	**6.01E+03**	5.06E+03	6.50E+03	**1.36E+03**
f_{14}	1.72E+02	2.89E+02	**8.39E+01**	4.92E+04	5.74E+04	**2.84E+04**	7.10E+05	4.03E+05	**3.60E+05**
f_{15}	9.76E+01	1.40E+02	**4.12E+01**	1.84E+03	7.51E+03	**7.08E+02**	9.19E+03	8.45E+03	**5.57E+03**
f_{16}	8.06E+00	**1.40E+00**	2.91E+00	6.14E+02	**5.64E+02**	5.99E+02	1.27E+03	**1.15E+03**	1.24E+03
f_{17}	2.35E+00	**8.20E-01**	2.91E+00	2.02E+02	1.53E+02	**1.39E+02**	9.36E+02	**7.31E+02**	8.49E+02
f_{18}	**1.53E+03**	1.86E+03	1.60E+03	**1.77E+05**	1.89E+05	1.93E+05	9.38E+05	9.79E+05	**8.97E+05**
f_{19}	9.75E+01	1.80E+02	**3.77E+01**	1.58E+03	6.20E+03	**1.21E+03**	**7.47E+03**	1.08E+04	7.85E+03
f_{20}	2.84E-01	**1.84E-02**	2.12E-01	2.70E+02	2.00E+02	**1.63E+02**	7.12E+02	**5.93E+02**	6.28E+02
f_{21}	1.10E+02	1.16E+02	**1.09E+02**	2.65E+02	**2.29E+02**	2.40E+02	4.12E+02	**3.16E+02**	3.92E+02
f_{22}	7.54E+01	**5.72E+01**	6.26E+01	6.07E+02	**2.31E+02**	4.38E+02	4.65E+03	**4.16E+03**	4.86E+03
f_{23}	3.02E+02	3.08E+02	**2.97E+02**	4.21E+02	**4.03E+02**	4.20E+02	6.48E+02	**5.68E+02**	6.27E+02
f_{24}	**1.04E+02**	1.09E+02	1.19E+02	4.80E+02	4.91E+02	**4.67E+02**	1.01E+03	8.04E+02	**7.40E+02**
f_{25}	**2.19E+02**	3.64E+02	2.91E+02	**3.85E+02**	3.87E+02	3.86E+02	**5.08E+02**	5.17E+02	5.16E+02
f_{26}	1.07E+02	1.55E+02	**9.72E+01**	**4.31E+02**	8.52E+02	4.77E+02	**1.92E+03**	2.51E+03	2.30E+03
f_{27}	3.95E+02	3.93E+02	**3.91E+02**	5.13E+02	**5.10E+02**	5.11E+02	6.49E+02	**6.32E+02**	6.40E+02
f_{28}	**2.77E+02**	2.98E+02	2.91E+02	**4.02E+02**	4.09E+02	4.09E+02	**4.85E+02**	4.96E+02	4.96E+02
f_{29}	2.53E+02	**2.50E+02**	2.64E+02	6.14E+02	**5.49E+02**	6.03E+02	1.03E+03	**8.18E+02**	8.67E+02
f_{30}	1.87E+04	**9.41E+03**	1.22E+04	7.17E+03	7.86E+03	**5.08E+03**	**7.23E+05**	7.32E+05	7.31E+05
Wilcoxon (+/=/-) v.s. ABC		17/0/12	18/0/11		17/0/12	20/0/9		20/0/9	19/0/10

Firstly, in Table 3.7, the comparison between variant ABC_RL$_{eq}$ and the basic ABC is concerned. It is observed that ABC_RL$_{eq}$ achieves significant improvement in functions f_6 and f_9. According to the Wilcoxon test results, for D = 10 and 30, ABC_RL$_{eq}$ obtains smaller errors than ABC does on 17 benchmarks. As for D = 50, it outperforms the basic ABC algorithm on 20 functions. In this case, the effectiveness of utilizing improved search equations with heavy-tailed based scale factors can be verified.

Secondly, in order to demonstrate the effect of RL strategy, ABC_RL$_{QL}$ is compared with ABC algorithm. It can be found that, for all the dimension cases, ABC_RL$_{QL}$ achieves better results on more than half of the 29 functions. So the

algorithm's performance can be improved by using RL method to enlarge and adjust the nb_{up}. Moreover, similar conclusion can be derived from the number of best results in boldface.

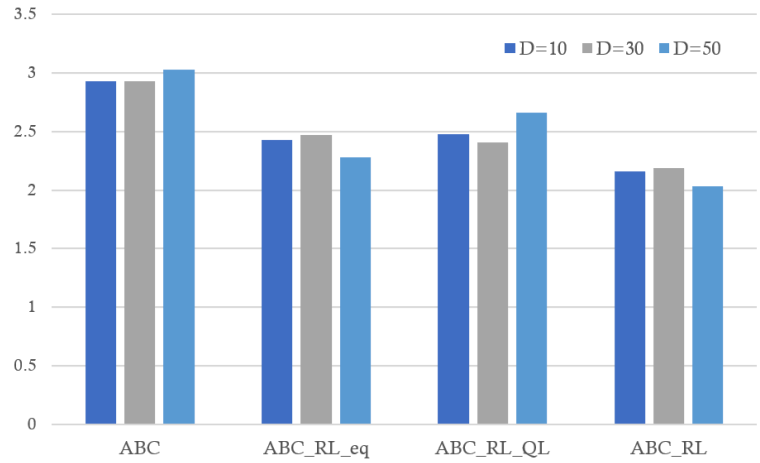


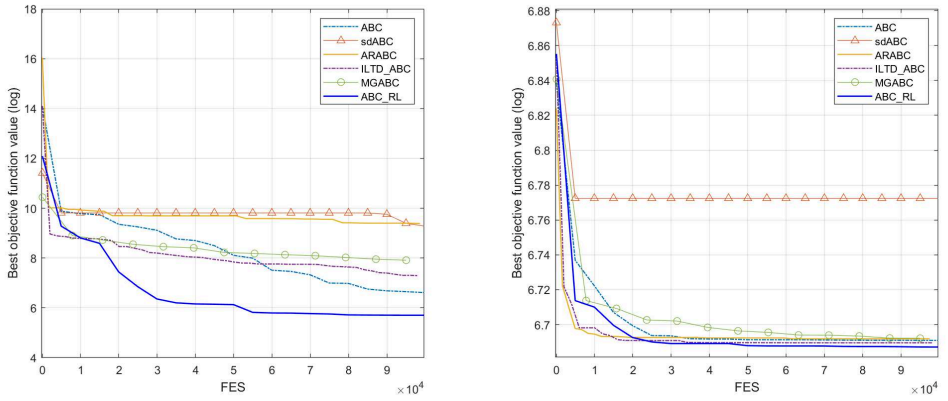
Fig. 3.4. Friedman test results for effectiveness demonstration of modifications

Furthermore, these two versions are compared with the final ABC_RL algorithm as well. In Figure 3.4, the lower the bar is, the better the ranking is. For all the dimension cases, ABC_RL always gets the best ranking. Meanwhile, the rankings of ABC_RL$_{eq}$ and ABC_RL$_{QL}$ are both better than those of ABC. Hence, the effectiveness of the proposed strategies can be proved. And it can be concluded that these strategies work better together than they do individually.

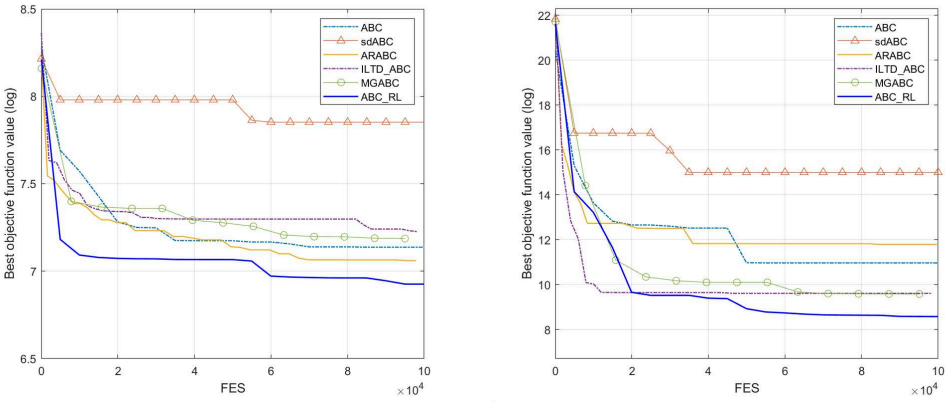
3.4.5 Convergence behavior analysis

The convergence curves of compared algorithms were plotted to fully compare their performance. The convergence speed of solving different types of problems with $D = 10$ can be observed in Figure 3.5. Note that, the values of $log(f(\cdot))$ are presented because the objective function values are too large in the earlier stage of searching process.

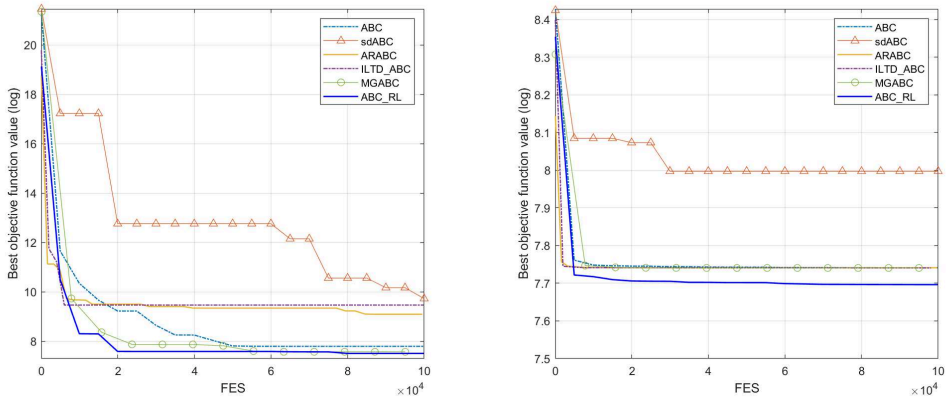
In Figure 3.5, (a) presents the convergence curves of ABC algorithms on unimodal function f_3. Note that, the results of solving unimodal functions can indicate the exploitation ability of executed algorithm as there is a unique optimum. And it can be found that, ABC_RL is able to achieve the best accuracy while its convergence rate is competitive to other competitors. Meanwhile, figures (b) and (c) (f_8 and f_{12}) show the cases of multimodal problems. A similar



(a) f_3 **(b)** f_8



(c) f_{10} **(d)** f_{12}



(e) f_{18} **(f)** f_{22}

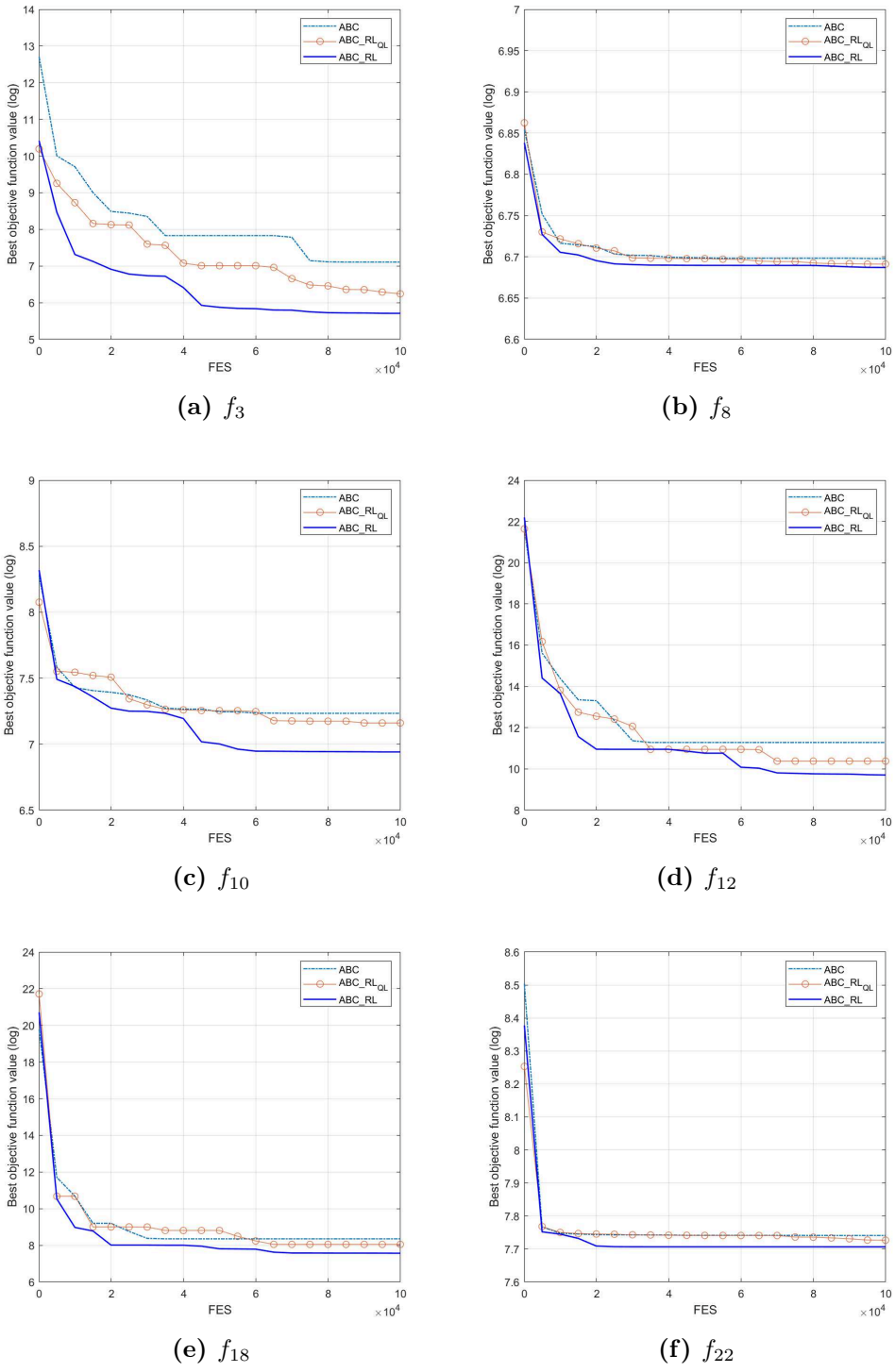
Fig. 3.5. The convergence performance of ABC_RL and compared ABC algorithms

conclusion can be obtained from (c), where ABC_RL converges rapidly to the smallest value among the concerned methods. And in (b), the convergence curves are close to each other except for the curve of sdABC. (d) and (e) indicate how the algorithms converge in solving hybrid functions. In these two figures, the advantages in solution accuracy of ABC_RL can be found. At the same time, its convergence rate is competitive to the others. The curve of ILTD_ABC descends the fastest in (d). Furthermore, as for composition function f_{22} displayed in (e), the performance of the compared algorithms is similar except for sdABC. And ABC_RL is able to achieve a better result in terms of solution precision.

In addition, as mentioned at the beginning of this work, it is worth studying the parameter adjusting of ABC via the RL method. As the convergence rate of ABC is acknowledged to be limited, the improvement strategies should avoid decelerating the algorithm even if the main purpose is not to enhance the convergence speed. In this context, the analysis of the influence of RL on the convergence rate of ABC is meaningful. Therefore, the convergence process of variant ABC_RL$_{QL}$ is compared to that of the original ABC algorithm and the final ABC_RL algorithm as presented in Figure 3.6.

According to Figure 3.6, it can be observed that the trend of convergence curves of the three involved algorithms is basically the same, especially in figures (a), (b), and (f). In (a), the final solution obtained by variant ABC_RL$_{QL}$ is better than that of the original ABC. And the final ABC_RL achieves the best solution accuracy. When solving multimodal and hybrid functions, i.e., (b)-(e), in the initial stage, the original ABC converges slightly faster than ABC_RL$_{QL}$ because ABC_RL$_{QL}$ encounters some stagnation points. After that, their convergence processes are nearly the same in the middle stage. It is worth pointing out that ABC_RL$_{QL}$ can further find more accurate solutions in the later stage.

In this way, the impact of using the Q-learning method to adjust the parameter nb_{up} on the convergence rate can be analyzed. The above observations indicate that in complex problems, the exploring efficiency of ABC may be slightly affected by RL-based strategy only in the early stage, but better solutions can be found later with the help of RL. One possible reason for this situation is that the learning experience of RL is not enough in the early stage, so the instability of RL may affect the convergence speed of external ABC algorithm. Nonetheless, from the



(a) f_3 **(b)** f_8

(c) f_{10} **(d)** f_{12}

(e) f_{18} **(f)** f_{22}

Fig. 3.6. The influence of RL method on convergence rate of ABC algorithm

middle and late stages, RL is able to gradually provide more useful help to the algorithm.

Moreover, the convergence performance of the final ABC_RL provides strong support for utilizing the other improving strategies simultaneously. The algorithm is able to keep a nice balance between diversification and intensification via adopting the enhanced solution search equations (i.e., Eq.(3.5) and (3.8)) with RL-based parameter tuning. Thus, when all these strategies are applied together, the proposed ABC_RL can not only converge faster than basic ABC, but also reach outstanding solutions.

3.5 Conclusion

In this chapter, to improve the search efficiency of ABC, a RL-based ABC algorithm (named ABC_RL) is proposed. Different from the ABCDC in the previous chapter and many other existing ABC versions, RL is used to intelligently adjust a parameter of ABC_RL. More precisely, since the frequency of perturbations has a significant impact on the performance of algorithm, the Q-learning method is used to change the frequency by learning from the historical updating experience in employed bee phase. Moreover, the information of the global best and two neighbors is considered in the onlooker bee phase to better lead the search direction. Furthermore, random scale factors drawn from a heavy-tailed distribution are adopted in the two search strategies to enrich the algorithm's randomness.

Experiments were carried out on 29 CEC 2017 benchmark problems. ABC_RL was compared to five effective ABC algorithms in different dimension cases. The comparison results demonstrated that the proposed algorithm outperformed the other competitors in terms of solution accuracy and overall performance. This part of work also provides a novel way of incorporating RL into SIAs.

3. REINFORCEMENT LEARNING BASED ABC ALGORITHM (ABC_RL)

Chapter 4

Learning based ABC algorithm (ABCL)

Contents

4.1 Introduction

One of the primary purposes of improving the optimization algorithms is to apply them to solve problems of practical interest with the purpose of reducing cost consumption and increasing efficiency. In this context, compared to the previous chapters, this part of work keeps in mind the situations of solving real-world problems while carrying out the improvement of the ABC algorithm. These days, there exist many on-line optimization problems, such as on-line robot path planning. And the time required for the algorithm to compute during the process has become another essential factor to be considered. That is to say, we aim at finding the improvement strategies that can ensure the solution accuracy while taking computational complexity and operation efficiency into account. The motivation for this derives from the fact that many modification strategies that are effective in resolving functional optimization problems do not always help us find the ideal answer rapidly in practical applications. Therefore, it is also meaningful to improve the algorithm's performance without overcomplicating it.

As mentioned in the previous chapters, ABC has been discovered to be excellent at diversification but not so effective in terms of intensification. This also leads to the convergence speed of ABC being relatively slow in certain problems. In order to tackle this weakness, the solution search equation of ABC needs to be enhanced since only one neighbor of uncertain quality is considered. Moreover, the purpose of scout bee phase is to avoid stagnation. More precisely, the equation for producing new solutions is adopted to replace those food source positions that have not been updated for a long time. However, the quality of the newly generated solutions can not be guaranteed since the previous search experience is ignored. This is also one of the reasons for the low convergence speed of ABC.

In addition, researchers found that the scout bee phase demonstrates is occasionally redundant in the search process (Anuar *et al.*, 2016). In this case, the scout bee phase has also been improved in some ABC variants, and some ABC algorithms have removed this component. For instance, Xiang & An (2013) utilized chaotic search in the scout bee phase in order to further enhance the algorithm's performance. Cui *et al.* (2017b) removed the scout bee phase from their proposed APABC because new solutions could be introduced via an adaptive method of population size. Moreover, our ABCDC algorithm proposed in Chapter 2 does not contain the scout bee phase either. The proposed method

for adjusting the population composition enables ABCDC to abandon the worst solutions timely. In this part, we would like to maintain the scout bee phase and try to strengthen it. More precisely, the successful search experience is concerned when producing new solutions, so that more valuable solutions can be transmitted to the subsequent iteration.

With the purpose of enhancing the exploitation ability and saving computation time efficiently, the search strategy of the teaching-learning based optimization (TLBO) algorithm was chosen to be incorporated into the proposed algorithm. It has been discovered that the TLBO algorithm converges quickly and is effective at exploitation. At the same, the onlooker bee phase of ABC is mainly responsible for exploiting locally. After the employed bees search the whole space, the onlookers will search locally around the solutions with higher quality. In this context, it is natural to attempt to embed the advantages of TLBO into the onlooker bee phase. Moreover, the way of choosing the regions to be further exploited can be made better. The basic ABC algorithm uses the roulette wheel selection method in onlooker bee phase for selecting the solutions to be updated. However, this process has the risk of time consumption. Consequently, the onlooker bee phase of the proposed algorithm is enhanced by the strategy of the learning phase of TLBO. And there is no longer a need to calculate probabilities and use the roulette selection method, which can help streamline the algorithm.

Therefore, in this chapter, enhancing the performance of ABC while avoiding it becoming too complex is investigated. In this context, a learning-based ABC (ABCL) algorithm is proposed for improving the exploitation ability as well as the search efficiency. So that more energy and time can be saved when solving problems like local path planning. The proposed improvement strategies include: firstly, the global best solution is adopted in the employed bee phase and scout bee phase to guide the swarm in a promising search direction. Secondly, the learning phase of TLBO is embedded in the onlooker bee phase to improve the exploitation ability and simplify the computational complexity. Then, the performance of the proposed algorithm has been verified through numerical optimization problems.

The rest of this chapter is organized as follows. The standard TLBO algorithm is introduced in section 4.2. In section 4.3, the proposed algorithm is explained

in details. In section 4.4, the experimental studies are presented. Finally, the conclusion is given in the section 4.5.

Remark 4.1 *The ultimate objective is to solve on-line multi-robot path planning (MRPP) problems more effectively. And the implementation method and simulation tests are given in Chapter 6 along with other applications.*

4.2 Preliminaries

4.2.1 Teaching-learning based optimization (TLBO) algorithm

TLBO is also a swarm-based stochastic optimization algorithm that was proposed in 2011. It was inspired by the process of teaching–learning inside classrooms (Rao *et al.*, 2011). The search process is achieved through two phases iteratively, namely the teaching phase and the learning phase. It has been found that TLBO has advantages like fast convergence rate, simple concept, and outstanding effectiveness (Zou *et al.*, 2019). Learners of TLBO are the individuals of the meta-heuristic algorithms, and each learner represents a candidate solution to the concerned problem. For the purpose of improving the class's knowledge level, learners learn from the teacher in the teaching phase, while they enhance their states by learning from others in the learning phase. The principle process of TLBO is explained as follows.

4.2.1.1 Initialization

Same initialization method is utilized in TLBO as the Eq.(1.1) of ABC algorithm. For solving a $D-$dimensional optimization problem, $x_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,D})$ is the i^{th} learner and $i = 1, \cdots, NP$ where NP is the population size.

4.2.1.2 Teaching phase

In the teaching phase, the learner with the best fitness value is appointed as the $Teacher$. The learners are improved via the following equations.

$$tf = round(1 + rand(0, 1)), \tag{4.1}$$

$$x_i^{new} = x_i + rand \times (Teacher - tf \times Mean), \tag{4.2}$$

where x_i^{new} is the new state of the i^{th} learner. $Teacher$ is the learner with the best fitness value and $Mean$ is the average state of the population. tf is a teaching factor while $rand$ is a vector with random values in the range of $[0, 1]$. The better learner is then selected greedily and enters the learning phase.

4.2.1.3 Learning phase

Then in the learning phase, learners use the Eq.(4.3) to improve their knowledge levels.

$$x_i^{new} = \begin{cases} x_i + rand(0,1) \times (x_k - x_i) & if\ f(x_k) < f(x_i), \\ x_i + rand(0,1) \times (x_i - x_k) & otherwise, \end{cases} \tag{4.3}$$

where x_i^{new} is the new updated learner of x_i while x_k is randomly selected from the rest of the class. And a random vector in range of $[0, 1]$ is generated as the scale factor of Eq.(4.3).

Then, the fitness values of the two learners $f(x_i)$ and $f(v_i)$ are compared together, the same selection method is adopted. After this, the search process enters into the teaching phase again and will repeat until the termination conditions are reached. The pseudo-code of the TLBO algorithm is shown in Algorithm 11.

4.3 Proposed ABCL algorithm

In order to generate high-quality solutions effectively, the improvement strategies are presented in this part. Firstly, a modified solution search equation based on the global best individual is adopted in employed bee phase of the proposed ABCL algorithm. Meanwhile, considering the strong exploitation ability of TLBO, its learning strategy is incorporated into the onlooker bee phase. And the global best information is involved in scout bee phase as well.

4.3.1 Enhanced employed bee phase

In original ABC algorithm, the new candidate solutions are generated via Eq.(1.2) with a randomly selected neighbor x_k and a random number $\theta_{i,j}$. Hence, this search strategy is good at exploration thanks to the randomness (Zhu & Kwong,

Algorithm 11 Pseudo-code of TLBO algorithm

```
Initialize the learners with Eq.(1.1) and evaluate the initial population
repeat
    % Teaching phase %
    for i = 1 → NP do
        Generate factor tf with Eq.(4.1)
        Produce new learner x_i^new with Eq.(4.2)
        if f(x_i^new) < f(x_i) then
            Replace x_i with x_i^new
        else
            Learner x_i remains the same
        end if
    end for
    % Learning phase %
    for i = 1 → NP do
        Select a learner k ≠ i randomly
        Update x_i with Eq.(4.3)
        if f(x_i^new) < f(x_i) then
            Replace x_i with x_i^new
        else
            Learner x_i remains the same
        end if
    end for
    Update the Teacher and the Mean
until the termination condition is reached.
```

2010). Based on the literature, being inspired by PSO, the global best information has been widely embedded into ABC variants (Gao *et al.*, 2014; Wang *et al.*, 2020; Xiao *et al.*, 2021; Zhu & Kwong, 2010). In this way, the colony can be guided in a hopeful direction while the algorithm's exploitation ability can be improved.

In addition, limited useful information is learned from the swarm since only one individual is considered in Eq.(1.2). To this point, various modifications have been investigated. One of the most effective modifications is inspired by the evolutionary operators of the Differential Evolution (DE) algorithm. The mutation and crossover strategies can not only allow the ABC to learn more information from the population but also enlarge the dimensions being updated each time (Banharnsakun *et al.*, 2011; Chen *et al.*, 2019b; Gao *et al.*, 2012, 2016; Liang *et al.*, 2017).

Based on the discussions above, an improved search strategy adopting both

of the aforementioned ideas is proposed in Eq.(4.4) as

$$v_{i,j} = x_{i,j} + \rho_{i,j} \times (gbest_j - x_{i,j}) + \mu_{i,j} \times (x_{k_1,j} - x_{k_2,j}), \tag{4.4}$$

where $v_{i,j}$ is the updated j^{th} element of the i^{th} food source. And $gbest_j$ is the j^{th} element of the global best individual. $k_1 \neq k_2 \neq i$ are selected randomly from $\{1, \cdots, SN\}$ and j is chosen from $\{1, \cdots, D\}$. The parameter $\rho_{i,j}$ is a uniform random number within the range [0, 1] while $\mu_{i,j}$ is another random number in [-0.5, 0.5].

As seen from Eq.(4.4), the global best position is utilized to lead the colony in a good direction as the scale factor $\rho_{i,j}$ is a positive random number. Meanwhile, different from the previous ABC variants with global best solution (eq. GABC (Zhu & Kwong, 2010)), the last term $\mu_{i,j} \times (x_{k_1,j} - x_{k_2,j})$ selects two other solutions. In this case, more colony information can be learned. However, considering the information may not always be favorable, the coefficient $\mu_{i,j}$ is defined within a smaller range [-0.5, 0.5].

4.3.2 Learning-based onlooker bee phase

In basic ABC, the onlooker bees are supposed to search around the qualified food sources which correspond to exploitation. It has been observed that ABC is weak in exploitation locally. At the same time, the TLBO algorithm has been found to be good at exploitation but relatively poor at exploration. Based on the complementarity of ABC and TLBO, a hybrid algorithm was proposed by Chen *et al.* (2018), which embedded the teaching phase of TLBO in employed bee phase and combined the learning phase with onlooker bee phase.

Similarly, in this work, the learning strategy of TLBO is incorporated in the onlooker bee phase so that individuals can learn interactively within the colony. Different from the work of Chen *et al.* (2018), the proposed learning-based search strategy (i.e., Eq.(4.5)) is adopted by all the individuals without the roulette wheel selection method. In original ABC and its numerous variants, the roulette wheel selection method is adopted to select the high-quality candidate solutions in onlooker bee phase. As explained in section 1.3, the selection probability of each candidate solution is proportional to its fitness value. The greater the fitness, the more likely the individual is selected. Nonetheless, this process lessens

the population diversity and may prolong the computation time. Thus, in the proposed ABCL algorithm, the calculations of fitness values, probabilities, and roulette wheel selection are dropped to avoid wasting time in path planning.

The learning-based search equation is expressed as below.

$$v_{i,j} = \begin{cases} x_{i,j} + rand \times (x_{k,j} - x_{i,j}) & if\ f(x_k) < f(x_i), \\ x_{i,j} + rand \times (x_{i,j} - x_{k,j}) & otherwise, \end{cases} \tag{4.5}$$

where $k \neq i$ are randomly selected from $\{1, \cdots, SN\}$ and $j \in \{1, \cdots, D\}$ is the randomly chosen dimension to be updated. If x_k has a smaller objective value than x_i, then the individual x_i is going to learn from x_k. Otherwise, x_i will learn knowledge by itself (Ma *et al.*, 2021). In fact, the neighbor is selected randomly in Eq.(1.2) of the original ABC. As Zhu & Kwong (2010) mentioned, the good solution and the bad solution have an equal probability of being chosen. So this update equation cannot guarantee that individuals are moving in a valuable direction. This drawback can be prevented by utilizing different equations after comparing the objective function values of x_i and the random neighbor. Furthermore, the population diversity can be maintained at the same time.

4.3.3 Enhanced scout bee phase

The scout bee phase is responsible for supervising the qualities of candidate solutions. The food sources that haven't been updated over predefined *limit* times will be abandoned. However, the equation of generating new solutions only uses the lower and upper bound values, which is a waste of known information. In other words, if the search experience can be considered when producing a novel solution, the newly generated one will have a higher possibility to be in a hopeful position. In this case, the enhanced scout bee phase is more likely to avoid returning to the explored regions blindly. Hence, the global best information is considered as well as the boundary values in the proposed Eq.(4.6).

$$x_{i,j} = gbest_j + \phi_{i,j} \times (x_{max,j} - x_{min,j}), \quad j = 1, \cdots, D, \tag{4.6}$$

where $gbest$ is the global best solution while x_{min} and x_{max} are the lower and upper bounds respectively. To avoid losing randomness, the scale factor ϕ is a uniform random number in range of $[-1, 1]$.

4.3.4 The framework of ABCL algorithm

In order to explain the entire process of proposed algorithm, the pseudo-code of ABCL is shown in Algorithm 12 while the flowchart is represented in Figure 4.1. Note that, the termination conditions is defined by the maximal number of function evaluations max_FES. In other words, the iteration continues till the number of function evaluations FES reaches the predefined max_FES.

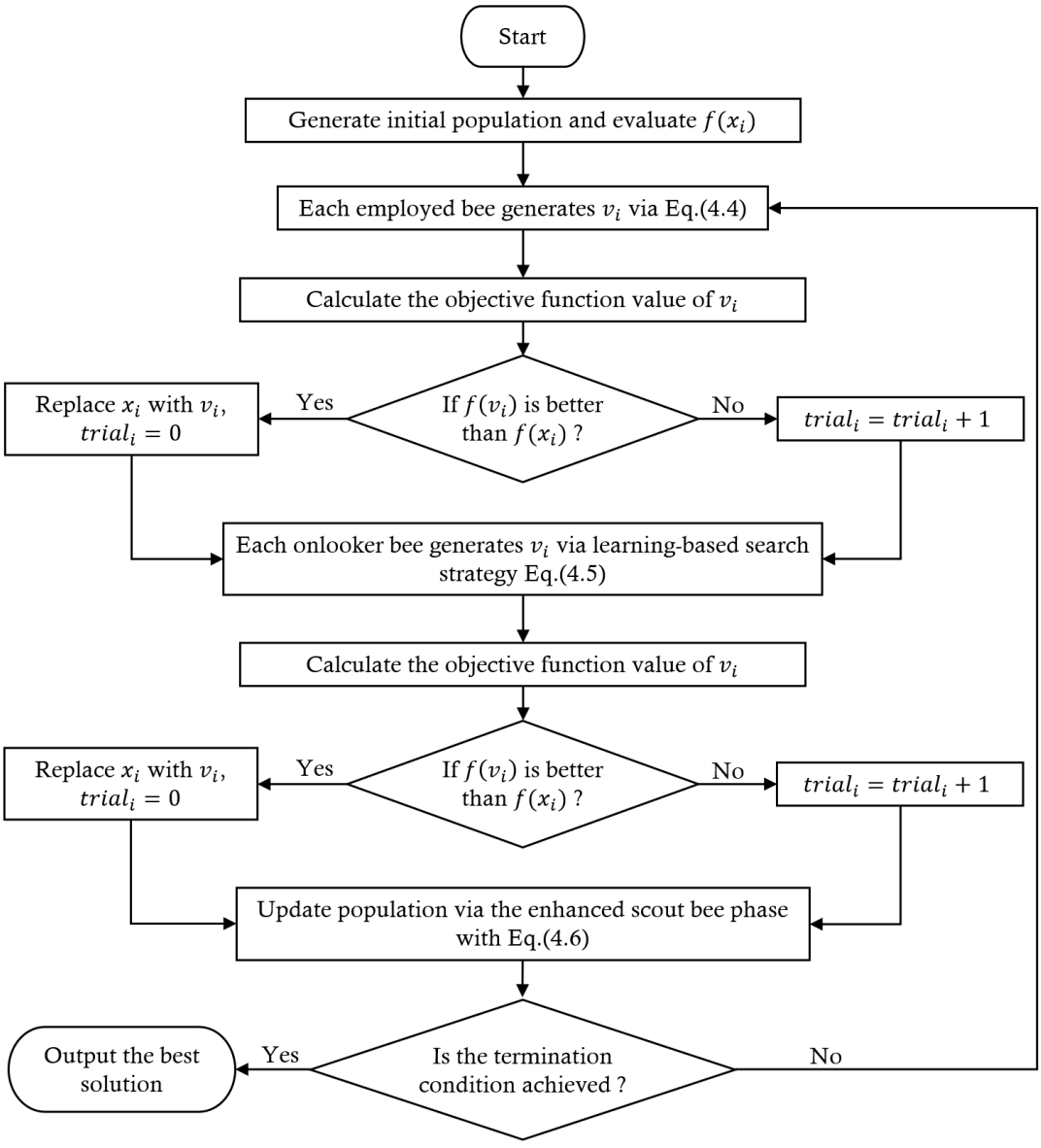


Fig. 4.1. The flowchart of ABCL algorithm

Algorithm 12 Pseudo-code of ABCL algorithm

Generate initial population x_i, $i = 1, \cdots, SN$ with Eq.(1.1)
Evaluate objective function values $f(x_i)$
$FES = SN$
while $FES \leq max\_FES$ **do**
    % `Enhanced employed bee phase` %
    for $i = 1 \rightarrow SN$ **do**
        Randomly select $k_1 \neq k_2 \neq i$ from $\{1, \cdots, SN\}$
        Randomly select dimension j from $\{1, \cdots, D\}$
        Produce v_i via Eq.(4.4)
        if $f(v_i) < f(x_i)$ **then**
            Replace x_i with v_i; $trial_i = 0$
        else
            $trial_i = trial_i + 1$
        end if
    end for
    % `Learning-based onlooker bee phase` %
    for $i = 1 \rightarrow SN$ **do**
        Randomly select $k \neq i$ from $\{1, \cdots, SN\}$
        Randomly select dimension j from $\{1, \cdots, D\}$
        Generate updated individual $v_{i,j}$ with Eq.(4.5)
        if $f(v_i) < f(x_i)$ **then**
            Replace x_i with v_i; $trial_i = 0$
        else
            $trial_i = trial_i + 1$
        end if
    end for
    $FES = FES + 2SN$
    % `Enhanced scout bee phase` %
    for $i = 1 \rightarrow SN$ **do**
        if $trial_i > limit$ **then**
            Generate a new candidate solution with Eq.(4.6)
            $trial_i = 0$
        end if
    end for
end while

4.4 Experiments on function optimization problems

In this section, experiments are carried out on a set of optimization problems to test the performance of ABCL. The proposed algorithm is compared with three ABC variants, namely the standard ABC, GABC (Zhu & Kwong, 2010), and DABC (Abbas & Ali, 2014). GABC is involved in the comparison because the global best solution was concerned in its effective search equation as well. The control parameter is set as the same as its best value of the original paper, i.e., $C = 1.5$.

4.4.1 Benchmark functions

The definitions of the benchmarks can be found in Table 4.1. Among these optimization problems, $f_1 - f_2$ and f_4 are uni-modal functions while f_3 is the discontinuous step function. f_5 is the Rosenbrock function which is a uni-modal problem when $D = 2$ and 3, but probably has multiple optima in higher dimensions (Kang *et al.*, 2011). As for the rest part, $f_6 - f_{12}$ are multi-modal functions.

Remark 4.2 *Since this algorithm is expected to perform well in practical problems, the experimental study part does not use too complicated optimization problems. Its effectiveness in solving the MRPP problem will be given in the next Chapter 6.*

4.4.2 Comparison with ABC variants

In the experiments, the swarm size is set as $SN = 30$ for all the comparative algorithms. The control parameter *limit* is defined as $SN \times D$ and $max\_FES = 5000 \times D$ with $D = 10$, 30, and 50. Each algorithm is executed 25 independent times on all the problems. Then the statistical results are calculated and compared in Table 4.2 - Table 4.4. The mean and standard deviation (std) of errors $f(X_{best}) - f(X^*)$ over the 25 runs are calculated, where X_{best} is the best solution calculated by algorithm and X^* is the exact global optimum.

In order to analyze the comparison results more comprehensively, the Wilcoxon tests are carried out to show the significance between ABCL and other concerned

Table 4.1: Benchmark functions

Function	Range	Min
$f_1(\mathbf{x}) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} x_i^2$	$[-100, 100]^D$	0
$f_2(\mathbf{x}) = \sum_{i=1}^{D} \lvert x_i \rvert^{(i+1)}$	$[-1, 1]^D$	0
$f_3(\mathbf{x}) = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	$[-100, 100]^D$	0
$f_4(\mathbf{x}) = \exp(0.5 \sum_{i=1}^{D} x_i)$	$[-10, 10]^D$	0
$f_5(\mathbf{x}) = \sum_{i=1}^{D} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-5, 10]^D$	0
$f_6(\mathbf{x}) = \sum_{i=1}^{D} [x_i^2 - 10cos(2\pi x_i) + 10]$	$[-5.12, 5.12]^D$	0
$f_7(\mathbf{x}) = \sum_{i=1}^{D} [y_i^2 - 10cos(2\pi y_i) + 10]$ $y_i = \left\{ \begin{array}{ll} x_i & \lvert x_i \rvert < \frac{1}{2} \\ \frac{round(2x_i)}{2} & \lvert x_i \rvert \geq \frac{1}{2} \end{array} \right\}$	$[-5.12, 5.12]^D$	0
$f_8(\mathbf{x}) = 418.98288727243380 \times D - \sum_{i=1}^{D} x_i \sin(\sqrt{\lvert x_i \rvert})$	$[-500, 500]^D$	0
$f_9(\mathbf{x}) = \frac{\pi}{D}\{10 \sin^2(\pi y_1) + \sum_{i=1}^{D-1} (y_1 - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})]$ $+(y_D - 1)^2\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{1}{4}(x_i + 1),\ u_{x_i,a,k,m} = \left\{ \begin{array}{ll} k(x_i - a)^m & x_i > a \\ 0 & -a \leq x_i \leq a \\ k(-x_i - a)^m & x_i < -a \end{array} \right\}$	$[-100, 100]^D$	0
$f_{10}(\mathbf{x}) = \frac{1}{10}\{\sin^2(\pi x_1) + \sum_{i=1}^{D-1} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})] +$ $(x_D - 1)^2 [1 + \sin^2(s\pi x_{i+1})]\} + \sum_{i=1}^{D} u(xi, 5, 100, 4)$	$[-100, 100]^D$	0
$f_{11}(\mathbf{x}) = \sum_{i=1}^{D-1} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})] + \sin^2(3\pi x_1) + \lvert x_D - 1 \rvert [1 + \sin^2(3\pi x_D)]$	$[-10, 10]^D$	0
$f_{12}(\mathbf{x}) = \sum_{i=1}^{D} \left(\sum_{k=0}^{k_{max}} [a^k \cos(2\pi b^k (x_i + 0.5))] \right) - D \sum_{k=0}^{k_{max}} [a^k \cos(2\pi b^k 0.5)]$ $a = 0.5, b = 3, k_{max} = 20$	$[-0.5, 0.5]^D$	0

algorithms while the Friedman test is adopted to evaluate the overall performances of all the concerned algorithms. The test results can be found at the bottom of the comparison tables. The symbol "+/=/-" indicates the number of functions where ABCL is better than, similar to, or worse than the competitor, respectively. At the end, the average rankings are presented in Figure 4.2.

Table 4.2: Comparison between ABCL and other ABC variants with $D = 10$

	ABC		DABC		GABC		ABCL	
Function	mean	± std	mean	± std	mean	± std	mean	± std
f_1	3.79E-25	8.29E-25	1.37E-39	5.41E-39	5.36E-50	1.46E-49	4.48E-50	6.03E-50
f_2	4.14E-45	2.06E-44	3.43E-37	1.72E-36	4.85E-83	2.39E-82	1.27E-82	4.26E-82
f_3	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_4	1.93E-22	0.00E+00	1.93E-22	0.00E+00	1.93E-22	0.00E+00	1.93E-22	0.00E+00
f_5	4.91E-02	4.50E-02	1.66E-01	1.80E-01	2.75E-01	6.16E-01	1.43E-02	1.72E-02
f_6	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_7	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_8	3.13E-12	4.66E-12	6.18E-13	4.33E-13	1.09E-13	3.02E-13	7.28E-14	2.52E-13
f_9	3.29E-31	9.66E-31	4.71E-32	1.12E-47	4.71E-32	1.12E-47	4.71E-32	1.12E-47
f_{10}	7.68E-31	3.47E-30	4.71E-32	1.12E-47	4.71E-32	1.12E-47	4.71E-32	1.12E-47
f_{11}	1.29E-27	2.27E-27	1.21E-30	1.79E-46	1.21E-30	1.79E-46	1.21E-30	1.79E-46
f_{12}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
+/=/-	7/5/0		4/8/0		3/8/1		-	

According to the comparison results with $D = 10$, the difference in performance of the compared algorithms is small as the low-dimensional case is simple to solve. Even so, the advantages of ABCL and GABC algorithms are noticeable, especially in terms of their high accuracy on functions such as f_1 and f_2. Similar conclusions can be derived from the Wilcoxon test results, the proposed algorithm performs better than GABC does on 3 problems. Meanwhile, they obtain very close results on 8 out of 12 problems. Compared with the standard ABC algorithm, ABCL surpasses it on more than half of the benchmarks. Moreover, it can be seen that the performance of DABC is also remarkable. It fails to outperform ABCL on 4 out of 12 problems.

According to Table 4.3, the ABCL is found effective and competitive on both uni-modal and multi-modal problems with $D = 30$. The advantages are significant, especially on f_1, f_5, and f_8. Based on the Wilcoxon tests, there is no case that ABCL performs worse than any other competitor on the considered benchmarks. The proposed algorithm obtains better solutions than ABC does on 7 functions. And they have similar results on the rest 5 problems. At the same time, the differences between results of GABC and ABCL are not very large since

Table 4.3: Comparison between ABCL and other ABC variants with $D = 30$

Function	ABC mean	± std	GABC mean	± std	DABC mean	± std	ABCL mean	± std
f_1	4.68E-23	6.32E-23	1.35E-44	1.48E-44	1.91E-37	5.05E-37	9.99E-45	1.80E-44
f_2	1.14E-46	4.59E-46	9.79E-82	4.89E-81	3.31E-25	1.64E-24	9.60E-82	2.61E-81
f_3	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_4	7.18E-66	3.23E-81	7.18E-66	3.23E-81	7.18E-66	3.23E-81	7.18E-66	3.23E-81
f_5	5.29E-02	6.36E-02	3.81E+00	7.43E+00	8.16E-02	1.38E-01	1.85E-02	2.21E-02
f_6	0.00E+00	0.00E+00	0.00E+00	0.00E+00	7.11E-17	3.55E-16	0.00E+00	0.00E+00
f_7	0.00E+00	0.00E+00	0.00E+00	0.00E+00	7.82E-15	3.80E-14	0.00E+00	0.00E+00
f_8	1.19E-09	5.64E-09	1.42E+01	3.93E+01	1.62E-11	7.09E-11	1.09E-12	9.09E-13
f_9	2.43E-31	3.55E-31	1.57E-32	5.59E-48	1.57E-32	5.59E-48	1.57E-32	5.59E-48
f_{10}	7.63E-31	2.05E-30	1.57E-32	5.59E-48	1.57E-32	5.59E-48	1.57E-32	5.59E-48
f_{11}	3.31E-25	8.20E-25	3.91E-30	2.15E-45	5.74E-30	9.14E-30	3.91E-30	2.15E-45
f_{12}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
+/=/-	7/5/0		4/8/0		7/5/0		-	

they obtain similar errors on 8 functions. And ABCL has better performance on the other 4 functions, which can help to prove the effectiveness of embedding the learning strategy in the ABC algorithm. Compared with DABC, ABCL is able to achieve smaller errors on 7 benchmarks.

Table 4.4: Comparison between ABCL and other ABC variants with $D = 50$

Function	ABC mean	± std	DABC mean	± std	GABC mean	± std	ABCL mean	± std
f_1	2.70E-22	2.94E-22	1.51E-36	5.15E-36	5.18E-43	4.08E-43	3.71E-43	3.97E-43
f_2	1.96E-46	9.61E-46	6.08E-21	2.71E-20	1.25E-83	4.33E-83	8.93E-82	2.52E-81
f_3	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_4	2.67E-109	9.65E-125	2.67E-109	9.65E-125	2.67E-109	9.65E-125	2.67E-109	9.65E-125
f_5	5.50E-02	6.10E-02	4.61E-02	4.59E-02	7.54E+00	1.95E+01	5.42E-02	7.11E-02
f_6	0.00E+00	0.00E+00	5.68E-16	2.22E-15	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_7	0.00E+00	0.00E+00	1.49E-14	5.42E-14	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_8	4.74E+00	2.37E+01	7.81E-10	3.79E-09	2.41E+01	6.00E+01	1.53E-04	7.64E-04
f_9	3.65E-31	4.65E-31	9.42E-33	1.40E-48	9.42E-33	1.40E-48	9.42E-33	1.40E-48
f_{10}	7.86E-31	1.62E-30	9.42E-33	1.40E-48	9.42E-33	1.40E-48	9.42E-33	1.40E-48
f_{11}	4.90E-24	6.64E-24	1.39E-28	6.63E-28	6.61E-30	1.43E-45	6.61E-30	1.43E-45
f_{12}	5.12E-15	8.08E-15	8.53E-15	1.00E-14	1.25E-14	1.18E-14	1.08E-14	1.18E-14
+/=/-	7/4/1		5/4/3		4/7/1		-	

In Table 4.4, when $D = 50$, ABCL is able to maintain its effectiveness in this case. Based on the Wilcoxon tests, ABCL outperforms the standard ABC algorithm on 7 functions and about the same on 4 problems. Moreover, there is 1 function where ABCL fails to surpass the ABC algorithm. Compared to DABC, the proposed algorithm obtains better solutions on 5 functions. And they obtain similar results on 4 problems. Meanwhile, the differences between the results of

GABC and ABCL are still not significant. And ABCL outperforms GABC on 4 problems.

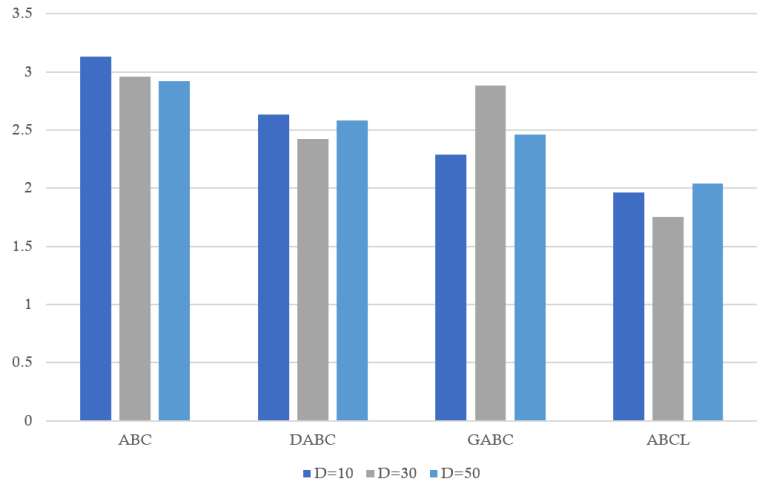
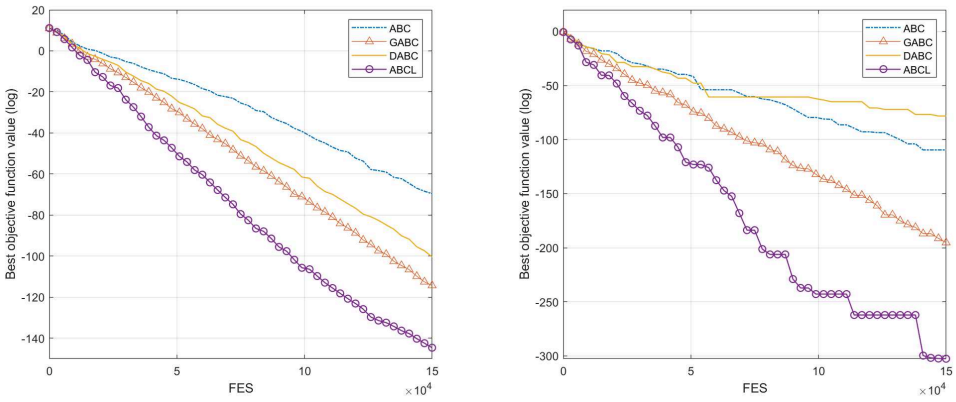


Fig. 4.2. Average rankings of ABC algorithms by Friedman test with $D = 10, 30,$ and 50

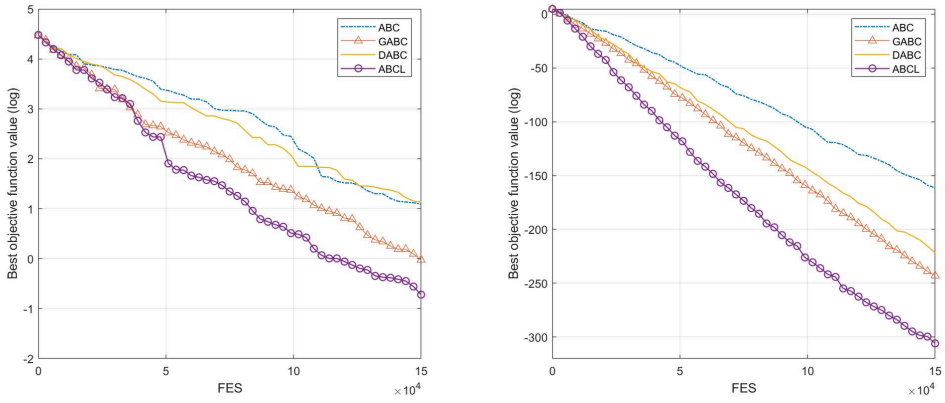
Furthermore, based on the average rankings given by the Friedman test in Figure 4.2, the proposed ABCL algorithm has the best rankings in all three cases. Therefore, it can be concluded that ABCL has the best overall performance among the competitors. Since the proposed algorithm performs well in solving both uni-modal and multi-modal functions, it keeps a nice balance between exploitation and exploration during the search process. The second best algorithm is given to GABC and followed by the DABC. Besides, it is normal that all the improved ABC variants achieve better performances than the basic ABC does. Therefore, the search effectiveness of ABCL can be proved by solving different types of problems.

4.4.3 Convergence behavior analysis

The convergence curves of compared algorithms were plotted in order to further compare their performance. The convergence process of solving different types of problems with $D = 30$ can be observed in Figure 4.3. Note that the values of $log(f(\cdot))$ are presented because the objective function values are too large in the earlier stages of the searching process.



(a) f_1 **(b)** f_4



(c) f_6 **(d)** f_9

Fig. 4.3. The convergence performance of ABCL and compared ABC algorithms

In Figure 4.3, (a) and (b) present the convergence curves of ABC algorithms on uni-modal functions f_1 and f_4, respectively. Since the results of solving uni-modal functions can indicate the exploitation ability of an algorithm, it can be found that ABCL's exploitation ability has been remarkably enhanced. And the proposed algorithm is able to achieve the best accuracy.

Moreover, subfigures (c) and (d) show the cases of solving multi-modal problems f_6 and f_9. A similar conclusion can be derived from these cases, where ABCL converges rapidly to the smallest value among the concerned methods. In this case, the diversification of ABCL is also outstanding compared to the other three methods. At the same time, the convergence rate of the GABC algorithm is competitive, which is the second best in the comparisons. Furthermore, the advantages in solution accuracy of ABCL are significant as shown in the figures. This indicates that it has the ability to find a better solution in a limited number of iterations.

4.5 Conclusion

In this chapter, an improved ABC algorithm named ABCL is proposed with the ultimate goal of solving the on-line MRPP problems (see Chapter 6). Hence, several improvement strategies have been adopted in order to enhance the intensification as well as the search efficiency. In the first place, the global best information is used to reinforce the algorithm. A solution search equation based on the global best and more neighbors is utilized in the employed bee phase. Secondly, the learning phase of the TLBO algorithm is introduced into the onlooker bee phase to improve the exploitation ability and search efficiency. Moreover, the scout bee phase is also enhanced with the global best solution. In this case, more useful information can be considered when generating a new candidate solution. Furthermore, the effectiveness of ABCL is proved through comparisons in solving benchmark optimization problems.

Chapter 5

Fractional-order ABC algorithm (FOABC)

Contents

5.1 Introduction

It can be found that, in most ABC variants, the solution search equations are often improved by enlarging the number of dimensions to be updated and expanding the quantity of information that may be learned from the colony. The proposed ABC algorithms in Chapters 2-4 also improve ABC by this idea. However, these kind of improvements disregard individuals' prior search history, which could result in the loss of valuable information. In fact, from a mathematical perspective, it can be found that the majority of improved ABC variants search the optima via integer-order operations. In addition to the common improvement strategies that were reviewed in the first chapter, new idea is produced and researched in this chapter. More precisely, it is interesting to investigate whether combining fractional order calculus will help enhance the ABC algorithm.

In this context, the fractional-order calculus (FOC), a novel mathematical method is embedded into ABC algorithm to reinforce the exploitation ability and improve the solution precision. In fact, compared to the integer-order derivative, the fractional-order derivative contains entire memory of its previous events (Couceiro *et al.*, 2012; Mousavi & Alfi, 2018; Yousri *et al.*, 2020). In recent years, the FOC has been introduced into certain optimization approaches as it can provide the historical information clearly. Solteiro Pires *et al.* (2010) incorporated the FOC into the velocity of individuals in PSO algorithm to better control the convergence speed. Then, the FOC was utilized to improve the convergence rate of darwinian particle swarm optimization (DPSO) which was extended from PSO (Couceiro *et al.*, 2012). Furthermore, in fractional calculus-based firefly algorithm (FOFA) (Mousavi & Alfi, 2018), a solution search equation based on the FOC was proposed to give each firefly more historical information. The performance of FOFA has been demonstrated by testing on benchmark functions and application of image segmentation. Deshmukh & Usha Rani (2019) updated the wolf positions in the grey wolf optimizer (GWO) using the memory property of the FOC, which led to boost the convergence rate. More recently, Yousri & Mirjalili (2020) enhanced the random walk of CS algorithm by employing the FOC. And the proposed FOCS has been effectively used to identify the parameters of three different types of chaotic financial systems. Meanwhile, an improved flower pollination algorithm has adopted the FOC as well. The local-search ability of original algorithm has been enhanced by adding the memory features (Yousri

et al., 2020). Based on the above literature, the incorporation of FOC brought surprising results and was also found to make an improvement in the convergence speed of the algorithm.

In this context, it has been observed that incorporating FOC into the ABC algorithm is very worthy to be studied. More specifically, the integration of FOC perspective into ABC enables the algorithm to access previous solutions from memory to find qualified solutions more efficiently. Meanwhile, ABC can also benefit from the acceleration in convergence as FOC has done for other meta-heuristic algorithms.

Therefore, in this chapter, in order to make full use of the individuals' memories, a FO-based search strategy is proposed in the onlooker bee phase of ABC. At the same time, a differential search strategy is adopted in the employed bee phase of the proposed FOABC algorithm to reinforce its diversification ability. Moreover, the scale factors of these solution search equations are generated via Lévy distribution to increase the randomness of algorithm. To validate the performance of FOABC, it is compared with six effective ABC algorithms and four improved non-ABC algorithms on the Congress on Evolutionary Computation 2017 (CEC 2017) benchmark functions.

The outline of this chapter is as follows. The proposed algorithm is introduced in section 5.2. Experimental studies are presented in section 5.3 with the discussions. The conclusion is given in the last section 5.4.

5.2 Proposed FOABC algorithm

In this section, the proposed FOABC will be explained in details. In the employed bee phase, a DE-based search strategy is adopted to reinforce the diversification of algorithm. Secondly, the FOC is adopted in the onlooker bee phase in order to improve the local-search ability. Moreover, random numbers drawn from the Lévy distribution are utilized in those two search equations.

5.2.1 Scale factors based on Lévy distribution

As one of the main components of the search equation, scale factor can affect the step size of each move. More precisely, this factor is usually a random number, depending on the range and distribution laws. In the standard ABC algorithm,

the scale factor $\phi \in [-1, 1]$ has been found to be insufficiently efficient. Additionally, when algorithm falls into a local optimum, the restricted range of ϕ may limit the algorithm to jump out quickly. In addition to the popular distributions like normal distribution and uniform distribution, the Lévy distribution (Lévy, 1938) is also very important in the field of meta-heuristics. Lee & Yao (2004) mentioned that the Lévy probability distribution has an infinite second moment. In this case, it is more probable to produce an offspring that is spatially far from its parent in the concerned evolutionary programming (EP) algorithm. Yang & Deb (2009) has proposed the CS algorithm with Lévy flights whose steps obey the Lévy distribution. Such search manners contain a series of straight flight pathways punctuated by a sudden change in direction. Hence, it enables the algorithm to explore the space efficiently.

Lévy flight is a special case of random walk whose step lengths obey the Lévy distribution which can be defined as below (Shlesinger, 1989; Yang, 2020):

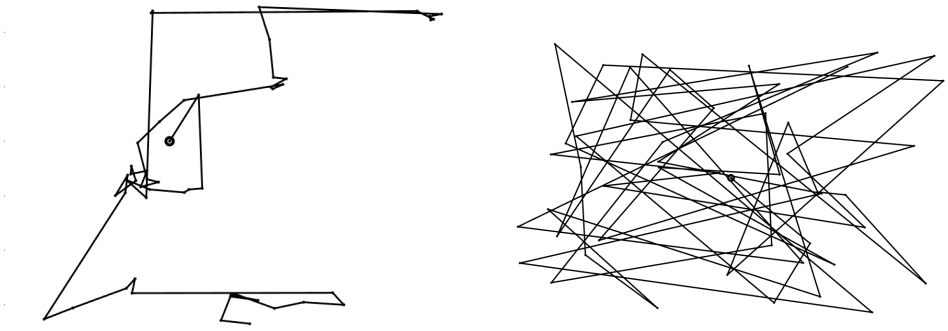
$$L\acute{e}vy(s) \sim |s|^{-1-\beta}, \tag{5.1}$$

where s is the step length and $\beta(0 < \beta \leq 2)$ is an index.

In actuality, Lévy flights are effective since they are powerful at searching uncharted and large areas. The path of Lévy flights of 50 steps with $\beta = 1.5$ is plotted in Figure 5.1a. It is obvious that large steps are occasionally created. Meanwhile, for visual comparison, the traditional random walk based on uniform distribution is also displayed in Figure 5.1b. It can be seen that such kind of random walk has a relatively average step size and does not have significant variation in direction throughout the movement. In this case, the Lévy flights have advantages in preventing the algorithm from being stuck in local optima and increasing the variety of solutions (Wang *et al.*, 2022).

Therefore, the idea of Lévy flights is utilized in the search equations. In order to generate such random step length, the Mantegna algorithm(Mantegna, 1994) has been widely used as shown in Eq.(5.2). It is one of the most accurate and efficient methods to generate stochastic variables drawn from the Lévy stable distribution (Liu *et al.*, 2018; Sharma *et al.*, 2016; Yang, 2020).

$$s = \frac{u}{|v|^{1/\beta}}, \tag{5.2}$$



(a) Lévy flights in consecutive 50 steps **(b)** Random walk with a uniform step-size distribution in consecutive 50 steps

Fig. 5.1. Demonstration of Lévy flights and traditional random walk in 2D

where u and v are drawn from normal distributions as:

$$u \sim N(0, \sigma_u^2), \quad v \sim N(0, \sigma_v^2), \tag{5.3}$$

with

$$\sigma_u = \left\{ \frac{\Gamma(1+\beta) sin(\pi\beta/2)}{\beta\Gamma[(1+\beta)/2] 2^{(\beta-1)/2}} \right\}^{1/\beta}, \quad \sigma_v = 1, \tag{5.4}$$

where $\Gamma(\cdot)$ is the Gamma function and is defined as follows.

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-z} dt, \tag{5.5}$$

when z is an integer, $\Gamma(z+1) = z\Gamma(z) = z!$.

Note that the step length mentioned above plays the same role as the random factor ϕ in search equation (1.2). In the proposed algorithm, the scale factors l in search strategies are generated by the following Eq.(5.6).

$$l = \alpha \times s, \tag{5.6}$$

where s is the step length generated via Eq.(5.2) and $\alpha > 0$ is a factor. In the basic CS, α is used to adjust the step size for Lévy flight considering the problem scale. Sharma *et al.* (2016) fixed this factor to 0.001 in their search strategy. And $\alpha = 0.01$ in the codes of CS proposed by researchers who develop it. In fact, this factor is also important since it can probably influence the effectiveness of the search strategies. If α is defined with a large value, then the novel solution

may jump over the optimal position or even go outside the design domain as the step size is too large. On the other hand, if the value of α is too small, then the convergence rate will be affected. In this case, instead of using a constant throughout the search process, the logistic map is applied on the factor α in (Liu *et al.*, 2018). The value of factor α changes along with the iteration and the logistic dynamic map can help the algorithm to prevent the situation of falling into local optima.

In our proposed algorithm, the two search equations in employed bee phase and onlooker bee phase both adopt the random numbers generated via Eq.(5.6). And based on our experimental tests, $\alpha_1 = 0.7$ for the Lévy-based random number in employed bee phase. The factor α_2 in onlooker bee phase should be set relatively small to avoid missing the optimal solution so $\alpha_2 = 0.5$.

Remark 5.1 *The possible interval for α is pretty large. And varying the values of α may also help to improve the algorithm performance when solving different kinds of problems.*

5.2.2 Differential search strategy for employed bee phase

As mentioned previously, the search strategy of ABC is not efficient enough because it modifies only one variable of the food source positions each time. Moreover, limited information can be learned from the swarm as only one solution is randomly selected from the swarm. In most of ABC variants, the nb_{up} (number of dimensions being updated) in the enhanced solution search equations is still one dimension each time. In order to tackle this weakness, differential search strategies were borrowed from the DE algorithm (Chen *et al.*, 2019b; Li & Yin, 2014; Zorarpacı & Özel, 2016) because the mutation and crossover operators enable the individuals to explore better via a sudden change. By using a predefined control parameter CR, the nb_{up} could be greater than one at each time.

Thus, a differential search strategy is utilized in the employed bee phase. To avoid getting over complicated, the classical but efficient DE search strategy "*rand/1/bin*" is adopted. The search strategy in employed bee phase of FOABC is shown as below:

$$v_{i,j} = \begin{cases} x_{r_1,j} + l_i^1 \times (x_{r_2,j} - x_{r_3,j}), \ if\ rand \leqslant CR\ or\ j = j_{rand}, \\ x_{i,j}, \ otherwise, \end{cases} \tag{5.7}$$

where v_i is the updated solution of x_i. $r_1 \neq r_2 \neq r_3 \neq i$ are randomly chosen from $\{1, \cdots, SN\}$. And j_{rand} is randomly selected from $\{1, \cdots, D\}$. CR is the crossover rate while l_i^1 is scale factor generated via Eq.(5.6) with $\alpha_1 = 0.7$.

For each dimension, if the condition of is satisfied, the new solution will get information from the mutating vector. Otherwise, it will keep the same as the original food source x_i. In this case, by embedding the DE-based operators into FOABC, the new candidate solutions are able to learn more information from other food sources and inherit from the previous ones at the same time. Moreover, the conditions in selection process allow the individuals update more than one dimension each time.

5.2.3 Fractional-order search strategy for onlooker bee phase

As mentioned in the beginning, compared to an integer-order derivative, fractional-order derivative has a memory of previous incidents (Couceiro *et al.*, 2012; Mousavi & Alfi, 2018; Yousri *et al.*, 2020). Considering its excellent competence of describing the historical information, the FOC is adopted in the onlooker bee phase in order to enhance the local-search ability as well as the solution precision.

5.2.3.1 Fractional-order calculus definition

Nowadays, three common definitions of fractional-order derivatives have been widely approved and used: Grunwald–Letnikov, Rieman–Liouville and Caputo definitions (Gu *et al.*, 2017). In this chapter, the definition of Grunwald-Letnikov (GL) derivative will be used and it can be implemented as Eq.(5.8) (Podlubny, 1999).

$$D^q[x(t)] = \lim_{h \to 0} \frac{1}{h^q} \sum_{n=0}^{\infty} (-1)^n \binom{q}{n} x(t - nh), \tag{5.8}$$

where

$$\begin{aligned} \binom{q}{n} &= \frac{\Gamma(q+1)}{\Gamma(n+1)\Gamma(q-n+1)} \\ &= \frac{q(q-1)(q-2)\cdots(q-n+1)}{n!}, \end{aligned} \tag{5.9}$$

where $D^q(\cdot)$ denotes the GL fractional derivative of order q. $\Gamma(\cdot)$ is the Gamma function defined in (5.5).

The definition of GL in Eq.(5.8) can be formulated as below in the discrete-time implementation.

$$D^q[x(t)] := \frac{1}{T^q}\sum_{n=0}^{r}(-1)^n \binom{q}{n} x(t-nT), \tag{5.10}$$

where T is the sampling period and r indicates the number of terms of the previous memory. $(-1)^n \binom{q}{n}$ are called the binomial coefficients. These coefficients are usually expressed as $c_n^{(q)}$ $(n = 0, 1, \cdots)$ and are calculated as below (Yameni Noupoue *et al.*, 2019):

$$\begin{cases} c_0^{(q)} = 1, \\ c_n^{(q)} = (1 - \dfrac{q+1}{n})c_{n-1}^{(q)}. \end{cases}$$

Considering a special case that $q = 1$, then the definition can be expressed as follows.

$$D^1[x(t)] = \frac{1}{T}(x(t) - x(t-T)) = x^t - x^{t-1}. \tag{5.11}$$

It can be found that $D^1[x(t)]$ represents the difference between two events at adjacent moments, x^t and x^{t-1}.

5.2.3.2 Proposed fractional-order solution search equation

As mentioned before, the FO derivative keeps memory of previous events compared to the integer-order derivative. And FOC has been verified that it is suitable to be used to describe the dynamic phenomena such as the trajectory of fireflies (Mousavi & Alfi, 2018). In the exploitation stage, FOC enables the algorithm to use information from earlier solutions when looking for new ones, thus determining a more plausible solution and changing the convergence tendency (Couceiro *et al.*, 2012). Therefore, considering its excellent ability in describing historical information, FOC is employed in the onlooker bee phase to enhance the local search ability of ABC.

Firstly, in order to show the derivation process more clearly, we rewrite the Eq.(1.2) as following:

$$x_{i,j}^{t+1} = x_{i,j}^t + l_{i,j}^2 \times (x_{i,j}^t - x_{k,j}^t), \tag{5.12}$$

where $x_{i,j}^{t+1}$ is the newly produced solution which is the same as $v_{i,j}$ and $k \neq i$ is randomly selected among $\{1, \cdots, SN\}$. Note that instead of generating a random number under normal distribution, the coefficient $l_{i,j}^2$ is produced via Eq.(5.6) in subsection 5.2.1.

According to the difference of two followed events in Eq.(5.11), the Eq.(5.12) can be then reformulated as

$$x_{i,j}^{t+1} - x_{i,j}^{t} = l_{i,j}^2 \times (x_{i,j}^{t} - x_{k,j}^{t}), \tag{5.13}$$

then it is easy to get the following formulas when $q = 1$:

$$D^1[x(t+1)] = x_{i,j}^{t+1} - x_{i,j}^{t} = l_{i,j}^2 \times (x_{i,j}^{t} - x_{k,j}^{t}). \tag{5.14}$$

The derivative definition above for $q = 1$ can be generalized as Eq.(5.15) with any derivative order q,

$$D^q[x_{i,j}^{t+1}] = l_{i,j}^2 \times (x_{i,j}^{t} - x_{k,j}^{t}). \tag{5.15}$$

Next, we are able to formulate the Eq.(5.15) by employing the discrete-time GL definition Eq.(5.10) with $T = 1$. The novel expression are shown below:

$$\begin{aligned} D^q[x_{i,j}^{t+1}] &:= x_{i,j}^{t+1} + \sum_{n=1}^{r} c_n^{(q)} x_{i,j}^{t+1-n} \\ &= l_{i,j}^2 \times (x_{i,j}^{t} - x_{k,j}^{t}), \end{aligned} \tag{5.16}$$

where r is the number of steps to record.

Based on Eq.(5.16), the general expression of the proposed fractional-order ABC solution search equation is obtained and expressed in the following:

$$x_{i,j}^{t+1} = -\sum_{n=1}^{r} c_n^{(q)} x_{i,j}^{t+1-n} + l_{i,j}^2 \times (x_{i,j}^{t} - x_{k,j}^{t}), \tag{5.17}$$

where r is the number of memory terms and q is derivative order. $c_n^{(q)}$ are the binomial coefficients. And j is randomly selected dimension from $\{1, \cdots, D\}$, $k \neq i$ is randomly chosen from $\{1, \cdots, SN\}$. The random factor $l_{i,j}^2$ is generated by Eq.(5.6) with $\alpha_2 = 0.5$.

For instance, $r = 4$, which means that the latest four terms will be stored in

the memory of each food source and will be utilized to produce a new candidate solution via the Eq.(5.17). In this case, the food sources will be updated as follows.

$$\begin{aligned} x_{i,j}^{t+1} = & \frac{1}{1!}qx_{i,j}^{t} + \frac{1}{2!}q(1-q)x_{i,j}^{t-1} + \frac{1}{3!}q(1-q)(2-q)x_{i,j}^{t-2} \\ & + \frac{1}{4!}q(1-q)(2-q)(3-q)x_{i,j}^{t-3} + l_{i,j}^{2} \times (x_{i,j}^{t} - x_{k,j}^{t}). \end{aligned} \tag{5.18}$$

It is evident that the values of parameters q and r can influence the final results. However, it is relatively difficult to understand their effects, especially the fractional order q. This drove the existing FO-enhanced meta-heuristic algorithms to set these parameters basically based on experimental validation. The following analysis takes Eq.(5.18) as an example search equation, and for the sake of clarity, the first four terms related to FOC are named as memory terms and the last term is called differential term.

Observing Eq.(5.18), when q is taken smaller in the range of [0,1], the coefficients of memory terms are also smaller, so the historical memory has less influence on the newly generated solution. In this way, the differential term may dominate the updating results. That means the historical information cannot play its role well. Put another way, the summation of the memory terms is equivalent to the base term $x_{i,j}^{t}$ in the original solution search equation. That is, if q is defined too small, the base term will be scaled to be smaller. Then, the local search will be affected, because the algorithm cannot approximately locate the position of base term. Hence, the local search may not be performed effectively.

On the other hand, the memory terms play a greater role when q takes a larger value, especially around 0.5. As mentioned before, in the onlooker bee phase, the selection probability of the solutions to be updated is higher, that is, their history information is more useful. In this case, the memory terms can play an active role in the proposed search equation. And in contrast to the previous case, when q is larger, the algorithm can better perform a local search around the base term.

As for the parameter r, the memory length, is relatively easy to understand, of course. Note that r taking a value too large may cause the algorithm to take more time on computation. Based on the above analysis, since the FO-based strategy is supposed to boost the local search, taking a relatively large value of q in $[0, 1]$ is more likely to yield the desired result. Furthermore, experiments are

done to fully evaluate the impact of these two parameters on the performance of FOABC while completing the analysis of their influence.

5.2.3.3 Implementation steps of modified onlooker bee phase

The proposed fractional-order search strategy is utilized in the onlooker bee phase to reinforce the exploitation ability of algorithm. The principle steps of implementing the FO-based onlooker bee phase is explained as follows.

Initializing memory cells

Each onlooker bee has a corresponding memory cell M_i. And initially all memory cells are empty. After having generated the initial food sources, the initial positions x_i are stored in their respective corresponding M_i. Note that when the number of iterations is less than the set r, the number of memory items that can be called is the same as the number of iterations. That is, the memory length involved in the solution search equation grows gradually to the fixed value r and then remains constant.

For instance, assuming the number of memory terms $r = 4$, at the third iteration, the FO-based equation contains only the memory terms corresponding to the first two iterations. Starting from the 5^{th} iteration, the memory cells maintain the latest four memory terms.

Updating the memory cells

In the onlooker bee phase, same roulette wheel selection method is utilized for choosing food sources to exploit. Once a food source is selected by an onlooker bee, it will be updated with Eq.(5.17). And then the new food source position will be stored into its associated memory cell. As we have introduced above, it is possible that the number of terms stored in the associated memory is smaller than r. Then in this case, the memory cell can be updated easily by adding the new position. Otherwise, when the size of memory cell is already r, a method called first in first out (FIFO) is utilized to update the memory (Yousri *et al.*, 2020). The oldest position will be abandoned and the new position will be stored into the memory cell. To demonstrate this updating mechanism more clearly, the process is shown in Figure 5.2.

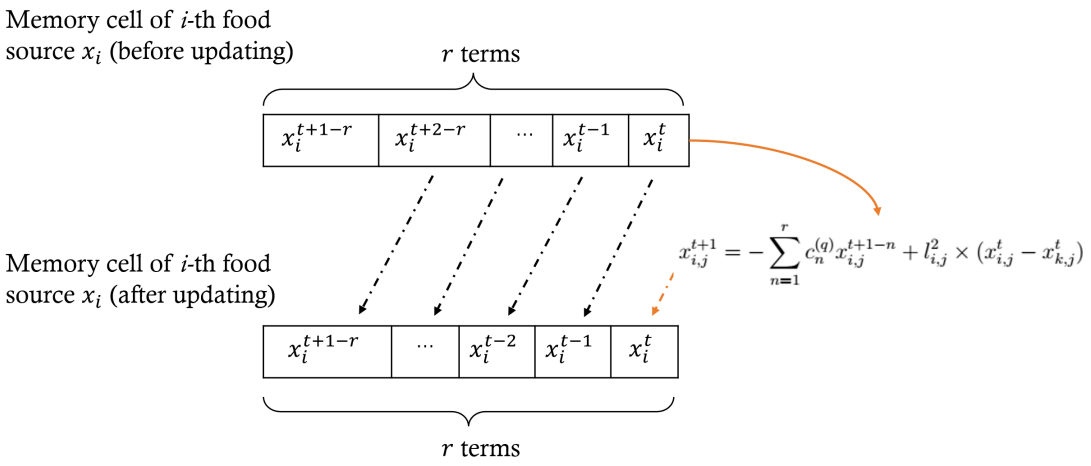


Fig. 5.2. Process of updating a food source's memory cell

5.2.4 The framework of FOABC algorithm

In order to explain the process of FOABC more clearly, the pseudo-code and flowchart are presented in Algorithm 13 and Figure 5.3, respectively.

5.3 Experiments on function optimization problems

In this section, several series of experiments are conducted in order to verify the performance of FOABC algorithm. Firstly, experimental studies are carried out to investigate sensitivity of FOABC to parameters r (the number of terms to memory) and q (fractional order) of FOC in subsection 5.3.1. Then in subsection 5.3.2, comparisons are made with five state-of-the-art ABC variants and the original ABC on 29 CEC 2017 benchmark problems with dimension $D = 10$, 30 and 50. Thirdly, the FOABC algorithm is compared with other improved meta-heuristic algorithms in subsection 5.3.3. And subsection 5.3.4 presents the redundancy elimination experiments to verify the effectiveness of each proposed strategy.

Same as Chapter 4, the CEC 2017 benchmark problems are utilized since it contains different kinds of single-objective real-number optimization problems: uni-modal functions ($f_1 - f_3$), simple multi-modal functions ($f_4 - f_{10}$), hybrid functions ($f_{11} - f_{20}$) and composition functions ($f_{21} - f_{30}$). And the code is

Algorithm 13 Pseudo-code of FOABC algorithm

Set $limit = SN \times D$, $CR = 0.8$, $\beta = 1.5$, $\sigma_v = 1$
Generate initial population x_i, $i = 1, \cdots, SN$ with Eq.(1.1)
Store the initial positions into corresponding memory cell M_i, $i = 1, \cdots, SN$
Evaluate objective function values of the population, $FES = SN$
while $FES \leq max\_FES$ **do**
    % `DE-based employed bee phase` %
  for $i = 1 \rightarrow SN$ **do**
    Generate l^1 via Eq.(5.6)
    Randomly select $j_{rand} \in \{1, \cdots, D\}$ and $r_1 \neq r_2 \neq r_3 \neq i$ from $\{1, \cdots, SN\}$
    Generate new food source position v_i via Eq.(5.7)
    if $f(v_i) < f(x_i)$ **then**
      Replace x_i with v_i, $trial_i = 0$
    else $trial_i = trial_i + 1$
    end if
  end for
  Evaluate the probability values with Eq.(1.5)
    % `FO-based onlooker bee phase` %
  for $t = 1 \rightarrow SN$ **do**
    Select x_i by roulette wheel method according *prob*
    Generate l^2 via Eq.(5.6)
    Randomly select $k \neq i$ and randomly select j from $\{1, \cdots, D\}$
    Produce the $v_{i,j}$ via Eq.(5.17)
    Update M_i with v_i based on FIFO method
    if $f(v_i) < f(x_i)$ **then**
      Replace x_i with v_i, $trial_i = 0$
    else $trial_i = trial_i + 1$
    end if
  end for
  $FES = FES + 2SN$
    % `Scout bee phase` %
  for $i = 1 \rightarrow SN$ **do**
    if $trial_i > limit$ **then**
      Generate new position with Eq.(1.1), $trial_i = 0$
    end if
  end for
end while

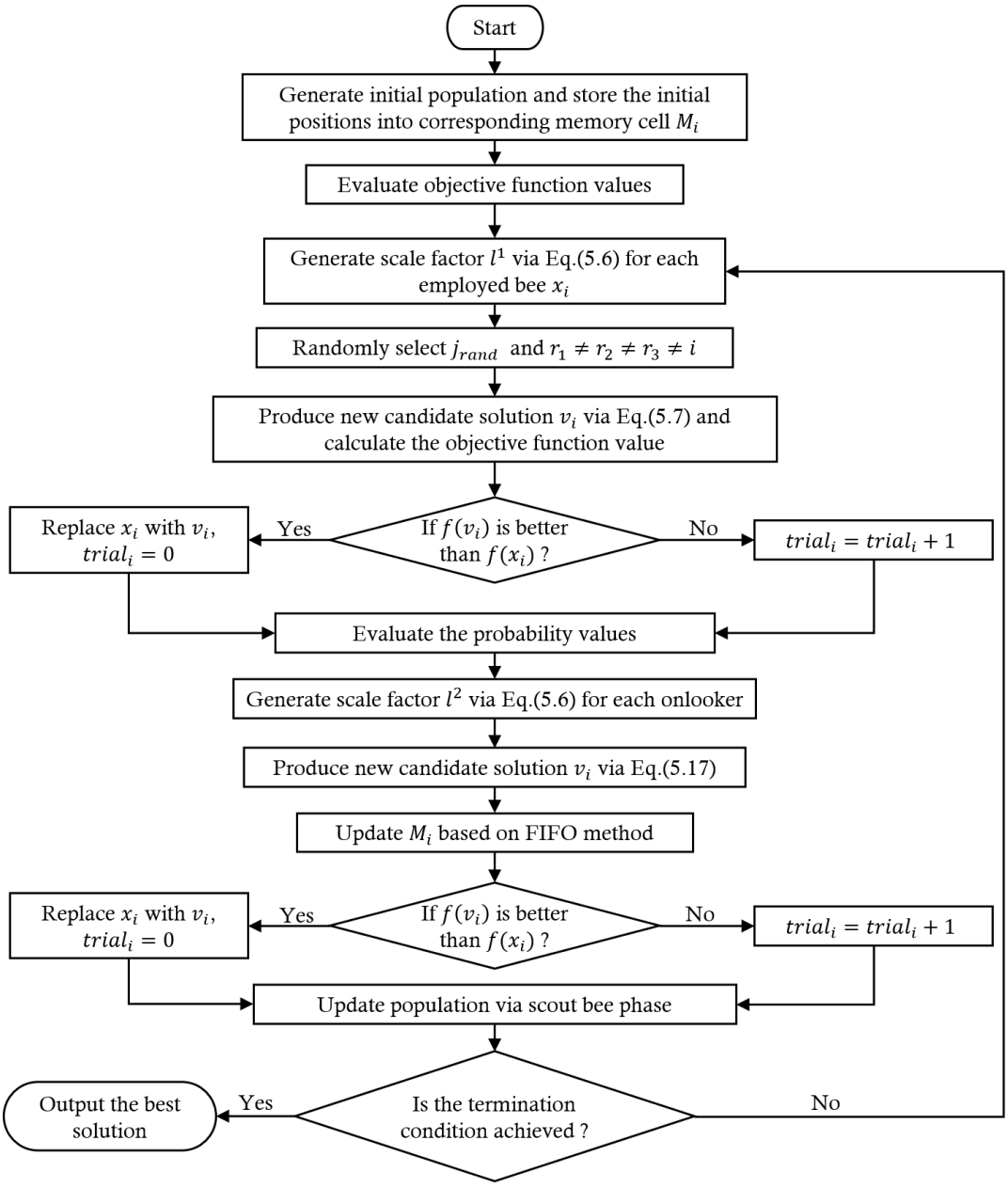


Fig. 5.3. The flowchart of FOABC algorithm

provided officially*. Notice that the function f_2 has been deleted in the code, so there is no results for f_2 in all the comparisons in this chapter.

In all the following experiments, the evaluation criteria of CEC 2017 is respected which means the maximum functions evaluations $max\_FES = 10^4 \cdot D$, the search ranges are $[-100, 100]^D$. Each involved algorithm was run 30 times independently on all the problems. Then the mean value and standard deviation (Std) of function error values $f(X_{best}) - f(X^*)$ are calculated for comparison. The best solution X_{best} is found by the compared algorithm while X^* is the exact global optimum.

Moreover, the Wilcoxon rank sum test at 0.05 significant level is employed as well. The symbol "+" indicates that FOABC is better than the compared algorithm. The symbols "-" and "=" denote that the result of FOABC is worse than and similar to the compared one, respectively. The total numbers of each symbol are counted in each comparison table. And Friedman tests are conducted to obtain the rankings of concerned methods.

5.3.1 Sensitivity analysis of r and q

In the proposed FOABC algorithm, the control parameters of FOC can influence the final performance. So in this part, experiments are done to study the sensitivity of FOABC to the fractional order q and terms of memory r.

The FOABC variants with different values of q and r are applied on CEC 2017 benchmark functions at dimension $D = 30$. The population size is fixed to $SN = 50$. In addition, for each optimization function, each algorithm is run 30 times independently. Then the statistical results of the errors are calculated.

The terms of memory r is tested as 4, 8, and 12. And for each tested value of r, the derivative order q varies from 0.1 to 0.9 with step of 0.1. The experimental results are shown in Tables 5.1 - 5.3 for $r = 4, 8$ and 12, respectively. The basic ABC algorithm is involved in order to observe and demonstrate the effectiveness of the FOABC variants.

Moreover, the Friedman test and Wilcoxon rank-sum test with significant difference 0.05 are conducted on the results. The algorithm with best ranking given by Friedman test is marked in **boldface**. And each FOABC version is compared against the standard ABC algorithm via Wilcoxon rank-sum tests.

*https://github.com/P-N-Suganthan/CEC2017-BoundContrained

Table 5.1: Comparison of FOABC variants with number of terms $r = 4$ and q taked values from 0.1 to 0.9 with ABC

			$r = 4$								
Functions		ABC	$q = 0.1$	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
f_1	Mean	3.293E+02	1.920E+03	2.505E+03	1.156E+03	1.536E+03	1.114E+03	1.916E+03	2.179E+03	1.755E+03	2.027E+03
	Std	3.536E+02	3.020E+03	3.518E+03	2.733E+03	2.888E+03	1.768E+03	4.410E+03	4.079E+03	2.956E+03	4.397E+03
f_3	Mean	7.361E+04	3.914E+04	3.694E+04	3.861E+04	3.835E+04	3.354E+04	3.575E+04	4.124E+04	4.149E+04	4.408E+04
	Std	2.424E+04	8.653E+03	9.621E+03	1.064E+04	1.066E+04	1.010E+04	9.689E+03	9.762E+03	1.104E+04	1.427E+04
f_4	Mean	1.890E+01	9.031E+01	9.405E+01	9.418E+01	8.663E+01	9.033E+01	8.926E+01	9.055E+01	9.053E+01	8.847E+01
	Std	2.274E+01	1.474E+01	1.719E+01	1.540E+01	1.799E+01	1.594E+01	2.137E+01	1.399E+01	1.186E+01	1.201E+01
f_5	Mean	8.327E+01	8.037E+01	7.525E+01	6.580E+01	6.062E+01	5.588E+01	5.099E+01	4.761E+01	4.659E+01	4.515E+01
	Std	1.315E+01	6.934E+00	6.169E+00	5.322E+00	8.241E+00	9.203E+00	6.637E+00	7.282E+00	6.493E+00	7.393E+00
f_6	Mean	7.420E-10	2.815E-05	2.619E-04	7.723E-05	5.211E-06	4.332E-05	1.199E-04	4.349E-05	1.419E-05	3.437E-05
	Std	8.144E-10	8.018E-05	1.325E-03	1.260E-04	1.233E-05	1.077E-04	4.643E-04	9.696E-05	3.843E-05	8.877E-05
f_7	Mean	1.006E+02	1.265E+02	1.216E+02	1.189E+02	1.066E+02	9.838E+01	9.294E+01	8.721E+01	7.852E+01	7.644E+01
	Std	8.186E+00	7.213E+00	8.907E+00	1.055E+01	7.975E+00	6.916E+00	6.657E+00	8.628E+00	8.695E+00	6.726E+00
f_8	Mean	8.879E+01	6.858E+01	6.436E+01	6.142E+01	5.581E+01	5.176E+01	4.982E+01	4.908E+01	4.792E+01	4.790E+01
	Std	1.432E+01	1.056E+01	8.143E+00	6.484E+00	6.178E+00	7.445E+00	6.899E+00	6.577E+00	8.717E+00	7.382E+00
f_9	Mean	7.745E+02	8.719E-01	7.078E-01	7.349E-01	5.154E-01	5.990E-01	6.004E-01	4.837E-01	5.900E-01	2.901E-01
	Std	4.200E+02	1.475E+00	1.232E+00	1.507E+00	9.960E-01	9.295E-01	1.021E+00	6.668E-01	7.108E-01	3.486E-01
f_{10}	Mean	2.220E+03	2.995E+03	2.899E+03	2.755E+03	2.816E+03	2.692E+03	2.595E+03	2.563E+03	2.482E+03	2.453E+03
	Std	2.321E+02	2.009E+02	2.450E+02	2.378E+02	1.951E+02	2.680E+02	3.162E+02	3.185E+02	3.437E+02	2.260E+02
f_{11}	Mean	2.331E+02	4.257E+01	4.343E+01	3.687E+01	3.316E+01	2.705E+01	2.719E+01	2.276E+01	2.152E+01	2.502E+01
	Std	1.787E+02	2.754E+01	2.624E+01	2.832E+01	2.001E+01	2.110E+01	1.621E+01	1.624E+01	1.552E+01	1.538E+01
f_{12}	Mean	5.541E+05	3.775E+04	3.881E+04	3.671E+04	5.543E+04	3.965E+04	2.774E+04	5.410E+04	3.883E+04	3.536E+04
	Std	2.568E+05	1.951E+04	2.784E+04	1.990E+04	1.187E+05	5.619E+04	2.000E+04	1.198E+05	2.786E+04	1.669E+04
f_{13}	Mean	8.049E+03	1.658E+04	1.093E+04	1.927E+04	1.338E+04	8.706E+03	1.214E+04	1.736E+04	1.963E+04	1.199E+04
	Std	7.078E+03	1.955E+04	9.798E+03	1.996E+04	1.644E+04	1.185E+04	1.323E+04	1.793E+04	1.899E+04	1.391E+04
f_{14}	Mean	4.534E+04	7.202E+01	7.379E+01	7.798E+01	1.089E+02	6.551E+01	7.987E+01	9.660E+01	7.471E+01	7.489E+01
	Std	3.859E+04	3.434E+01	3.064E+01	3.515E+01	1.140E+02	3.328E+01	7.165E+01	1.445E+02	5.655E+01	3.707E+01
f_{15}	Mean	1.301E+03	7.076E+02	2.389E+02	4.226E+02	7.225E+02	5.627E+02	1.519E+02	1.106E+03	1.400E+03	1.009E+02
	Std	9.227E+02	1.985E+03	3.904E+02	1.017E+03	2.562E+03	2.236E+03	1.865E+02	4.597E+03	5.460E+03	1.072E+02
f_{16}	Mean	6.639E+02	4.854E+02	4.093E+02	4.422E+02	4.153E+02	4.056E+02	4.074E+02	3.774E+02	3.731E+02	4.208E+02
	Std	1.315E+02	2.367E+02	2.183E+02	1.897E+02	2.025E+02	1.523E+02	1.590E+02	1.926E+02	1.657E+02	1.698E+02
f_{17}	Mean	2.231E+02	7.504E+01	1.072E+02	1.404E+02	1.268E+02	9.959E+01	1.359E+02	1.184E+02	9.497E+01	1.264E+02
	Std	7.667E+01	6.250E+01	8.164E+01	1.059E+02	1.022E+02	7.936E+01	1.073E+02	6.640E+01	8.056E+01	8.554E+01
f_{18}	Mean	1.459E+05	2.568E+04	3.040E+04	2.595E+04	2.608E+04	3.232E+04	2.249E+04	2.253E+04	2.419E+04	2.575E+04
	Std	6.886E+04	1.577E+04	3.044E+04	1.673E+04	1.675E+04	2.339E+04	1.979E+04	1.860E+04	1.750E+04	1.822E+04
f_{19}	Mean	1.468E+03	6.826E+01	7.224E+01	8.271E+01	2.137E+02	6.552E+01	2.073E+02	1.144E+02	5.408E+01	1.153E+02
	Std	1.176E+03	9.645E+01	1.204E+02	1.964E+02	9.425E+02	1.137E+02	6.932E+02	1.823E+02	4.995E+01	3.329E+02
f_{20}	Mean	2.485E+02	1.291E+02	1.264E+02	1.219E+02	1.249E+02	1.199E+02	1.210E+02	1.611E+02	1.290E+02	1.328E+02
	Std	9.764E+01	9.550E+01	1.169E+02	9.743E+01	1.052E+02	7.254E+01	1.099E+02	1.078E+02	9.483E+01	8.475E+01
f_{21}	Mean	2.410E+02	2.716E+02	2.670E+02	2.647E+02	2.588E+02	2.553E+02	2.543E+02	2.524E+02	2.499E+02	2.494E+02
	Std	7.941E+01	8.816E+00	6.413E+00	8.557E+00	6.459E+00	6.503E+00	6.672E+00	6.995E+00	8.102E+00	9.231E+00
f_{22}	Mean	6.289E+02	8.513E+02	1.576E+03	5.057E+02	9.561E+02	1.435E+03	7.650E+02	7.503E+02	7.706E+02	6.337E+02
	Std	1.063E+03	1.394E+03	1.612E+03	1.053E+03	1.343E+03	1.466E+03	1.228E+03	1.203E+03	1.239E+03	1.096E+03
f_{23}	Mean	4.159E+02	4.454E+02	4.413E+02	4.341E+02	4.216E+02	4.148E+02	4.061E+02	4.025E+02	4.016E+02	3.994E+02
	Std	2.521E+01	8.987E+00	8.036E+00	1.055E+01	1.033E+01	8.819E+00	1.137E+01	7.153E+00	6.379E+00	9.122E+00
f_{24}	Mean	4.864E+02	5.203E+02	5.190E+02	5.109E+02	5.008E+02	4.910E+02	4.841E+02	4.769E+02	4.740E+02	4.704E+02
	Std	2.022E+02	1.799E+01	1.018E+01	8.052E+00	8.127E+00	1.128E+01	7.683E+00	7.834E+00	8.109E+00	9.157E+00
f_{25}	Mean	3.843E+02	3.873E+02	3.867E+02	3.872E+02	3.870E+02	3.870E+02	3.871E+02	3.868E+02	3.869E+02	3.873E+02
	Std	7.261E-01	1.213E+00	1.833E+00	1.356E+00	1.806E+00	1.472E+00	9.534E-01	1.165E+00	1.216E+00	3.326E-01
f_{26}	Mean	4.237E+02	1.755E+03	1.759E+03	1.704E+03	1.635E+03	1.600E+03	1.569E+03	1.532E+03	1.530E+03	1.558E+03
	Std	4.844E+02	1.421E+02	1.139E+02	1.242E+02	1.151E+02	1.390E+02	1.075E+02	1.209E+02	1.056E+02	9.872E+01
f_{27}	Mean	5.120E+02	5.026E+02	5.027E+02	5.032E+02	5.020E+02	5.025E+02	5.015E+02	5.048E+02	5.046E+02	5.037E+02
	Std	4.572E+00	7.552E+00	6.264E+00	7.089E+00	5.569E+00	4.706E+00	5.470E+00	8.661E+00	5.965E+00	6.006E+00
f_{28}	Mean	4.027E+02	3.815E+02	3.794E+02	3.750E+02	3.848E+02	3.637E+02	3.677E+02	3.882E+02	3.706E+02	3.729E+02
	Std	3.061E+00	4.280E+01	4.567E+01	4.334E+01	3.501E+01	4.777E+01	5.353E+01	3.883E+01	5.022E+01	6.178E+01
f_{29}	Mean	6.100E+02	5.318E+02	5.250E+02	5.296E+02	5.279E+02	4.981E+02	5.184E+02	5.605E+02	5.253E+02	5.423E+02
	Std	9.597E+01	6.193E+01	5.840E+01	8.544E+01	7.582E+01	4.371E+01	5.772E+01	8.994E+01	7.893E+01	8.270E+01
f_{30}	Mean	9.600E+03	4.694E+03	4.632E+03	4.627E+03	5.059E+03	4.528E+03	3.800E+03	3.978E+03	5.001E+03	4.581E+03
	Std	2.420E+03	2.728E+03	2.279E+03	2.587E+03	2.809E+03	2.049E+03	2.144E+03	1.752E+03	2.698E+03	1.929E+03
Wilcoxon test	+/=/-		17/0/12	17/0/12	18/0/11	17/0/12	19/0/10	20/0/9	20/0/9	19/0/10	20/0/9
Friedman ranking		6.76	6.90	6.59	6.41	6.14	4.24	4.24	5.28	**4.21**	4.24
Final ranking		9	10	8	7	6	2	3	5	**1**	4

Table 5.2: Comparison of FOABC variants with number of terms $r = 8$ and q taked values from 0.1 to 0.9 with ABC

						$r = 8$					
Functions		ABC	$q = 0.1$	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
f_1	Mean	3.293E+02	1.762E+03	2.085E+03	1.703E+03	1.434E+03	2.681E+03	9.833E+02	2.032E+03	2.348E+03	3.062E+03
	Std	3.536E+02	3.489E+03	4.100E+03	2.662E+03	3.299E+03	4.487E+03	2.860E+03	4.230E+03	4.475E+03	4.601E+03
f_3	Mean	7.361E+04	3.570E+04	3.864E+04	3.766E+04	3.630E+04	3.817E+04	4.171E+04	3.934E+04	4.191E+04	4.164E+04
	Std	2.424E+04	9.242E+03	8.883E+03	9.476E+03	7.894E+03	1.026E+04	1.004E+04	1.202E+04	1.098E+04	1.126E+04
f_4	Mean	1.890E+01	8.914E+01	9.365E+01	9.457E+01	8.716E+01	8.448E+01	9.131E+01	8.843E+01	8.564E+01	8.948E+01
	Std	2.274E+01	2.249E+01	1.560E+01	1.626E+01	2.283E+01	2.015E+01	1.585E+01	1.238E+01	9.695E+00	1.123E+01
f_5	Mean	8.327E+01	7.853E+01	7.042E+01	6.301E+01	5.708E+01	5.014E+01	4.877E+01	4.453E+01	4.499E+01	4.467E+01
	Std	1.315E+01	8.257E+00	7.259E+00	6.155E+00	6.357E+00	7.139E+00	7.195E+00	8.045E+00	5.415E+00	7.346E+00
f_6	Mean	7.420E-10	2.340E-04	2.522E-04	6.013E-05	4.319E-05	1.946E-05	6.792E-05	5.952E-05	5.398E-05	2.098E-05
	Std	8.144E-10	1.237E-03	1.357E-03	1.519E-04	1.410E-04	5.004E-05	2.337E-04	2.946E-04	1.581E-04	5.646E-05
f_7	Mean	1.006E+02	1.254E+02	1.219E+02	1.120E+02	1.011E+02	9.296E+01	8.625E+01	8.265E+01	7.656E+01	7.865E+01
	Std	8.186E+00	7.194E+00	9.851E+00	8.389E+00	8.379E+00	7.991E+00	8.626E+00	8.044E+00	6.378E+00	6.751E+00
f_8	Mean	8.879E+01	6.865E+01	6.188E+01	5.696E+01	5.019E+01	5.132E+01	4.934E+01	4.986E+01	4.581E+01	4.575E+01
	Std	1.432E+01	7.789E+00	9.729E+00	6.374E+00	7.100E+00	7.289E+00	7.936E+00	7.887E+00	8.874E+00	8.239E+00
f_9	Mean	7.745E+02	7.408E-01	6.820E-01	3.829E-01	5.225E-01	8.748E-01	7.031E-01	4.745E-01	5.019E-01	1.751E-01
	Std	4.200E+02	1.215E+00	1.312E+00	4.681E-01	8.095E-01	1.259E+00	1.450E+00	5.751E-01	6.705E-01	2.260E-01
f_{10}	Mean	2.220E+03	2.970E+03	2.855E+03	2.708E+03	2.646E+03	2.715E+03	2.535E+03	2.626E+03	2.498E+03	2.450E+03
	Std	2.321E+02	3.525E+02	2.789E+02	2.845E+02	2.497E+02	3.512E+02	2.657E+02	2.218E+02	3.114E+02	3.196E+02
f_{11}	Mean	2.331E+02	5.267E+01	4.019E+01	3.198E+01	2.778E+01	2.290E+01	2.332E+01	2.045E+01	2.165E+01	2.779E+01
	Std	1.787E+02	2.922E+01	2.394E+01	2.223E+01	1.854E+01	1.250E+01	9.567E+00	1.318E+01	1.448E+01	1.882E+01
f_{12}	Mean	5.541E+05	3.753E+04	3.711E+04	5.016E+04	3.173E+04	3.425E+04	3.858E+04	4.859E+04	3.203E+04	4.203E+04
	Std	2.568E+05	2.242E+04	1.261E+04	7.932E+04	1.775E+04	4.257E+04	2.529E+04	7.659E+04	1.789E+04	3.116E+04
f_{13}	Mean	8.049E+03	1.195E+04	1.837E+04	1.668E+04	9.991E+03	9.511E+03	8.718E+03	1.072E+04	1.236E+04	1.309E+04
	Std	7.078E+03	1.580E+04	1.841E+04	1.686E+04	1.245E+04	1.075E+04	1.287E+04	1.416E+04	1.513E+04	1.791E+04
f_{14}	Mean	4.534E+04	7.609E+01	8.620E+01	1.008E+02	9.161E+01	5.542E+01	7.599E+01	6.158E+01	7.414E+01	5.992E+01
	Std	3.859E+04	2.924E+01	7.235E+01	1.136E+02	9.642E+01	1.379E+01	3.871E+01	2.224E+01	5.898E+01	2.547E+01
f_{15}	Mean	1.301E+03	6.136E+02	4.348E+02	2.389E+02	2.681E+02	1.124E+02	1.764E+02	5.085E+02	2.377E+02	1.039E+03
	Std	9.227E+02	1.858E+03	9.981E+02	3.544E+02	4.892E+02	6.725E+01	2.824E+02	1.077E+03	8.642E+02	4.985E+03
f_{16}	Mean	6.639E+02	3.886E+02	4.435E+02	4.778E+02	3.672E+02	4.291E+02	3.346E+02	3.559E+02	4.153E+02	4.123E+02
	Std	1.315E+02	1.836E+02	1.818E+02	1.884E+02	1.550E+02	1.861E+02	1.969E+02	1.831E+02	1.492E+02	1.501E+02
f_{17}	Mean	2.231E+02	1.201E+02	8.914E+01	1.076E+02	1.249E+02	1.162E+02	1.072E+02	8.717E+01	1.231E+02	1.134E+02
	Std	7.667E+01	8.095E+01	7.778E+01	8.463E+01	1.054E+02	9.110E+01	7.883E+01	6.976E+01	8.460E+01	7.463E+01
f_{18}	Mean	1.459E+05	3.591E+04	2.854E+04	2.763E+04	2.169E+04	2.319E+04	3.569E+04	2.314E+04	1.967E+04	2.501E+04
	Std	6.886E+04	2.727E+04	1.853E+04	1.964E+04	1.539E+04	1.718E+04	4.183E+04	1.790E+04	1.298E+04	1.981E+04
f_{19}	Mean	1.468E+03	8.750E+01	2.544E+02	3.173E+02	3.345E+02	6.563E+01	6.196E+01	4.930E+01	4.344E+01	1.687E+02
	Std	1.176E+03	1.930E+02	8.389E+02	1.214E+03	1.503E+03	8.788E+01	6.794E+01	4.462E+01	3.829E+01	6.790E+02
f_{20}	Mean	2.485E+02	1.108E+02	1.198E+02	1.336E+02	1.015E+02	9.710E+01	1.426E+02	1.608E+02	1.082E+02	1.474E+02
	Std	9.764E+01	1.063E+02	1.160E+02	8.675E+01	1.045E+02	8.013E+01	9.411E+01	1.107E+02	8.982E+01	1.056E+02
f_{21}	Mean	2.410E+02	2.709E+02	2.668E+02	2.580E+02	2.541E+02	2.516E+02	2.531E+02	2.522E+02	2.508E+02	2.533E+02
	Std	7.941E+01	8.204E+00	5.931E+00	7.267E+00	7.240E+00	9.024E+00	7.853E+00	7.721E+00	8.596E+00	8.412E+00
f_{22}	Mean	6.289E+02	7.367E+02	6.016E+02	1.095E+03	1.102E+03	1.214E+03	8.303E+02	5.491E+02	9.851E+02	1.183E+03
	Std	1.063E+03	1.304E+03	1.144E+03	1.449E+03	1.456E+03	1.401E+03	1.246E+03	1.024E+03	1.381E+03	1.355E+03
f_{23}	Mean	4.159E+02	4.407E+02	4.342E+02	4.241E+02	4.141E+02	4.068E+02	4.016E+02	4.014E+02	3.974E+02	3.982E+02
	Std	2.521E+01	9.130E+00	9.028E+00	7.187E+00	9.290E+00	8.596E+00	7.341E+00	9.267E+00	6.948E+00	8.449E+00
f_{24}	Mean	4.864E+02	5.271E+02	5.136E+02	5.057E+02	4.901E+02	4.827E+02	4.803E+02	4.735E+02	4.711E+02	4.723E+02
	Std	2.022E+02	1.015E+01	1.090E+01	9.581E+00	1.057E+01	8.373E+00	6.869E+00	9.566E+00	1.080E+01	9.258E+00
f_{25}	Mean	3.843E+02	3.870E+02	3.864E+02	3.871E+02	3.873E+02	3.866E+02	3.870E+02	3.871E+02	3.871E+02	3.872E+02
	Std	7.261E-01	1.656E+00	1.803E+00	1.492E+00	9.359E-01	1.635E+00	1.293E+00	1.292E+00	1.062E+00	2.254E-01
f_{26}	Mean	4.237E+02	1.769E+03	1.674E+03	1.671E+03	1.643E+03	1.517E+03	1.506E+03	1.510E+03	1.538E+03	1.579E+03
	Std	4.844E+02	1.400E+02	1.759E+02	9.360E+01	1.031E+02	1.130E+02	9.206E+01	1.231E+02	9.199E+01	8.702E+01
f_{27}	Mean	5.120E+02	5.031E+02	5.035E+02	5.013E+02	5.035E+02	5.034E+02	5.045E+02	5.044E+02	5.053E+02	5.036E+02
	Std	4.572E+00	5.949E+00	8.183E+00	8.930E+00	4.609E+00	6.350E+00	6.945E+00	8.319E+00	7.633E+00	6.000E+00
f_{28}	Mean	4.027E+02	3.871E+02	3.715E+02	3.770E+02	3.753E+02	3.643E+02	3.836E+02	3.915E+02	3.519E+02	3.559E+02
	Std	3.061E+00	4.510E+01	4.404E+01	3.891E+01	4.054E+01	4.450E+01	4.683E+01	4.699E+01	4.855E+01	6.768E+01
f_{29}	Mean	6.100E+02	5.074E+02	5.255E+02	5.212E+02	5.355E+02	5.205E+02	5.317E+02	5.400E+02	5.181E+02	5.193E+02
	Std	9.597E+01	8.707E+01	6.373E+01	6.803E+01	6.618E+01	6.251E+01	5.398E+01	7.230E+01	8.039E+01	7.437E+01
f_{30}	Mean	9.600E+03	4.683E+03	4.036E+03	4.851E+03	4.360E+03	5.230E+03	5.102E+03	5.064E+03	4.732E+03	5.099E+03
	Std	2.420E+03	2.444E+03	2.142E+03	2.453E+03	2.351E+03	2.746E+03	2.956E+03	2.651E+03	2.479E+03	2.893E+03
Wilcoxon test	+/=/-		17/0/12	18/0/11	17/0/12	18/0/11	20/0/9	20/0/9	21/0/8	20/0/9	20/0/9
Friedman ranking		6.90	6.62	6.59	6.48	5.31	4.55	4.97	4.55	**3.93**	5.10
Final ranking		10	9	8	7	6	2	4	3	**1**	5

Table 5.3: Comparison of FOABC variants with number of terms $r = 12$ and q taked values from 0.1 to 0.9 with ABC

			$r = 12$								
Functions		ABC	$q = 0.1$	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
f_1	Mean	3.293E+02	3.326E+03	8.278E+02	1.190E+03	2.849E+03	1.456E+03	1.471E+03	1.577E+03	7.784E+02	2.111E+03
	Std	3.536E+02	4.517E+03	1.492E+03	2.578E+03	4.886E+03	2.765E+03	3.320E+03	3.274E+03	2.497E+03	3.998E+03
f_3	Mean	7.361E+04	3.911E+04	4.165E+04	3.990E+04	3.918E+04	3.842E+04	4.250E+04	4.093E+04	4.304E+04	4.683E+04
	Std	2.424E+04	1.061E+04	1.200E+04	1.033E+04	1.166E+04	1.010E+04	1.062E+04	9.949E+03	1.130E+04	1.456E+04
f_4	Mean	1.890E+01	9.162E+01	9.128E+01	8.810E+01	9.620E+01	9.181E+01	8.853E+01	8.984E+01	9.091E+01	8.611E+01
	Std	2.274E+01	1.571E+01	1.565E+01	1.235E+01	1.884E+01	1.562E+01	1.442E+01	1.270E+01	1.364E+01	5.832E+00
f_5	Mean	8.327E+01	7.741E+01	6.837E+01	6.064E+01	5.353E+01	4.656E+01	4.557E+01	4.577E+01	4.600E+01	4.212E+01
	Std	1.315E+01	7.978E+00	7.709E+00	6.970E+00	7.498E+00	7.219E+00	6.333E+00	5.077E+00	6.332E+00	8.087E+00
f_6	Mean	7.420E-10	4.165E-05	5.479E-04	5.132E-04	1.907E-05	2.764E-04	2.961E-05	2.313E-05	7.209E-05	6.186E-05
	Std	8.144E-10	1.381E-04	2.255E-03	2.450E-03	6.595E-05	1.263E-03	8.405E-05	5.918E-05	2.837E-04	1.864E-04
f_7	Mean	1.006E+02	1.219E+02	1.194E+02	1.060E+02	9.703E+01	8.779E+01	8.349E+01	8.001E+01	7.877E+01	7.856E+01
	Std	8.186E+00	7.473E+00	8.995E+00	8.916E+00	6.705E+00	6.190E+00	6.695E+00	6.739E+00	6.967E+00	7.700E+00
f_8	Mean	8.879E+01	6.623E+01	6.057E+01	5.620E+01	5.131E+01	4.973E+01	4.771E+01	4.881E+01	4.718E+01	4.982E+01
	Std	1.432E+01	6.782E+00	7.349E+00	7.027E+00	7.483E+00	6.435E+00	7.195E+00	8.756E+00	7.980E+00	8.467E+00
f_9	Mean	7.745E+02	1.211E+00	6.017E-01	6.407E-01	4.384E-01	2.263E-01	4.466E-01	8.415E-01	3.323E-01	4.087E-01
	Std	4.200E+02	2.734E+00	1.053E+00	9.649E-01	6.639E-01	4.482E-01	4.690E-01	1.641E+00	4.254E-01	1.320E+00
f_{10}	Mean	2.220E+03	2.870E+03	2.795E+03	2.778E+03	2.682E+03	2.616E+03	2.612E+03	2.499E+03	2.418E+03	2.412E+03
	Std	2.321E+02	2.187E+02	3.556E+02	2.321E+02	2.031E+02	2.754E+02	2.722E+02	3.002E+02	3.542E+02	2.996E+02
f_{11}	Mean	2.331E+02	4.113E+01	2.656E+01	2.779E+01	2.309E+01	2.448E+01	2.219E+01	2.078E+01	1.819E+01	1.779E+01
	Std	1.787E+02	2.803E+01	1.644E+01	2.177E+01	1.246E+01	1.469E+01	1.118E+01	1.021E+01	8.268E+00	9.121E+00
f_{12}	Mean	5.541E+05	4.395E+04	5.679E+04	3.247E+04	3.326E+04	4.607E+04	6.371E+04	5.778E+04	9.548E+04	6.558E+04
	Std	2.568E+05	3.642E+04	1.201E+05	4.190E+04	2.469E+04	6.344E+04	8.131E+04	1.020E+05	1.891E+05	6.382E+04
f_{13}	Mean	8.049E+03	2.064E+04	1.123E+04	2.010E+04	1.534E+04	1.421E+04	1.675E+04	1.325E+04	1.173E+04	1.150E+04
	Std	7.078E+03	2.206E+04	1.278E+04	2.057E+04	1.826E+04	1.464E+04	1.842E+04	1.755E+04	1.563E+04	1.365E+04
f_{14}	Mean	4.534E+04	9.851E+01	9.693E+01	5.800E+01	7.787E+01	5.955E+01	5.525E+01	8.126E+01	6.055E+01	6.936E+01
	Std	3.859E+04	9.987E+01	1.429E+02	1.633E+01	3.943E+01	2.892E+01	1.403E+01	4.476E+01	2.888E+01	5.820E+01
f_{15}	Mean	1.301E+03	1.812E+03	3.408E+02	9.482E+02	1.833E+02	2.212E+02	1.206E+02	1.091E+02	7.544E+02	3.575E+02
	Std	9.227E+02	7.453E+03	4.808E+02	4.163E+03	2.052E+02	5.543E+02	1.559E+02	1.453E+02	2.956E+03	1.097E+03
f_{16}	Mean	6.639E+02	4.057E+02	3.981E+02	4.257E+02	4.308E+02	4.408E+02	3.659E+02	4.248E+02	4.113E+02	4.574E+02
	Std	1.315E+02	1.672E+02	1.830E+02	1.521E+02	1.505E+02	1.496E+02	1.502E+02	1.570E+02	1.809E+02	1.873E+02
f_{17}	Mean	2.231E+02	8.190E+01	9.895E+01	1.311E+02	1.128E+02	1.158E+02	1.081E+02	9.481E+01	1.100E+02	9.542E+01
	Std	7.667E+01	7.632E+01	7.681E+01	9.327E+01	8.001E+01	9.235E+01	7.699E+01	6.712E+01	8.625E+01	6.856E+01
f_{18}	Mean	1.459E+05	2.600E+04	3.462E+04	2.747E+04	2.643E+04	2.353E+04	2.843E+04	2.961E+04	3.073E+04	2.529E+04
	Std	6.886E+04	2.174E+04	2.776E+04	2.207E+04	1.667E+04	1.275E+04	2.579E+04	2.508E+04	4.419E+04	2.094E+04
f_{19}	Mean	1.468E+03	5.782E+01	1.104E+02	4.931E+01	4.579E+01	7.785E+01	1.942E+03	3.066E+02	3.258E+01	3.001E+01
	Std	1.176E+03	5.007E+01	3.327E+02	3.042E+01	2.879E+01	1.617E+02	9.924E+03	1.128E+03	2.080E+01	1.969E+01
f_{20}	Mean	2.485E+02	1.520E+02	1.418E+02	1.248E+02	1.393E+02	9.828E+01	1.205E+02	1.335E+02	1.426E+02	1.552E+02
	Std	9.764E+01	9.114E+01	1.036E+02	1.046E+02	1.025E+02	9.465E+01	1.020E+02	9.342E+01	9.766E+01	1.088E+02
f_{21}	Mean	2.410E+02	2.719E+02	2.643E+02	2.594E+02	2.555E+02	2.521E+02	2.513E+02	2.504E+02	2.463E+02	2.521E+02
	Std	7.941E+01	8.325E+00	6.993E+00	7.418E+00	8.020E+00	8.111E+00	6.985E+00	7.642E+00	8.832E+00	7.287E+00
f_{22}	Mean	6.289E+02	8.728E+02	8.936E+02	1.200E+03	7.728E+02	7.453E+02	9.668E+02	1.203E+03	7.726E+02	1.033E+03
	Std	1.063E+03	1.427E+03	1.352E+03	1.474E+03	1.249E+03	1.203E+03	1.353E+03	1.379E+03	1.140E+03	1.345E+03
f_{23}	Mean	4.159E+02	4.353E+02	4.347E+02	4.205E+02	4.137E+02	4.049E+02	4.031E+02	4.006E+02	3.979E+02	4.012E+02
	Std	2.521E+01	1.522E+01	7.764E+00	8.142E+00	7.331E+00	7.797E+00	9.345E+00	7.307E+00	8.223E+00	6.694E+00
f_{24}	Mean	4.864E+02	5.260E+02	5.113E+02	4.974E+02	4.888E+02	4.801E+02	4.762E+02	4.742E+02	4.695E+02	4.712E+02
	Std	2.022E+02	1.084E+01	1.179E+01	1.028E+01	8.612E+00	8.373E+00	9.851E+00	1.019E+01	1.046E+01	8.257E+00
f_{25}	Mean	3.843E+02	3.869E+02	3.870E+02	3.871E+02	3.873E+02	3.869E+02	3.874E+02	3.871E+02	3.873E+02	3.872E+02
	Std	7.261E-01	1.780E+00	1.456E+00	1.252E+00	1.089E+00	1.487E+00	8.437E-01	1.097E+00	3.224E-01	2.716E-01
f_{26}	Mean	4.237E+02	1.740E+03	1.650E+03	1.635E+03	1.578E+03	1.540E+03	1.551E+03	1.533E+03	1.524E+03	1.553E+03
	Std	4.844E+02	1.215E+02	2.780E+02	1.256E+02	1.025E+02	1.001E+02	1.117E+02	8.082E+01	1.169E+02	1.134E+02
f_{27}	Mean	5.120E+02	5.031E+02	5.032E+02	5.019E+02	5.016E+02	5.049E+02	5.050E+02	5.029E+02	5.043E+02	5.035E+02
	Std	4.572E+00	5.667E+00	4.787E+00	5.718E+00	4.380E+00	7.105E+00	7.640E+00	7.377E+00	6.010E+00	7.532E+00
f_{28}	Mean	4.027E+02	3.897E+02	3.924E+02	3.839E+02	3.652E+02	3.724E+02	3.665E+02	3.787E+02	3.667E+02	3.523E+02
	Std	3.061E+00	3.279E+01	3.436E+01	4.221E+01	4.710E+01	4.260E+01	4.555E+01	5.471E+01	5.800E+01	6.663E+01
f_{29}	Mean	6.100E+02	5.256E+02	5.032E+02	5.043E+02	5.265E+02	4.895E+02	5.125E+02	5.256E+02	5.161E+02	5.149E+02
	Std	9.597E+01	6.241E+01	6.375E+01	5.959E+01	7.229E+01	7.306E+01	7.505E+01	7.751E+01	7.210E+01	8.883E+01
f_{30}	Mean	9.600E+03	5.686E+03	5.356E+03	4.764E+03	4.394E+03	4.215E+03	4.614E+03	6.073E+03	4.741E+03	4.638E+03
	Std	2.420E+03	2.804E+03	2.713E+03	2.482E+03	2.507E+03	2.321E+03	2.025E+03	2.852E+03	2.473E+03	2.000E+03
Wilcoxon test	+/=/-		16/0/13	17/0/12	17/0/12	19/0/10	20/0/9	19/0/10	20/0/9	20/0/9	20/0/9
Friedman ranking		6.79	7.14	6.55	6.07	5.34	4.48	4.76	4.90	**4.41**	4.55
Final ranking		9	10	8	7	6	2	4	5	**1**	3

Based on the comparison results in Tables 5.1 - 5.3, firstly, most of the FOABC variants have better results than the original ABC algorithm on the considered benchmark functions. Similar conclusion is suggested by the results of Friedman tests in each case of r. The basic ABC ranked the ninth among 10 compared methods when fractional order $r = 4$ and 12. Besides, the basic ABC stays in the last position when $r = 8$. In this case, the effectiveness of the proposed improving strategies can be demonstrated. Furthermore, for all the considered values of r, when the fractional order q is defined too small (i.e., $[0.1, 0.4]$), the corresponding performance is not as satisfying as the other variants.

In Table 5.1, when $r = 4$, the variants with $q \in [0.5, 0.9]$ have close in terms of mean and Std values on most of the functions. Meanwhile, their average rankings given by Friedman test are the top five. Moreover, according to the Wilcoxon test, all the variants of FOABC perform better than the basic ABC on over half of the benchmark functions. As for the case $r = 8$, FOABC variants with q in the same range of $[0.5, 0.9]$ have similar results and outstanding positions in Table 5.2. And when fractional order q is larger than 0.5, the corresponding algorithms can obtain superior solutions than ABC algorithm on more than 19 test functions, which can be found in the line of Wilcoxon test. The results in Table 5.3 suggests a similar conclusion.Moreover, the versions with $q = 0.8$ obtain the best rankings in all the three comparison tables. Therefore, for the proposed algorithm, the value of fractional order q cannot be set too small.

Another comparison is made to find out the optimal configuration for FOABC. In Table 5.4, the FOABC variants who ranked the best in the three aforementioned tables are compared together. The result indicates that $FOABC_{r=12,q=0.8}$ exceeds the other two variants. Therefore, this setting will be used to compare with other competitive algorithms in the later sections.

Table 5.4: Friedman test results of three competitive FOABC variants

Number of terms r	Derivative order q	Average ranking
4	0.8	2.21
8	0.8	1.93
12	**0.8**	**1.86**

5.3.2 Comparison with ABC variants

In this section, FOABC is compared with the basic ABC and five latest ABC variants on 29 CEC 2017 benchmarks with $D = 10$, 30, and 50. In order to make fair comparisons, the control parameters of the competitors are set the same as their original papers suggested which are presented in Table 5.5. In addition, the swarm size is set the same $SN = 50$ for all the compared ABC variants. The comparison results in terms of the mean and Std of the function errors are shown in Tables 5.6-5.8. For each function, the best results are marked in **boldface** by comparing the mean values.

Table 5.5: Parameter settings of FOABC and compared ABC algorithms

Algorithm	Parameter setting
ABC (Karaboga, 2005)	$limit = SN \times D$
NSABC (Wang *et al.*, 2020)	$limit = 100$, $k = 10$, $C = 1.5$
iff-ABC (Aslan *et al.*, 2020)	$limit = SN \cdot D$, $pr = 80$
MGABC (Zhou *et al.*, 2021a)	$limit = 100$, $MR = 0.5$, $q = 0.1$, $p = 0.1$
ILTD_ABC (Gao *et al.*, 2019)	$limit = 100$
sdABC (Chen *et al.*, 2019b)	$limit = SN \cdot D$, $pa_{min} = 0.2$
FOABC	$limit = SN \cdot D$, $CR = 0.8$, $r = 12$, $q = 0.8$

Table 5.6 presents the results with $D = 10$, where FOABC performs outstandingly, especially in solving the uni-modal and hybrid problems. Specifically, the number of the minimum errors achieved by FOABC is 10 while the second best is 7 by *iff*-ABC. In solving the uni-modal functions (i.e., f_1 and f_3), the superiority of FOABC is significant compared to the other competitors. More precisely, the errors of FOABC on these two functions are much closer to zero while all the other algorithms have errors bigger than 10^2. The exploitation ability of algorithms can be tested on the uni-modal problem as there is a unique optimum. In this context, the comparison result verifies the exploitation capability of FOABC. Hence, among the multi-modal functions ($f_4 - f_{10}$), NSABC obtains the best results on 4 of them whereas MGABC attains two. As for the hybrid functions ($f_{11} - f_{20}$), the advantage of FOABC is notable by comparing not only the values of errors but also the number of functions in boldface. FOABC achieves the best on 7 among 10 hybrid functions. Meanwhile, FOABC doesn't keep the same superiority in solving the rest benchmarks which can be explained according to the No Free Lunch Theorem. The *iff*-ABC algorithm performs the best on composition

Table 5.6: Comparison between FOABC and other ABC variants with $D = 10$

Functions		ABC		NSABC		*iff*-ABC		MGABC		ILTD_ABC		sdABC		FOABC$_{r=12,q=0.8}$
f_1	Mean	3.07E+02	+	2.52E+03	+	2.64E+02	+	4.20E+02	+	9.65E+03	+	1.87E+09	+	**4.33E-09**
	Std	2.46E+02		2.55E+03		3.12E+02		5.72E+02		2.40E+04		3.30E+09		**1.50E-08**
f_3	Mean	6.26E+02	+	3.17E+03	+	4.19E+03	+	1.32E+03	+	2.16E+03	+	4.77E+03	+	**1.71E-14**
	Std	2.97E+02		2.49E+03		3.07E+03		6.60E+02		9.26E+02		3.36E+03		**3.04E-14**
f_4	Mean	2.42E-01	-	3.05E+00	+	**1.58E-01**	-	5.50E+00	+	5.32E+00	+	1.40E+01	+	8.01E-01
	Std	1.89E-01		1.38E+00		**2.26E-01**		2.05E+00		1.22E+00		2.59E+01		4.66E-01
f_5	Mean	7.37E+00	+	**3.35E+00**	-	7.30E+00	+	4.67E+00	-	5.00E+00	+	4.53E+01	+	4.68E+00
	Std	1.79E+00		**1.13E+00**		2.08E+00		1.18E+00		1.96E+00		1.76E+01		1.50E+00
f_6	Mean	3.17E-09	-	3.41E-14	-	5.95E-09	-	**0.00E+00**	-	2.91E-04	+	4.55E+01	+	3.78E-07
	Std	5.40E-09		5.30E-14		7.33E-09		**0.00E+00**		1.53E-04		1.05E+01		2.07E-06
f_7	Mean	1.78E+01	+	**1.34E+01**	-	1.74E+01	+	1.57E+01	+	1.57E+01	+	1.06E+02	+	1.40E+01
	Std	2.32E+00		**1.71E+00**		3.47E+00		1.07E+00		2.14E+00		1.94E+01		1.51E+00
f_8	Mean	8.09E+00	+	**3.71E+00**	-	9.14E+00	+	4.88E+00	-	5.05E+00	-	4.27E+01	+	5.21E+00
	Std	2.53E+00		**1.35E+00**		2.12E+00		1.17E+00		2.81E+00		1.22E+01		1.88E+00
f_9	Mean	5.26E-03	-	1.12E-05	-	5.73E-02	+	**3.46E-06**	-	9.36E-05	-	4.09E+01	+	1.51E-02
	Std	1.71E-02		3.27E-05		1.07E-01		**7.46E-06**		1.47E-04		9.31E+01		8.29E-02
f_{10}	Mean	2.72E+02	+	**1.67E+02**	-	2.11E+02	+	1.92E+02	+	1.67E+02	-	1.06E+03	+	1.87E+02
	Std	8.96E+01		**1.28E+02**		9.08E+01		8.27E+01		1.38E+02		5.54E+02		9.31E+01
f_{11}	Mean	3.86E+00	+	3.57E+00	+	4.97E+00	+	2.31E+00	+	3.10E+00	+	1.16E+01	+	**1.13E+00**
	Std	1.93E+00		1.53E+00		1.91E+00		1.26E+00		1.18E+00		1.39E+01		**7.72E-01**
f_{12}	Mean	3.89E+04	+	2.49E+04	+	2.33E+04	+	3.55E+04	+	9.75E+03	+	3.43E+07	+	**6.74E+01**
	Std	2.32E+04		1.61E+04		1.45E+04		6.57E+04		5.75E+03		7.07E+07		**1.33E+02**
f_{13}	Mean	6.83E+02	+	5.71E+03	+	4.07E+02	+	6.11E+03	+	7.36E+03	+	2.86E+02	+	**4.78E+00**
	Std	6.10E+02		6.83E+03		3.65E+02		3.71E+03		5.30E+03		1.28E+03		**1.99E+00**
f_{14}	Mean	1.81E+02	+	1.33E+03	+	1.57E+02	+	1.39E+03	+	2.41E+03	+	9.37E+01	+	**1.65E+00**
	Std	1.75E+02		2.04E+03		2.28E+02		1.66E+03		1.38E+03		4.55E+02		**1.13E+00**
f_{15}	Mean	1.69E+02	+	8.48E+02	+	1.61E+02	+	4.43E+02	+	1.38E+03	+	2.43E+01	+	**5.95E-01**
	Std	2.10E+02		1.58E+03		3.19E+02		7.51E+02		9.98E+02		6.10E+01		**8.15E-01**
f_{16}	Mean	8.88E+00	+	1.11E+01	+	1.31E+01	+	**2.24E+00**	-	2.74E+01	+	1.84E+02	+	3.12E+00
	Std	1.22E+01		2.52E+01		3.00E+01		**3.24E+00**		5.19E+01		1.19E+02		4.70E+00
f_{17}	Mean	2.89E+00	+	3.28E+00	+	**2.03E+00**	-	3.74E+00	+	1.46E+01	+	1.13E+02	+	2.72E+00
	Std	1.74E+00		5.76E+00		**8.07E-01**		5.15E+00		1.42E+01		4.18E+01		5.13E+00
f_{18}	Mean	1.26E+03	+	4.61E+03	+	1.16E+03	+	1.15E+03	+	6.64E+03	+	3.55E+03	+	**3.74E-01**
	Std	9.39E+02		4.10E+03		9.51E+02		1.21E+03		6.19E+03		1.02E+04		**4.50E-01**
f_{19}	Mean	8.69E+01	+	3.28E+03	+	1.09E+02	+	7.59E+02	+	4.71E+03	+	2.40E+02	+	**5.49E-02**
	Std	1.16E+02		3.18E+03		1.87E+02		9.07E+02		3.91E+03		1.30E+03		**1.82E-01**
f_{20}	Mean	3.14E-01	-	2.28E+00	+	9.21E-01	+	**4.34E-02**	-	7.20E-01	+	1.24E+02	+	4.44E-01
	Std	4.11E-01		5.14E+00		5.75E-01		**9.95E-02**		2.20E+00		7.00E+01		5.51E-01
f_{21}	Mean	1.13E+02	-	1.15E+02	-	**1.01E+02**	-	1.13E+02	-	1.76E+02	+	1.12E+02	-	1.58E+02
	Std	1.99E+01		3.62E+01		**2.72E+01**		2.62E+01		4.85E+01		3.00E+01		5.46E+01
f_{22}	Mean	**6.18E+01**	-	8.16E+01	-	6.36E+01	-	9.49E+01	-	1.00E+02	+	5.33E+02	+	9.61E+01
	Std	**3.38E+01**		3.84E+01		3.34E+01		2.04E+01		3.34E-01		2.65E+02		1.76E+01
f_{23}	Mean	**2.87E+02**	-	3.08E+02	-	3.04E+02	-	3.08E+02	-	3.09E+02	+	3.16E+02	+	3.09E+02
	Std	**8.25E+01**		2.93E+00		5.75E+01		1.65E+00		2.47E+00		9.12E+00		3.17E+00
f_{24}	Mean	1.01E+02	-	2.37E+02	-	**9.00E+01**	-	1.62E+02	-	3.02E+02	-	1.84E+02	-	3.39E+02
	Std	2.50E+01		1.14E+02		**2.53E+01**		1.01E+02		9.19E+01		1.20E+02		2.54E+00
f_{25}	Mean	1.86E+02	-	4.12E+02	-	**1.53E+02**	-	4.22E+02	+	4.18E+02	+	4.40E+02	+	4.12E+02
	Std	1.05E+02		2.14E+01		**6.78E+01**		2.20E+01		2.27E+01		6.48E+01		2.19E+01
f_{26}	Mean	1.20E+02	-	2.76E+02	-	**1.17E+02**	-	2.43E+02	-	2.73E+02	-	5.27E+02	+	3.09E+02
	Std	1.00E+02		9.81E+01		**9.94E+01**		1.04E+02		7.85E+01		4.40E+02		2.56E+01
f_{27}	Mean	3.84E+02	-	**3.72E+02**	-	3.94E+02	+	3.92E+02	+	3.91E+02	+	3.94E+02	+	3.90E+02
	Std	5.12E+01		**7.45E-01**		2.03E+00		1.93E+00		1.48E+00		1.08E+01		1.34E+00
f_{28}	Mean	2.57E+02	-	4.33E+02	+	**2.52E+02**	-	3.28E+02	-	4.11E+02	+	5.22E+02	+	3.45E+02
	Std	1.08E+02		9.20E+01		**1.05E+02**		6.79E+01		1.68E+02		1.24E+02		1.01E+02
f_{29}	Mean	2.49E+02	+	2.47E+02	+	2.47E+02	+	2.59E+02	+	2.56E+02	+	2.92E+02	+	**2.35E+02**
	Std	3.05E+01		8.33E+00		2.72E+01		1.17E+01		1.57E+01		6.72E+01		**3.56E+00**
f_{30}	Mean	1.76E+04	+	**3.70E+02**	-	1.24E+04	+	3.05E+04	+	1.68E+04	+	1.04E+06	+	7.41E+02
	Std	1.56E+04		**1.68E+02**		2.12E+04		2.68E+04		2.76E+04		2.67E+06		2.79E+02
Total	+/=/-	17/0/12		15/0/14		19/0/10		17/0/12		24/0/5		27/0/2		

functions ($f_{21} - f_{30}$). Nevertheless, it is worth mentioning that all the algorithms involved in this type of problem obtain similar results.

The advantage of FOABC is also remarkable based on the results of the Wilcoxon tests. Compared to the original ABC algorithm, FOABC attains better solutions on 17 out of 29 functions. And it outperforms NSABC on 15 problems. Comparing to *iff*-ABC and MGABC, FOABC surpasses them on 19 and 17 functions, respectively. ILTD_ABC fails to achieve better results than FOABC does on 24 functions. Moreover, FOABC achieves better solutions than sdABC does on 27 functions.

Table 5.7 shows the comparison at $D = 30$, where FOABC obtains the best results on 10 functions. And it is also followed by NSABC. Note that the difficulty of searching will increase when the size of dimensions augments and the swarm size keeps the same. Accordingly, FOABC loses a bit of superiority in solving the uni-modal functions. Nevertheless, the performance of FOABC on multi-modal functions becomes better as it achieves the best results on 2 functions. As for solving the hybrid functions, FOABC still possesses significant advantages compared to the other methods. There are 6 out of 10 problems where FOABC obtains the smallest errors. The results of the basic ABC algorithm and NSABC has slightly superiorities on composition functions.

Furthermore, the Wilcoxon test results also justify the outstanding performance of the proposed algorithm. More precisely, FOABC outperforms the basic ABC on 20 out of 29 problems. And compared to NSABC, there are 17 functions that FOABC obtains better solutions. FOABC performs better than *iff*-ABC on 20 problems. Meanwhile, it exceeds MGABC and ILTD_ABC on 19 and 22 problems, respectively. Moreover, the sdABC algorithm fails to overcome the proposed algorithm in this comparison.

Table 5.8 presents the comparison with $D = 50$, where FOABC manages to maintain its advantages as it achieves the minimum errors on 14 functions. It is interesting to note that this number is better than the previous cases. For solving uni-modal problems, each of FOABC and MGABC performs the best on 1 uni-modal function. And the performance of FOABC algorithm on multi-modal functions is similar to the previous dimension cases. Meanwhile, NSABC suffers from the effects of increasing dimensions in solving multi-modal problems. In addition, there are still 6 hybrid functions in f_{11}-f_{20} that FOABC achieves the

Table 5.7: Comparison between FOABC and other ABC variants with $D = 30$

Functions		ABC		NSABC		$iff-$ABC		MGABC		ILTD_ABC		sdABC		FOABC$_{r=12,q=0.8}$
f_1	Mean	1.73E+02	-	2.38E+03	+	**1.29E+02**	-	1.71E+03	+	6.76E+03	+	7.90E+09	+	1.50E+03
	Std	1.32E+02		4.02E+03		**1.46E+02**		1.37E+03		8.77E+03		4.91E+09		2.35E+03
f_3	Mean	7.69E+04	+	5.47E+04	+	1.13E+05	+	4.13E+04	+	5.24E+04	+	5.37E+04	+	**1.10E+04**
	Std	1.97E+04		1.20E+04		1.52E+04		6.83E+03		7.66E+03		2.24E+04		**5.59E+03**
f_4	Mean	2.38E+01	-	4.57E+01	-	**1.72E+01**	-	9.69E+01	+	7.38E+01	-	1.93E+03	+	8.71E+01
	Std	2.49E+01		2.82E+01		**2.53E+01**		2.16E+01		1.75E+01		2.64E+03		1.76E+01
f_5	Mean	8.23E+01	+	4.13E+01	+	8.42E+01	+	5.95E+01	+	6.78E+01	+	1.90E+02	+	**3.73E+01**
	Std	1.01E+01		7.09E+00		1.53E+01		1.03E+01		9.96E+00		5.23E+01		**9.07E+00**
f_6	Mean	4.80E-10	-	**2.65E-14**	-	3.19E-09	-	1.14E-13	-	1.34E-03	-	3.24E+01	+	3.55E-03
	Std	5.32E-10		**8.28E-14**		3.57E-09		0.00E+00		4.69E-04		2.26E+01		1.18E-02
f_7	Mean	9.87E+01	+	**6.94E+01**	-	9.89E+01	+	8.88E+01	+	1.18E+02	+	3.46E+02	+	7.84E+01
	Std	1.09E+01		**9.02E+00**		1.05E+01		9.94E+00		3.66E+01		1.18E+02		9.53E+00
f_8	Mean	9.20E+01	+	**3.88E+01**	-	9.14E+01	+	5.72E+01	+	7.30E+01	+	1.75E+02	+	4.15E+01
	Std	1.43E+01		**6.79E+00**		1.40E+01		9.65E+00		1.66E+01		5.20E+01		1.01E+01
f_9	Mean	6.33E+02	+	7.93E+00	+	6.92E+02	+	5.73E+01	+	1.24E+02	+	6.92E+03	+	**6.11E+00**
	Std	3.85E+02		8.46E+00		3.97E+02		5.79E+01		2.81E+02		1.96E+03		**7.86E+00**
f_{10}	Mean	2.33E+03	+	2.40E+03	+	2.24E+03	-	2.14E+03	-	**2.11E+03**	-	5.82E+03	+	2.30E+03
	Std	2.03E+02		4.85E+02		2.19E+02		2.17E+02		**4.30E+02**		7.76E+02		2.58E+02
f_{11}	Mean	1.90E+02	+	1.97E+02	+	3.51E+02	+	6.42E+01	+	9.61E+01	+	1.14E+03	+	**2.71E+01**
	Std	9.13E+01		1.26E+02		2.68E+02		3.79E+01		2.51E+01		1.66E+03		**2.19E+01**
f_{12}	Mean	4.65E+05	+	5.92E+05	+	3.69E+05	+	9.61E+05	+	6.25E+05	+	2.12E+08	+	**2.77E+04**
	Std	2.11E+05		4.21E+05		2.74E+05		5.72E+05		3.05E+05		7.62E+08		**1.54E+04**
f_{13}	Mean	7.83E+03	-	1.73E+04	+	**3.12E+03**	-	1.08E+04	-	1.85E+04	+	1.20E+08	+	1.11E+04
	Std	4.90E+03		1.76E+04		**2.40E+03**		7.88E+03		1.47E+04		6.56E+08		1.33E+04
f_{14}	Mean	6.70E+04	+	4.46E+04	+	6.90E+04	+	4.26E+05	+	2.18E+05	+	1.74E+02	+	**3.93E+01**
	Std	5.27E+04		6.03E+04		4.41E+04		3.80E+05		2.21E+05		5.32E+02		**2.55E+01**
f_{15}	Mean	1.36E+03	-	7.24E+03	+	**7.63E+02**	-	1.02E+03	-	2.80E+03	+	3.07E+07	+	1.59E+03
	Std	1.06E+03		7.53E+03		**6.99E+02**		1.30E+03		1.80E+03		1.17E+08		3.22E+03
f_{16}	Mean	6.72E+02	+	5.31E+02	+	6.50E+02	+	6.18E+02	+	6.44E+02	+	8.62E+02	+	**4.46E+02**
	Std	1.30E+02		1.74E+02		1.47E+02		1.23E+02		2.33E+02		4.30E+02		**2.00E+02**
f_{17}	Mean	1.98E+02	+	1.34E+02	+	2.47E+02	+	**1.17E+02**	-	1.86E+02	+	3.20E+02	+	1.33E+02
	Std	8.71E+01		8.02E+01		9.15E+01		**6.12E+01**		1.51E+02		3.00E+02		1.18E+02
f_{18}	Mean	1.77E+05	+	2.26E+05	+	1.68E+05	+	2.22E+05	+	2.88E+05	+	2.55E+06	+	**9.43E+03**
	Std	9.35E+04		1.44E+05		9.02E+04		1.99E+05		1.99E+05		5.49E+06		**7.41E+03**
f_{19}	Mean	1.37E+03	+	1.38E+04	+	9.30E+02	+	3.21E+03	+	3.55E+03	+	7.33E+07	+	**4.48E+02**
	Std	1.29E+03		1.64E+04		1.13E+03		2.56E+03		2.73E+03		1.74E+08		**1.81E+03**
f_{20}	Mean	2.39E+02	+	1.59E+02	-	2.81E+02	+	**1.56E+02**	-	2.82E+02	+	5.78E+02	+	1.85E+02
	Std	8.22E+01		9.80E+01		1.03E+02		**4.55E+01**		1.38E+02		2.06E+02		1.23E+02
f_{21}	Mean	2.58E+02	+	**2.39E+02**	-	2.52E+02	+	2.49E+02	+	2.50E+02	+	3.27E+02	+	2.42E+02
	Std	6.70E+01		**2.32E+01**		7.29E+01		9.45E+00		9.87E+00		4.11E+01		8.45E+00
f_{22}	Mean	4.68E+02	-	1.00E+02	-	3.89E+02	-	**1.00E+02**	-	1.00E+02	-	1.58E+03	+	8.48E+02
	Std	9.43E+02		7.25E-07		8.61E+02		**0.00E+00**		7.03E-03		8.58E+02		1.18E+03
f_{23}	Mean	4.24E+02	+	3.97E+02	+	4.11E+02	+	**3.87E+02**	-	4.08E+02	+	5.10E+02	+	3.94E+02
	Std	2.43E+01		8.64E+00		2.82E+01		**5.50E+01**		1.13E+01		3.65E+01		8.04E+00
f_{24}	Mean	**4.37E+02**	-	5.18E+02	+	4.74E+02	+	4.99E+02	+	5.32E+02	+	8.32E+02	+	4.72E+02
	Std	**2.04E+02**		2.25E+01		2.02E+02		7.65E+01		2.08E+01		3.32E+02		9.01E+00
f_{25}	Mean	3.85E+02	-	**3.78E+02**	-	3.84E+02	-	3.94E+02	+	3.86E+02	-	6.17E+02	+	3.87E+02
	Std	1.13E+00		**1.06E+00**		1.21E+00		1.44E+01		2.01E+00		1.94E+02		1.45E+00
f_{26}	Mean	**3.41E+02**	-	1.36E+03	-	3.59E+02	-	5.57E+02	-	1.29E+03	-	2.69E+03	+	1.49E+03
	Std	**3.73E+02**		2.71E+02		4.04E+02		6.53E+02		5.73E+02		1.74E+03		1.40E+02
f_{27}	Mean	5.12E+02	+	**4.97E+02**	-	5.12E+02	+	5.10E+02	-	5.03E+02	-	5.40E+02	+	5.11E+02
	Std	5.36E+00		**1.06E+01**		8.29E+00		5.23E+00		4.51E+00		2.10E+01		7.45E+00
f_{28}	Mean	4.04E+02	+	4.95E+02	+	3.97E+02	+	4.06E+02	+	4.00E+02	+	1.42E+03	+	**3.65E+02**
	Std	3.77E+00		1.63E+01		1.28E+01		4.59E+00		5.00E+00		8.94E+02		**5.86E+01**
f_{29}	Mean	5.91E+02	+	**4.27E+02**	-	5.87E+02	+	5.11E+02	+	5.40E+02	+	8.42E+02	+	4.95E+02
	Std	8.52E+01		**8.21E+01**		9.03E+01		6.60E+01		6.55E+01		3.24E+02		7.33E+01
f_{30}	Mean	6.55E+03	+	**3.13E+03**	-	5.52E+03	+	5.37E+03	+	4.87E+03	+	1.23E+05	+	4.50E+03
	Std	2.32E+03		**3.14E+03**		1.65E+03		1.21E+03		1.61E+03		4.58E+05		2.64E+03
Total	+/=/-	20/0/9		17/0/12		20/0/9		19/0/10		22/0/7		29/0/0		

Table 5.8: Comparison between FOABC and other ABC variants with $D = 50$

Functions		ABC		NSABC		$iff-$ABC		MGABC		ILTD_ABC		sdABC		FOABC$_{r=12,q=0.8}$
f_1	Mean	1.85E+03	-	5.87E+03	-	1.11E+03	-	**5.30E+02**	-	6.79E+03	+	3.13E+10	+	6.42E+03
	Std	1.91E+03		7.17E+03		1.23E+03		**7.17E+02**		1.10E+04		1.98E+10		6.87E+03
f_3	Mean	2.07E+05	+	2.67E+05	+	2.61E+05	+	1.16E+05	+	1.18E+05	+	1.47E+05	+	**1.15E+05**
	Std	3.23E+04		2.97E+04		2.23E+04		1.28E+04		9.03E+03		2.86E+04		**1.74E+04**
f_4	Mean	4.06E+01	-	5.67E+01	-	**3.33E+01**	-	9.12E+01	-	4.54E+01	-	5.84E+03	+	1.19E+02
	Std	1.33E+01		2.48E+01		**9.37E+00**		4.83E+01		2.16E+01		5.65E+03		6.27E+01
f_5	Mean	1.98E+02	+	1.19E+02	+	2.01E+02	+	1.59E+02	+	2.19E+02	+	4.08E+02	+	**1.03E+02**
	Std	1.69E+01		1.68E+01		1.98E+01		2.29E+01		4.84E+01		9.31E+01		**1.95E+01**
f_6	Mean	5.20E-10	-	9.40E-13	-	1.44E-09	-	**1.10E-13**	-	2.90E-03	-	3.75E+01	+	5.50E-02
	Std	3.95E-10		1.81E-13		1.25E-09		**2.08E-14**		1.13E-03		8.72E+00		6.02E-02
f_7	Mean	2.13E+02	+	**1.52E+02**	-	2.09E+02	+	2.01E+02	+	3.40E+02	+	7.59E+02	+	1.78E+02
	Std	1.90E+01		**1.68E+01**		1.79E+01		2.35E+01		1.14E+02		1.14E+02		2.01E+01
f_8	Mean	1.96E+02	+	1.16E+02	+	2.00E+02	+	1.55E+02	+	1.86E+02	+	4.40E+02	+	**1.06E+02**
	Std	2.17E+01		1.44E+01		2.66E+01		1.65E+01		3.13E+01		8.54E+01		**1.59E+01**
f_9	Mean	4.52E+03	+	5.38E+02	+	4.40E+03	+	6.33E+02	+	3.09E+03	+	2.51E+04	+	**2.96E+02**
	Std	1.49E+03		2.07E+02		1.18E+03		5.64E+02		2.53E+03		1.06E+04		**2.68E+02**
f_{10}	Mean	4.29E+03	-	4.23E+03	-	4.16E+03	-	**3.98E+03**	-	4.23E+03	-	1.08E+04	+	4.50E+03
	Std	3.46E+02		5.60E+02		3.74E+02		**3.87E+02**		8.69E+02		2.69E+03		2.69E+02
f_{11}	Mean	7.17E+02	+	4.40E+03	+	9.19E+02	+	8.24E+02	+	1.67E+03	+	3.75E+03	+	**7.12E+01**
	Std	5.26E+02		2.87E+03		7.47E+02		4.70E+02		5.91E+02		3.53E+03		**2.33E+01**
f_{12}	Mean	3.40E+06	+	6.73E+06	+	2.70E+06	+	2.26E+06	+	1.69E+06	+	2.86E+09	+	**4.26E+05**
	Std	1.30E+06		3.05E+06		1.13E+06		8.71E+05		5.34E+05		3.04E+09		**2.87E+05**
f_{13}	Mean	4.46E+03	-	6.11E+03	-	2.01E+03	-	**1.50E+03**	-	2.76E+03	-	1.21E+08	+	8.38E+03
	Std	2.17E+03		8.22E+03		1.69E+03		**1.69E+03**		3.82E+03		1.89E+08		9.85E+03
f_{14}	Mean	6.22E+05	+	1.29E+06	+	7.31E+05	+	2.44E+06	+	9.57E+05	+	8.02E+04	+	**4.45E+03**
	Std	5.16E+05		7.46E+05		5.04E+05		1.21E+06		5.85E+05		3.94E+05		**3.95E+03**
f_{15}	Mean	6.64E+03	+	2.18E+04	+	8.08E+03	+	1.42E+04	+	1.45E+04	+	5.89E+08	+	**5.38E+03**
	Std	3.86E+03		4.68E+04		5.70E+03		4.95E+03		6.74E+03		1.56E+09		**5.54E+03**
f_{16}	Mean	1.25E+03	+	1.29E+03	+	1.25E+03	+	9.59E+02	-	**9.20E+02**	-	1.89E+03	+	1.02E+03
	Std	2.33E+02		2.19E+02		1.75E+02		2.41E+02		**2.53E+02**		4.64E+02		2.46E+02
f_{17}	Mean	8.71E+02	+	9.21E+02	+	8.92E+02	+	8.72E+02	+	8.82E+02	+	2.15E+03	+	**6.05E+02**
	Std	2.02E+02		2.12E+02		1.61E+02		1.97E+02		2.07E+02		3.34E+03		**1.89E+02**
f_{18}	Mean	1.03E+06	+	2.17E+06	+	8.36E+05	+	2.40E+06	+	1.27E+06	+	2.92E+06	+	**9.10E+04**
	Std	5.53E+05		1.69E+06		5.06E+05		9.96E+05		5.62E+05		1.06E+07		**6.00E+04**
f_{19}	Mean	9.14E+03	-	1.59E+04	+	**6.38E+03**	-	1.70E+04	+	1.98E+04	+	1.17E+05	+	1.52E+04
	Std	4.92E+03		1.16E+04		**3.44E+03**		5.00E+03		1.15E+04		4.52E+05		1.27E+04
f_{20}	Mean	7.45E+02	+	7.26E+02	+	6.94E+02	+	**4.67E+02**	-	5.61E+02	-	1.27E+03	+	5.65E+02
	Std	1.44E+02		2.02E+02		1.54E+02		**1.74E+02**		2.40E+02		4.55E+02		2.45E+02
f_{21}	Mean	4.04E+02	+	3.29E+02	+	4.09E+02	+	3.28E+02	+	3.41E+02	+	5.37E+02	+	**3.09E+02**
	Std	4.87E+01		2.14E+01		2.48E+01		1.59E+01		2.37E+01		4.97E+01		**1.47E+01**
f_{22}	Mean	4.76E+03	-	4.74E+03	-	4.67E+03	-	**3.89E+03**	-	4.47E+03	-	8.33E+03	+	5.17E+03
	Std	1.02E+03		4.54E+02		1.30E+03		**1.96E+03**		2.27E+03		4.88E+03		4.45E+02
f_{23}	Mean	6.48E+02	+	5.55E+02	+	6.55E+02	+	5.53E+02	+	5.69E+02	+	8.52E+02	+	**5.43E+02**
	Std	3.77E+01		3.13E+01		2.89E+01		1.96E+01		1.86E+01		6.84E+01		**1.83E+01**
f_{24}	Mean	9.60E+02	+	8.87E+02	+	1.03E+03	+	7.61E+02	+	8.14E+02	+	1.06E+03	+	**6.16E+02**
	Std	1.49E+02		5.22E+01		6.92E+01		3.71E+01		3.85E+01		1.17E+02		**1.98E+01**
f_{25}	Mean	5.12E+02	-	**4.39E+02**	-	4.94E+02	-	5.91E+02	+	5.59E+02	+	3.58E+03	+	5.45E+02
	Std	1.73E+01		**1.51E+01**		2.32E+01		1.29E+01		2.27E+01		2.01E+03		3.44E+01
f_{26}	Mean	1.87E+03	-	2.55E+03	+	1.31E+03	-	**8.62E+02**	-	1.80E+03	-	6.88E+03	+	2.34E+03
	Std	1.56E+03		4.93E+02		1.46E+03		**1.47E+03**		1.25E+03		2.64E+03		1.88E+02
f_{27}	Mean	6.43E+02	-	**5.00E+02**	-	6.52E+02	-	6.26E+02	-	5.98E+02	-	8.83E+02	+	6.68E+02
	Std	4.10E+01		**2.61E-04**		3.62E+01		2.88E+01		2.45E+01		1.34E+02		6.80E+01
f_{28}	Mean	4.83E+02	-	4.83E+02	-	**4.74E+02**	-	5.32E+02	+	5.00E+02	-	2.60E+03	+	5.06E+02
	Std	1.26E+01		2.34E+01		**1.26E+01**		1.97E+01		1.94E+01		1.39E+03		2.34E+01
f_{29}	Mean	1.01E+03	+	8.11E+02	+	1.08E+03	+	8.58E+02	+	6.89E+02	+	1.85E+03	+	**6.62E+02**
	Std	1.48E+02		1.33E+02		2.26E+02		1.53E+02		1.91E+02		1.59E+03		**1.64E+02**
f_{30}	Mean	7.24E+05	-	**7.02E+03**	-	7.13E+05	-	8.45E+05	-	8.81E+05	+	9.89E+06	+	8.47E+05
	Std	5.09E+04		**8.54E+03**		5.18E+04		6.47E+04		1.14E+05		9.46E+06		2.54E+05
Total	+/=/-	17/0/12		18/0/11		17/0/12		18/0/11		19/0/10		29/0/0		

best results. MGABC outperforms the others on 2 of these problems. Moreover, FOABC has become more competitive in the composition functions, while the performance of ABC and NSABC has declined when $D = 50$.

Considering the Wilcoxon test results, compared to the basic ABC and *iff*-ABC algorithms, FOABC surpasses both of them on 17. And the number of problems that FOABC exceeds the NSABC and MGABC are both 18. ILTD_ABC fails to outperform FOABC on 19 out of 29 functions. Moreover, sdABC is not comparable to the proposed algorithm in this situation.

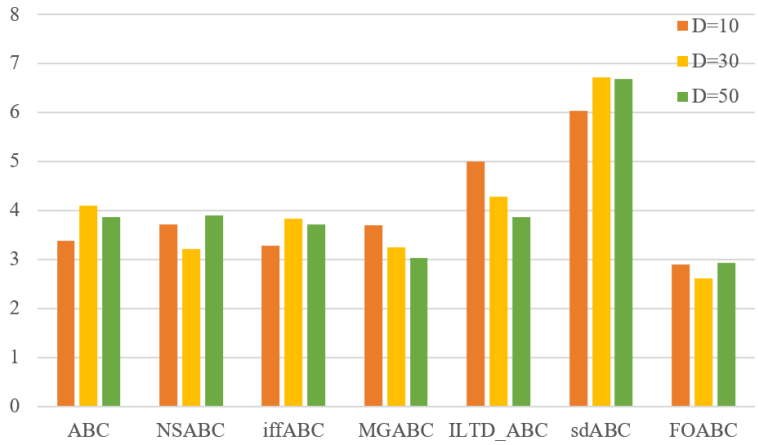


Fig. 5.4. Average rankings of ABC algorithms by Friedman test with $D = 10, 30,$ and 50

Furthermore, the Friedman tests are employed to fully evaluate the performance of involved algorithms. The average rankings for $D = 10$, 30, and 50 are presented in Figure 5.4. It can be observed that FOABC always owns the best average ranking in all three cases. For 10-dimensional problems, *iff*-ABC is the second-best followed by the basic ABC algorithm. When $D = 30$, NSABC and MGABC have better performance than the other methods. And when $D = 50$, MGABC becomes the second-best followed by *iff*-ABC. Therefore, FOABC has outstanding and stable performance throughout the three cases.

5.3.3 Comparison with non-ABC algorithms

It is crucial to further compare FOABC with other FO-based algorithms and improved versions of other meta-heuristics in order to completely evaluate its efficacy. As presented in Table 5.9, five effective enhanced algorithms are involved

in the following comparisons. It is worth pointing out that the FODPSO, FOCS, and FOFA algorithms were chosen because they also incorporate FOC in their basic algorithms. In this case, by comparing FOABC with them, it is possible to check whether the proposed manner of incorporating FOC into ABC is more effective. And the algorithms LIPS and CLPSO are two excellent algorithms that have been widely compared in various experiments. The tests are also conducted on the CEC 2017 benchmarks with $D = 10$, 30, and 50.

Table 5.9: Parameter settings of FOABC and compared non-ABC algorithms

Algorithm	Parameter setting
CLPSO (Liang *et al.*, 2006)	$N = 40$, $c_1 = 1.49445$, $c_2 = 1.49445$
FODPSO (Couceiro *et al.*, 2012)	$N = 20$, $nswarms = 5$, $\alpha = 0.6$
LIPS (Qu *et al.*, 2013)	$N = 50$, $weight = 0.729843788$
FOCS (Yousri & Mirjalili, 2020)	$N = 30$, $pa = 0.25$, $r = 4$, $\alpha = 0.4$
FOFA (Mousavi & Alfi, 2018)	$N = 30$, $\gamma = 0.9$, $\beta_0 = 1.8$, $\alpha = 0.25$, $r = 4$, $\nu = 0.7$
FOABC	$SN = 50$, $limit = SN \cdot D$, $CR = 0.8$, $r = 8$, $q = 0.8$

The comparison when $D = 10$ is presented in Table 5.10 where the superiority of FOABC is obvious. The proposed algorithm achieves the best performance on 17 out of 29 functions. The errors obtained by FOABC on uni-modal functions are much smaller than that of other involved algorithms. As for the multi-modal problems, FOABC performs the best on 4 problems among $f_4 - f_{10}$. Compared to the CLPSO, there are 2 multi-modal functions that FOABC fails to outperform CLPSO while FOCS has the same comparing results. LIPS has a clear advantage on f_9 compared to the others. And FOABC performs better than FOFA on all the multi-modal problems.

Moreover, the advantage of FOABC can also be found considering the following hybrid functions $f_{11} - f_{20}$. Except that CLPSO outperforms FOABC on f_{16} and f_{20}, all the competitors fail to surpass the proposed algorithm. In solving the rest composition problems, all the involved algorithms obtain relatively similar results. More precisely, algorithms CLPSO, FODPSO, and FOABC all achieve the best on 3 problems. Meanwhile, LIPS outperforms the others on 1 problem. Considering the Wilcoxon test results, there are 18 of 29 functions that FOABC performs better than CLPSO. And it surpasses the LIPS algorithm on 23 out of 29 problems. Moreover, although FOFA and FOABC get close results, FOABC

Table 5.10: Comparison between FOABC and other improved meta-heuristic algorithms with $D = 10$

Functions		CLPSO		LIPS		FODPSO		FOFA		FOCS		FOABC$_{r=12,q=0.8}$
f_1	Mean	1.52E+02	+	4.90E+02	+	3.43E+01	+	2.95E+10	+	5.44E+01	+	**4.33E-09**
	Std	1.16E+02		7.65E+02		5.59E+01		9.68E+07		1.12E+02		**1.50E-08**
f_3	Mean	2.97E+00	+	6.10E-07	+	4.67E-10	+	3.13E+05	+	2.86E-02	+	**1.71E-14**
	Std	3.02E+00		1.99E-06		7.45E-10		1.37E+05		3.62E-02		**3.04E-14**
f_4	Mean	2.58E+00	+	3.18E+00	+	6.02E-01	-	5.38E+03	+	**4.15E-01**	-	8.01E-01
	Std	1.33E+00		1.55E+00		8.01E-01		3.30E+01		**2.03E-01**		4.66E-01
f_5	Mean	5.29E+00	+	6.60E+00	+	3.11E+01	+	2.15E+02	+	1.35E+01	+	**4.68E+00**
	Std	1.41E+00		2.59E+00		7.73E+00		2.88E+00		3.46E+00		**1.50E+00**
f_6	Mean	**1.69E-10**	-	8.14E-02	+	6.59E+00	+	1.21E+02	+	3.53E-02	+	3.78E-07
	Std	**1.22E-10**		2.21E-01		4.56E+00		4.28E+00		2.18E-02		2.07E-06
f_7	Mean	1.82E+01	+	1.42E+01	+	2.27E+01	+	1.95E+02	+	2.87E+01	+	**1.40E+01**
	Std	2.10E+00		1.21E+00		3.97E+00		7.51E+00		5.16E+00		**1.51E+00**
f_8	Mean	5.60E+00	+	6.53E+00	+	1.17E+01	+	1.35E+02	+	1.32E+01	+	**5.21E+00**
	Std	1.12E+00		2.69E+00		3.98E+00		2.43E+00		3.60E+00		**1.88E+00**
f_9	Mean	1.38E-08	-	**6.33E-13**	-	6.35E+00	+	1.78E+03	+	1.09E-02	-	1.51E-02
	Std	1.72E-08		**4.14E-13**		1.96E+01		2.41E+02		2.87E-02		8.29E-02
f_{10}	Mean	2.30E+02	+	4.66E+02	+	6.14E+02	+	4.94E+03	+	4.51E+02	+	**1.87E+02**
	Std	9.12E+01		1.39E+02		1.81E+02		4.64E+01		1.45E+02		**9.31E+01**
f_{11}	Mean	2.46E+00	+	1.56E+01	+	1.36E+01	+	5.89E+07	+	4.78E+00	+	**1.13E+00**
	Std	9.22E-01		1.13E+01		5.37E+00		1.73E+06		1.23E+00		**7.72E-01**
f_{12}	Mean	1.84E+04	+	4.08E+04	+	1.59E+03	+	5.57E+09	+	8.93E+02	+	**6.74E+01**
	Std	1.11E+04		1.30E+05		7.31E+02		4.68E+07		2.77E+02		**1.33E+02**
f_{13}	Mean	8.40E+01	+	9.16E+02	+	6.73E+02	+	2.74E+09	+	1.25E+01	+	**4.78E+00**
	Std	1.11E+02		1.77E+03		5.32E+02		2.32E+07		3.10E+00		**1.99E+00**
f_{14}	Mean	4.56E+01	+	4.27E+02	+	7.83E+01	+	2.12E+09	+	1.18E+01	+	**1.65E+00**
	Std	4.04E+01		7.56E+02		4.59E+01		2.31E+07		4.65E+00		**1.13E+00**
f_{15}	Mean	2.31E+01	+	4.30E+02	+	1.00E+02	+	6.73E+08	+	2.92E+00	+	**5.95E-01**
	Std	2.15E+01		7.95E+02		8.28E+01		2.65E+07		9.28E-01		**8.15E-01**
f_{16}	Mean	**1.71E+00**	-	6.73E+01	+	1.22E+02	+	1.77E+03	+	5.83E+00	+	3.12E+00
	Std	**5.85E-01**		8.66E+01		8.41E+01		2.09E+01		7.94E+00		4.70E+00
f_{17}	Mean	4.82E+00	+	3.68E+01	+	2.97E+01	+	1.46E+03	+	2.75E+01	+	**2.72E+00**
	Std	3.06E+00		1.42E+01		8.34E+00		1.91E+01		2.89E+00		**5.13E+00**
f_{18}	Mean	5.14E+02	+	1.99E+03	+	1.80E+03	+	1.41E+10	+	1.30E+01	+	**3.74E-01**
	Std	4.52E+02		1.75E+03		1.82E+03		8.86E+07		4.97E+00		**4.50E-01**
f_{19}	Mean	1.02E+01	+	1.32E+03	+	1.97E+02	+	1.19E+10	+	2.39E+00	+	**5.49E-02**
	Std	9.33E+00		2.22E+03		2.66E+02		9.92E+07		5.50E-01		**1.82E-01**
f_{20}	Mean	**5.51E-02**	-	3.10E+01	+	4.28E+01	+	1.04E+03	+	1.14E+01	+	4.44E-01
	Std	**1.41E-01**		1.71E+01		1.75E+01		2.79E+01		5.97E+00		5.51E-01
f_{21}	Mean	1.07E+02	-	1.89E+02	+	**1.00E+02**	-	7.12E+02	+	1.67E+02	+	1.58E+02
	Std	4.84E+00		4.08E+01		**1.26E+00**		5.29E+00		5.71E+01		5.46E+01
f_{22}	Mean	**7.71E+01**	-	9.55E+01	-	9.24E+01	-	2.98E+03	+	8.56E+01	-	9.61E+01
	Std	**2.31E+01**		1.87E+01		1.95E+01		3.75E+01		3.17E+01		1.76E+01
f_{23}	Mean	3.06E+02	-	**3.03E+02**	-	3.51E+02	+	1.94E+03	+	3.13E+02	+	3.09E+02
	Std	5.09E+00		**3.89E+01**		1.24E+01		2.61E+01		3.51E+00		3.17E+00
f_{24}	Mean	1.50E+02	-	2.54E+02	-	**1.34E+02**	-	9.83E+02	+	3.29E+02	-	3.39E+02
	Std	6.09E+01		1.05E+02		**1.00E+02**		2.73E+00		6.11E+01		2.54E+00
f_{25}	Mean	**3.73E+02**	-	3.91E+02	-	3.88E+02	-	2.28E+03	+	4.04E+02	-	4.12E+02
	Std	**5.94E+01**		8.45E+01		5.44E+01		1.48E+01		1.59E+01		2.19E+01
f_{26}	Mean	1.62E+02	-	2.65E+02	-	**1.41E+02**	-	3.09E+03	+	3.00E+02	-	3.09E+02
	Std	1.08E+02		1.96E+02		**1.09E+02**		1.18E+01		2.81E-06		2.56E+01
f_{27}	Mean	3.92E+02	+	4.04E+02	+	4.17E+02	+	2.26E+03	+	3.90E+02	+	**3.90E+02**
	Std	1.70E+00		5.89E+00		1.67E+01		2.12E+01		2.75E+00		**1.34E+00**
f_{28}	Mean	**2.68E+02**	-	4.35E+02	+	3.16E+02	-	1.69E+03	+	4.04E+02	+	3.45E+02
	Std	**1.02E+02**		1.42E+02		4.33E+01		7.22E+00		1.53E+02		1.01E+02
f_{29}	Mean	2.59E+02	+	2.88E+02	+	2.91E+02	+	4.06E+04	+	2.67E+02	+	**2.35E+02**
	Std	8.73E+00		2.40E+01		1.45E+01		1.24E+03		2.75E+01		**3.56E+00**
f_{30}	Mean	9.17E+03	+	1.44E+05	+	2.66E+03	+	4.88E+08	+	1.26E+05	+	**7.41E+02**
	Std	7.47E+03		6.07E+05		8.45E+02		5.08E+06		2.81E+05		**2.79E+02**
Total	+/=/-	18/0/11		23/0/6		22/0/7		29/0/0		23/0/6		

still outperforms it in terms of average values. Meanwhile, FODPSO achieves smaller errors than FOABC on 7 problems and fails on the rest part. In comparison with algorithms FOFA and FOCS, the proposed algorithm obtains better solutions on 29 and 23 problems, respectively.

The comparison results of $D = 30$ and 50 are shown in Table 5.11 - Table 5.12, respectively. In Table 5.11, the advantages of algorithms CLPSO and FOABC are evident. CLPSO obtains the smallest errors on 9 functions while FOABC has 17 out of 29. Moreover, the advantage of FOABC is significant in solving multi-modal and composition problems. According to the Wilcoxon test results, there are 18 functions that FOABC exceeds CLPSO. In addition, FOABC outperforms both LIPS and FOCS on 24 problems, while it surpasses FODPSO on 22 out of 29 functions. And FOFA fails to exceed FOABC on the concerned benchmarks.

As shown in Table 5.12, there are 12 functions where FOABC obtains the best results while CLPSO achieves the best on 10 functions. FOCS algorithm manages to obtain the best on 5 problems. It can be found that the proposed algorithm is able to find competitive solutions on multi-modal, hybrid and composition functions. Considering the results of Wilcoxon tests, the CLPSO obtains better results than FOABC does on 15 out of 29. LIPS fails to overcome FOABC on 23 functions. Compared with other FO-based algorithms, the proposed algorithm still has a distinct superiority at $D = 50$. More precisely, FOABC outperforms FODPSO on 25 functions while it surpasses FOCS on 23 problems. Additionally, FOABC outperforms FOFA in terms of average values across the board.

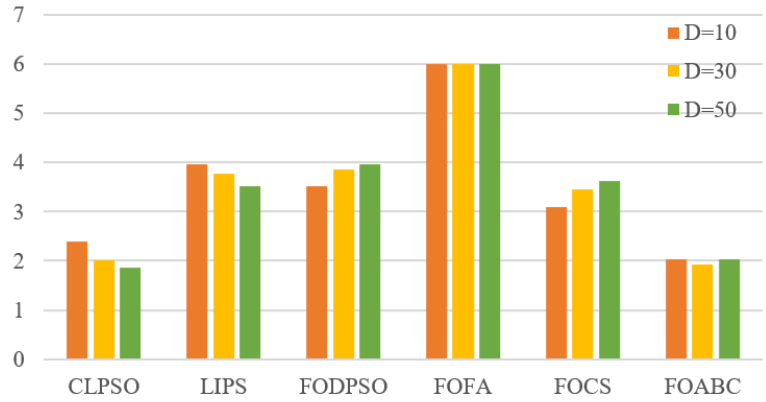


Fig. 5.5. Average rankings of non-ABC algorithms and FOABC by Friedman test with $D = 10, 30$, and 50

Table 5.11: Comparison between FOABC and other improved meta-heuristic algorithms with $D = 30$

Functions		CLPSO		LIPS		FODPSO		FOFA		FOCS		FOABC$_{r=12,q=0.8}$
f_1	Mean	**1.49E+01**	-	3.78E+02	-	4.87E+06	+	8.43E+10	+	1.00E+10	+	1.50E+03
	Std	**2.89E+01**		9.29E+02		6.56E+06		1.74E+08		0.00E+00		2.35E+03
f_3	Mean	1.92E+04	+	2.06E+04	+	**8.07E+03**	-	4.15E+08	+	3.11E+04	+	1.10E+04
	Std	5.35E+03		8.02E+03		**2.09E+03**		1.46E+08		9.26E+03		5.59E+03
f_4	Mean	**5.56E+01**	-	1.37E+02	+	9.95E+01	+	3.46E+04	+	7.40E+01	-	8.71E+01
	Std	**2.28E+01**		6.22E+01		2.42E+01		1.29E+02		1.43E+01		1.76E+01
f_5	Mean	4.21E+01	+	5.50E+01	+	1.60E+02	+	6.16E+02	+	1.10E+02	+	**3.73E+01**
	Std	7.89E+00		1.17E+01		2.10E+01		3.15E+00		1.74E+01		**9.07E+00**
f_6	Mean	**2.27E-13**	-	7.57E+00	+	4.41E+01	+	1.41E+02	+	1.49E-01	+	3.55E-03
	Std	**4.22E-14**		3.58E+00		3.59E+00		1.71E+00		1.41E-01		1.18E-02
f_7	Mean	8.13E+01	+	9.33E+01	+	1.99E+02	+	9.18E+02	+	1.68E+02	+	**7.84E+01**
	Std	7.91E+00		1.80E+01		3.69E+01		1.02E+01		1.91E+01		**9.53E+00**
f_8	Mean	5.18E+01	+	6.59E+01	+	1.08E+02	+	5.11E+02	+	1.05E+02	+	**4.15E+01**
	Std	7.18E+00		1.71E+01		1.60E+01		2.29E+00		1.55E+01		**1.01E+01**
f_9	Mean	3.99E+01	+	3.79E+02	+	2.43E+03	+	2.72E+04	+	1.86E+02	+	**6.11E+00**
	Std	2.06E+01		2.84E+02		3.89E+02		1.47E+03		1.57E+02		**7.86E+00**
f_{10}	Mean	**2.21E+03**	-	2.70E+03	+	3.17E+03	+	1.01E+04	+	4.26E+03	+	2.30E+03
	Std	**2.55E+02**		4.28E+02		3.96E+02		7.53E+01		6.91E+02		2.58E+02
f_{11}	Mean	6.01E+01	+	1.85E+02	+	8.78E+01	+	5.76E+08	+	1.22E+02	+	**2.71E+01**
	Std	2.08E+01		1.27E+02		1.77E+01		1.18E+07		2.96E+01		**2.19E+01**
f_{12}	Mean	3.69E+05	+	1.46E+06	+	4.81E+04	+	2.93E+10	+	3.00E+09	+	**2.77E+04**
	Std	1.78E+05		4.56E+06		2.66E+04		5.81E+07		4.66E+09		**1.54E+04**
f_{13}	Mean	**3.16E+02**	-	2.77E+03	-	7.15E+03	-	4.38E+10	+	2.08E+03	-	1.11E+04
	Std	**2.99E+02**		2.40E+03		2.31E+03		8.06E+07		9.61E+02		1.33E+04
f_{14}	Mean	3.61E+04	+	1.02E+04	+	1.90E+03	+	1.21E+09	+	8.02E+01	+	**3.93E+01**
	Std	2.95E+04		8.23E+03		2.16E+03		1.21E+07		1.05E+01		**2.55E+01**
f_{15}	Mean	**9.76E+01**	-	1.52E+03	-	9.48E+02	-	6.34E+09	+	1.38E+02	-	1.59E+03
	Std	**4.87E+01**		1.90E+03		4.68E+02		5.64E+07		3.36E+01		3.22E+03
f_{16}	Mean	5.40E+02	+	6.85E+02	+	8.62E+02	+	2.54E+04	+	8.83E+02	+	**4.46E+02**
	Std	1.56E+02		1.78E+02		1.79E+02		1.27E+02		1.97E+02		**2.00E+02**
f_{17}	Mean	1.56E+02	+	2.55E+02	+	3.51E+02	+	2.72E+05	+	2.41E+02	+	**1.33E+02**
	Std	6.94E+01		9.06E+01		1.46E+02		3.72E+03		6.79E+01		**1.18E+02**
f_{18}	Mean	1.39E+05	+	1.67E+05	+	3.92E+04	+	4.63E+09	+	1.17E+04	+	**9.43E+03**
	Std	7.49E+04		1.02E+05		1.71E+04		3.86E+07		6.55E+03		**7.41E+03**
f_{19}	Mean	5.09E+01	-	1.11E+03	+	2.91E+02	-	6.51E+09	+	**4.68E+01**	-	4.48E+02
	Std	3.69E+01		1.41E+03		2.23E+02		4.46E+07		**1.17E+01**		1.81E+03
f_{20}	Mean	2.00E+02	+	3.01E+02	+	3.65E+02	+	3.41E+03	+	2.28E+02	+	**1.85E+02**
	Std	5.97E+01		9.96E+01		8.26E+01		2.23E+01		1.03E+02		**1.23E+02**
f_{21}	Mean	**2.28E+02**	-	2.63E+02	+	3.54E+02	+	1.12E+03	+	3.12E+02	+	2.42E+02
	Std	**4.64E+01**		1.55E+01		2.47E+01		2.75E+00		2.06E+01		8.45E+00
f_{22}	Mean	2.20E+02	-	**1.00E+02**	-	1.35E+02	-	1.09E+04	+	3.04E+03	+	8.48E+02
	Std	4.82E+02		**1.01E+00**		1.79E+01		5.57E+01		2.20E+03		1.18E+03
f_{23}	Mean	3.98E+02	+	4.41E+02	+	7.32E+02	+	5.64E+03	+	4.76E+02	+	**3.94E+02**
	Std	1.07E+01		2.12E+01		8.18E+01		3.47E+01		1.68E+01		**8.04E+00**
f_{24}	Mean	4.73E+02	+	4.96E+02	+	6.82E+02	+	2.79E+03	+	5.61E+02	+	**4.72E+02**
	Std	9.81E+01		3.19E+01		1.40E+02		2.34E+00		3.35E+01		**9.01E+00**
f_{25}	Mean	**3.87E+02**	-	4.26E+02	+	4.07E+02	+	6.65E+03	+	3.87E+02	-	3.87E+02
	Std	**1.04E+00**		2.52E+01		1.01E+01		3.42E+01		9.97E-01		1.45E+00
f_{26}	Mean	**7.67E+02**	-	1.28E+03	-	1.00E+03	-	1.35E+04	+	2.41E+03	+	1.49E+03
	Std	**5.18E+02**		8.61E+02		1.09E+03		3.29E+01		4.96E+02		1.40E+02
f_{27}	Mean	5.12E+02	+	6.07E+02	+	6.00E+02	+	7.79E+03	+	5.20E+02	+	**5.11E+02**
	Std	4.60E+00		2.10E+01		2.82E+01		4.39E+01		1.81E+01		**7.45E+00**
f_{28}	Mean	4.22E+02	+	4.73E+02	+	4.54E+02	+	7.39E+03	+	3.76E+02	+	**3.65E+02**
	Std	8.60E+00		8.76E+01		2.24E+01		2.02E+01		5.46E+01		**5.86E+01**
f_{29}	Mean	5.52E+02	+	9.44E+02	+	1.01E+03	+	2.23E+05	+	9.01E+02	+	**4.95E+02**
	Std	6.01E+01		1.20E+02		1.34E+02		5.13E+03		1.46E+02		**7.33E+01**
f_{30}	Mean	6.19E+03	+	8.33E+04	+	8.93E+03	+	1.02E+10	+	1.63E+04	+	**4.50E+03**
	Std	1.84E+03		1.27E+05		4.31E+03		3.03E+07		6.97E+03		**2.64E+03**
Total	+/=/-	18/0/11		24/0/5		23/0/6		29/0/0		24/0/5		

Table 5.12: Comparison between FOABC and other improved meta-heuristic algorithms with $D = 50$

Functions		CLPSO		LIPS		FODPSO		FOFA		FOCS		FOABC$_{r=12,q=0.8}$
f_1	Mean	**2.89E+01**	-	4.86E+02	-	1.01E+09	+	1.35E+11	+	1.00E+10	+	6.42E+03
	Std	**4.32E+01**		1.03E+03		7.67E+08		1.36E+08		0.00E+00		6.87E+03
f_3	Mean	7.05E+04	-	1.04E+05	-	**4.60E+04**	-	1.77E+14	+	1.06E+05	-	1.15E+05
	Std	1.26E+04		2.08E+04		**7.72E+03**		3.78E+12		2.63E+04		1.74E+04
f_4	Mean	8.21E+01	-	6.13E+02	+	4.55E+02	+	5.66E+04	+	**7.92E+01**	-	1.19E+02
	Std	2.61E+01		2.39E+02		8.02E+01		9.49E+01		**4.13E+01**		6.27E+01
f_5	Mean	1.14E+02	+	1.68E+02	+	2.69E+02	+	8.64E+02	+	2.50E+02	+	**1.03E+02**
	Std	1.38E+01		2.75E+01		2.41E+01		3.07E+00		2.36E+01		**1.95E+01**
f_6	Mean	**3.07E-13**	-	2.13E+01	+	5.24E+01	+	1.45E+02	+	6.43E-01	+	5.50E-02
	Std	**9.03E-14**		5.83E+00		3.88E+00		8.94E-01		4.32E-01		6.02E-02
f_7	Mean	**1.65E+02**	-	3.49E+02	+	4.86E+02	+	1.48E+03	+	3.91E+02	+	1.78E+02
	Std	**1.09E+01**		7.47E+01		5.85E+01		1.42E+01		4.06E+01		2.01E+01
f_8	Mean	1.13E+02	+	1.62E+02	+	2.84E+02	+	9.05E+02	+	2.62E+02	+	**1.06E+02**
	Std	1.43E+01		3.15E+01		2.36E+01		2.99E+00		2.62E+01		**1.59E+01**
f_9	Mean	1.01E+03	+	3.40E+03	+	8.75E+03	+	6.73E+04	+	2.02E+03	+	**2.96E+02**
	Std	3.25E+02		1.18E+03		8.12E+02		3.25E+03		1.28E+03		**2.68E+02**
f_{10}	Mean	**3.92E+03**	-	5.22E+03	+	5.62E+03	+	2.06E+04	+	9.24E+03	+	4.50E+03
	Std	**3.43E+02**		4.38E+02		5.66E+02		8.72E+01		7.63E+02		2.69E+02
f_{11}	Mean	1.30E+02	+	1.53E+03	+	2.34E+02	+	1.24E+06	+	2.89E+02	+	**7.12E+01**
	Std	3.72E+01		1.45E+03		4.26E+01		1.88E+05		5.40E+01		**2.33E+01**
f_{12}	Mean	3.79E+06	+	6.93E+06	+	2.07E+07	+	1.43E+11	+	1.00E+10	+	**4.26E+05**
	Std	1.56E+06		2.52E+07		1.56E+07		1.93E+08		0.00E+00		**2.87E+05**
f_{13}	Mean	**4.16E+02**	-	6.78E+03	-	1.98E+04	+	1.13E+11	+	1.00E+10	+	8.38E+03
	Std	**2.20E+02**		4.79E+03		8.22E+03		1.46E+08		0.00E+00		9.85E+03
f_{14}	Mean	3.93E+05	+	8.90E+04	+	9.92E+03	+	1.45E+09	+	**2.86E+02**	-	4.45E+03
	Std	2.37E+05		5.12E+04		6.31E+03		4.99E+06		**4.84E+01**		3.95E+03
f_{15}	Mean	**1.82E+02**	-	1.49E+03	-	4.29E+03	-	2.38E+10	+	3.33E+08	+	5.38E+03
	Std	**1.32E+02**		1.17E+03		1.71E+03		6.80E+07		1.83E+09		5.54E+03
f_{16}	Mean	1.18E+03	+	1.49E+03	+	1.45E+03	+	2.29E+04	+	1.99E+03	+	**1.02E+03**
	Std	2.08E+02		3.08E+02		1.94E+02		5.29E+01		2.52E+02		**2.46E+02**
f_{17}	Mean	7.95E+02	+	1.12E+03	+	1.15E+03	+	1.69E+05	+	1.28E+03	+	**6.05E+02**
	Std	1.27E+02		2.55E+02		1.41E+02		2.67E+03		1.81E+02		**1.89E+02**
f_{18}	Mean	8.29E+05	+	1.03E+06	+	**7.73E+04**	-	2.08E+09	+	2.07E+05	+	9.10E+04
	Std	4.05E+05		1.19E+06		**2.45E+04**		1.67E+07		1.51E+05		6.00E+04
f_{19}	Mean	3.14E+02	-	2.09E+03	-	5.08E+03	-	1.39E+10	+	**1.46E+02**	-	1.52E+04
	Std	3.66E+02		2.97E+03		3.13E+03		5.10E+07		**2.61E+01**		1.27E+04
f_{20}	Mean	6.12E+02	+	6.67E+02	+	8.83E+02	+	3.39E+03	+	1.06E+03	+	**5.65E+02**
	Std	1.66E+02		1.67E+02		1.69E+02		2.95E+01		2.09E+02		**2.45E+02**
f_{21}	Mean	3.23E+02	+	3.61E+02	+	5.38E+02	+	2.24E+03	+	4.57E+02	+	**3.09E+02**
	Std	1.06E+01		2.66E+01		3.50E+01		5.01E+00		2.49E+01		**1.47E+01**
f_{22}	Mean	**4.13E+03**	-	4.97E+03	-	6.79E+03	+	1.89E+04	+	9.29E+03	+	5.17E+03
	Std	**1.57E+03**		2.05E+03		5.94E+02		5.49E+01		1.31E+03		4.45E+02
f_{23}	Mean	5.66E+02	+	6.96E+02	+	1.28E+03	+	7.30E+03	+	7.48E+02	+	**5.43E+02**
	Std	1.15E+01		6.43E+01		1.25E+02		2.04E+01		6.19E+01		**1.83E+01**
f_{24}	Mean	7.64E+02	+	7.71E+02	+	1.14E+03	+	4.45E+03	+	8.44E+02	+	**6.16E+02**
	Std	4.07E+01		8.53E+01		8.22E+01		2.62E+00		7.81E+01		**1.98E+01**
f_{25}	Mean	5.42E+02	-	8.64E+02	+	7.61E+02	+	1.75E+04	+	**5.28E+02**	-	5.45E+02
	Std	1.57E+01		1.93E+02		7.99E+01		3.34E+01		**3.45E+01**		3.44E+01
f_{26}	Mean	**2.11E+03**	-	3.64E+03	+	2.41E+03	+	1.76E+04	+	4.62E+03	+	2.34E+03
	Std	**7.20E+02**		4.53E+02		7.33E+02		3.71E+01		6.03E+02		1.88E+02
f_{27}	Mean	**6.38E+02**	-	1.13E+03	+	1.16E+03	+	1.64E+04	+	8.15E+02	+	6.68E+02
	Std	**2.70E+01**		8.40E+01		1.73E+02		5.25E+01		1.63E+02		6.80E+01
f_{28}	Mean	5.28E+02	+	1.26E+03	+	9.42E+02	+	1.74E+04	+	**4.79E+02**	-	5.06E+02
	Std	1.54E+01		2.71E+02		1.53E+02		2.54E+01		**2.45E+01**		2.34E+01
f_{29}	Mean	7.90E+02	+	1.93E+03	+	1.96E+03	+	6.60E+06	+	1.80E+03	+	**6.62E+02**
	Std	1.38E+02		3.21E+02		2.39E+02		6.91E+04		3.07E+02		**1.64E+02**
f_{30}	Mean	**7.03E+05**	-	3.38E+07	+	1.21E+07	+	2.49E+10	+	2.19E+06	+	8.47E+05
	Std	**5.73E+04**		1.46E+07		2.32E+06		5.65E+07		5.77E+05		2.54E+05
Total	+/=/-	15/0/14		23/0/6		25/0/4		29/0/0		23/0/6		

A more visual evaluation can be obtained from the results of the Friedman tests. The average rankings are shown in Figure 5.5. It can be observed that FOABC has achieved the best rankings in cases of $D = 10$ and 30, while its ranking is very close to that of CLPSO when $D = 50$. In addition, the FOCS stays in the third position in the first two cases but has been overtaken by LIPS when $D = 50$. And the rankings of FOFA are always at the back of the comparisons. Therefore, similar conclusion can be derived from the Friedman tests, the performance of FOABC is very competitive compared to the enhanced variants of other meta-heuristic algorithms, particularly in low-dimensional problems. When compared to other FO-based algorithms, it can also be seen that FOABC performs better in terms of solution accuracy. In this case, the superior efficacy of the proposed FO-based improvement strategy can be validated.

5.3.4 Effectiveness of the proposed strategies

In this part, the effectiveness of each proposed improvement is tested and analyzed. Hence, two variants of FOABC are constructed, namely FOABC$_{DE}$ and FOABC$_{FO}$. The employed bee phase of FOABC$_{DE}$ adopts the DE-based search strategy, i.e., Eq.(5.7), with scale factors following Lévy distribution. And the rest of FOABC$_{DE}$ remains the same as the standard ABC algorithm. Moreover, in order to verify the efficacy of FOC in aiding the search process of ABC, FOABC$_{FO}$ only uses the FO-based solution search equation. More precisely, the proposed Eq.(5.17) is utilized in its onlooker bee phase while the rest of the variant is consistent with the basic one.

These variants are compared with the standard ABC aiming at verifying the effectiveness of the proposed strategies. Experimental results on the CEC 2017 benchmarks with $D = 50$ are presented in Table 5.13. The best results of each function are highlighted in **boldface**. And the results of the Wilcoxon and Friedman tests are given at the bottom of the table. In the Wilcoxon tests, these two variants are compared to the standard ABC and the final FOABC, respectively.

In Table 5.13, the variant FOABC$_{DE}$ and the basic ABC are first observed and compared. It can be noticed that FOABC$_{DE}$ has significant improvements in functions f_{11}, f_{14}, f_{18}, and f_{29}. Meanwhile, the advantage of FOABC$_{DE}$ is verified by the Wilcoxon test results between it and the basic ABC. FOABC$_{DE}$ variant obtains smaller errors than ABC does on 18 out of 29 problems. In this case, the

effectiveness of utilizing improved search equations with Lévy-based scale factors can be validated.

Table 5.13: Effectiveness of each modification of FOABC on benchmarks with $D = 50$

Functions	ABC		FOABC$_{DE}$		FOABC$_{FO}$		FOABC$_{r=12,q=0.8}$	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
f_1	**1.56E+03**	**1.10E+03**	6.67E+03	7.10E+03	1.51E+04	4.35E+03	6.42E+03	6.87E+03
f_3	2.14E+05	3.60E+04	**1.12E+05**	**2.20E+04**	2.12E+05	2.93E+04	1.15E+05	1.74E+04
f_4	4.20E+01	1.51E+01	9.83E+01	5.59E+01	**3.37E+01**	**1.22E+01**	1.19E+02	6.27E+01
f_5	1.98E+02	1.95E+01	1.10E+02	1.58E+01	1.80E+02	1.72E+01	**1.03E+02**	**1.95E+01**
f_6	**7.59E-10**	**5.36E-10**	2.84E-01	3.44E-01	2.66E-04	4.96E-05	5.50E-02	6.02E-02
f_7	2.11E+02	1.60E+01	1.96E+02	3.67E+01	1.98E+02	1.51E+01	**1.78E+02**	**2.01E+01**
f_8	2.04E+02	2.13E+01	1.07E+02	1.92E+01	1.84E+02	2.19E+01	**1.06E+02**	**1.59E+01**
f_9	5.26E+03	1.24E+03	4.35E+02	4.43E+02	4.52E+03	1.63E+03	**2.96E+02**	**2.68E+02**
f_{10}	4.09E+03	3.99E+02	4.59E+03	3.63E+02	**4.01E+03**	**2.99E+02**	4.50E+03	2.69E+02
f_{11}	7.92E+02	9.93E+02	7.78E+01	2.29E+01	5.63E+02	4.40E+02	**7.12E+01**	**2.33E+01**
f_{12}	3.58E+06	1.23E+06	**1.77E+05**	**1.30E+05**	3.48E+06	1.34E+06	4.26E+05	2.87E+05
f_{13}	**5.21E+03**	**3.77E+03**	6.45E+03	8.09E+03	1.14E+04	4.68E+03	8.38E+03	9.85E+03
f_{14}	6.81E+05	4.74E+05	**3.40E+03**	**3.75E+03**	5.98E+05	3.71E+05	4.45E+03	3.95E+03
f_{15}	7.03E+03	4.93E+03	8.10E+03	9.14E+03	9.59E+03	4.67E+03	**5.38E+03**	**5.54E+03**
f_{16}	1.28E+03	1.77E+02	1.06E+03	2.89E+02	1.22E+03	2.77E+02	**1.02E+03**	**2.46E+02**
f_{17}	8.82E+02	1.65E+02	6.92E+02	2.14E+02	8.01E+02	1.51E+02	**6.05E+02**	**1.89E+02**
f_{18}	1.01E+06	5.77E+05	9.84E+04	7.13E+04	9.20E+05	4.85E+05	**9.10E+04**	**6.00E+04**
f_{19}	**8.90E+03**	**4.35E+03**	9.54E+03	9.91E+03	9.53E+03	4.06E+03	1.52E+04	1.27E+04
f_{20}	7.44E+02	1.27E+02	**5.64E+02**	**2.15E+02**	7.59E+02	1.45E+02	5.65E+02	2.45E+02
f_{21}	4.05E+02	4.75E+01	**3.08E+02**	**1.73E+01**	3.98E+02	1.98E+01	3.09E+02	1.47E+01
f_{22}	5.19E+03	3.68E+02	5.17E+03	1.03E+03	**4.86E+03**	**9.54E+02**	5.17E+03	4.45E+02
f_{23}	6.53E+02	5.07E+01	5.46E+02	2.79E+01	6.46E+02	3.77E+01	**5.43E+02**	**1.83E+01**
f_{24}	1.02E+03	5.98E+01	6.19E+02	2.85E+01	9.83E+02	5.72E+01	**6.16E+02**	**1.98E+01**
f_{25}	5.12E+02	1.84E+01	5.43E+02	3.67E+01	**5.07E+02**	**1.73E+01**	5.45E+02	3.44E+01
f_{26}	**1.39E+03**	**1.47E+03**	2.44E+03	2.12E+02	1.73E+03	1.50E+03	2.34E+03	1.88E+02
f_{27}	6.56E+02	2.16E+01	6.82E+02	7.66E+01	**6.45E+02**	**3.21E+01**	6.68E+02	6.80E+01
f_{28}	4.87E+02	1.29E+01	5.03E+02	1.25E+01	**4.82E+02**	**1.33E+01**	5.06E+02	2.34E+01
f_{29}	1.08E+03	1.49E+02	7.53E+02	1.59E+02	9.91E+02	1.25E+02	**6.62E+02**	**1.64E+02**
f_{30}	7.51E+05	6.66E+04	7.14E+05	1.13E+05	**7.14E+05**	**4.57E+04**	8.47E+05	2.54E+05
Wilcoxon (+/=/-)								
v.s. ABC			18/0/11		22/0/7			
v.s. FOABC			12/0/17		10/0/19			
Friedman	3		2.31		2.55		**2.14**	

Secondly, in order to test the effectiveness of the embedding of FOC, the variant FOABC$_{FO}$ is compared with ABC algorithm. It can be found that, FOABC$_{FO}$ achieves better results on 22 benchmarks. Hence, the algorithm's performance can be improved by using FOC in the onlooker bee phase.

Furthermore, these two versions are compared with the final FOABC algorithm as well. From the Wilcoxon test results, these two variants can be found to be better than the basic ABC but less effective than the final version we proposed.

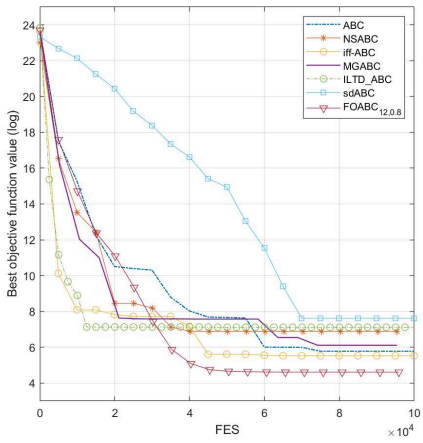
FOABC$_{DE}$ is unable to outperform FOABC on 17 functions, while FOABC$_{FO}$ also fails on 19 benchmarks. And a similar conclusion can be drawn from the number of best results marked in boldface.

According to the average rankings given by the Friedman test, the final FOABC obtains the best ranking. Meanwhile, the rankings of FOABC$_{DE}$ and FOABC$_{DE}$ are both better than those of ABC. In this case, the effectiveness of each proposed strategy can be proved. Moreover, it can be concluded that these modifications perform better when they are used cooperatively than when they are used singly.

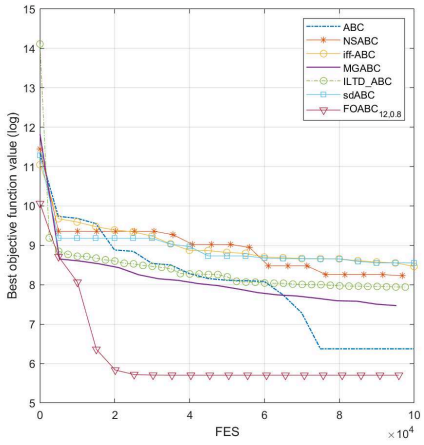
5.3.5 Convergence behavior analysis

In this part, the convergence rates of involved ABC algorithms are also compared together by plotting their convergence processes of six representative benchmark functions as shown in Figure 5.6. Note that, the values of $log(f)$ is considered since the objective function values are too large to plot in the earlier stage of searching process.

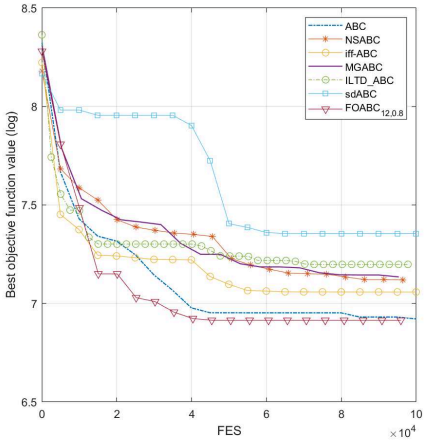
In Figure 5.6, (a) and (b) present the convergence curves of ABC algorithms on two uni-modal functions. It can be found that FOABC is able to reach the best solution precision and its convergence rate is competitive to other methods. Similar conclusion can be obtained from the figure (c) of the multi-modal function f_{10}. Although *iff*-ABC and ILTD_ABC converge faster in the early stage, they fail to determine a more precise solution. As for the hybrid and composition functions, the solutions found by most ABC algorithms are close to each other. It is obvious that ILTD_ABC and FOABC converge more rapidly than other competitors. Therefore, in terms of solution accuracy, the FOABC algorithm has outstanding performance, especially in solving uni-modal and hybrid functions. And its convergence rate is very competitive compared to other ABC variants.



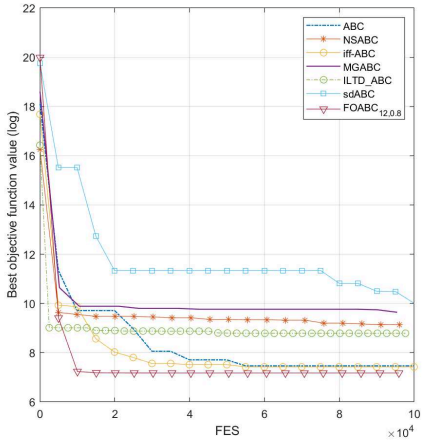
(a) f_1



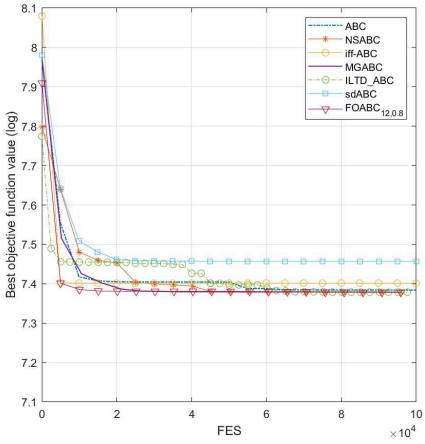
(b) f_3



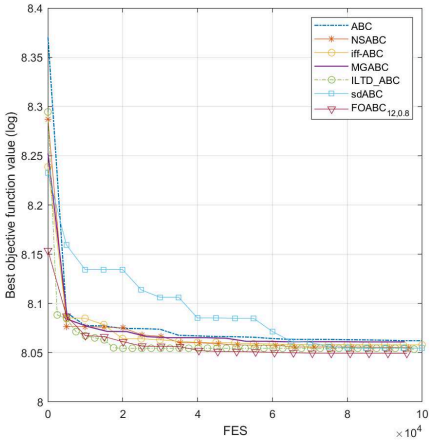
(c) f_{10}



(d) f_{13}



(e) f_{16}



(f) f_{29}

Fig. 5.6. The convergence performance of FOABC and compared ABC algorithms

5.4 Conclusion

With the purpose of enhancing the performance of ABC algorithm, a new fractional-order ABC algorithm (FOABC for short) is proposed in this chapter. Firstly, a DE-based search strategy is used in the employed bee phase to keep a nice balance between the exploration and exploitation. Secondly, in order to improve the local search ability, the FOC is incorporated into the search strategy of onlooker bees. In this way, the memory feature of FOC enriches the amount of available information for onlookers. Precisely, the information of the last several steps is considered when generating new solutions. Moreover, the random number used in the search equations is drawn from Lévy distribution in order to increase the randomness of the searching process.

Experiments are carried out to study the sensitivity of FOABC with respect to the parameters of FOC. Groups of experiments are conducted on CEC 2017 benchmark functions to evaluate the performance of FOABC from different perspectives. In the first two groups, FOABC is compared with six ABC algorithms and five other enhanced meta-heuristics, respectively. According to the results, FOABC outperforms the other ABC algorithms in terms of solution accuracy and robustness. Meanwhile, it also performs outstandingly in the second group of comparisons. Thirdly, redundancy elimination experiments are done to verify the effectiveness of each proposed strategy. As a result, FOABC has excellent performance in handling numerical optimization problems.

Chapter 6

Robot path planning via improved ABC algorithms

Contents

6.1 Introduction

After investigating different improvement strategies to enhance the effectiveness of the ABC algorithm in the previous chapters, applying them to some more meaningful practical problems is to be expected. At the same time, observing the importance and booming development of robotics-related fields, certain problems in this area are selected to be investigated in this chapter.

One of the most popular problems in domains linked to robotics is robot path planning (RPP). It has attracted considerable attention since path planning issues have widely existed for different types of robots, including industrial, mobile, etc. RPP problem aims at finding out an optimal collision-free path from the start point to the target point in an environment with obstacles (Fragapane *et al.*, 2021; Nazarahari *et al.*, 2019). And a series of optimization problems can be formulated by considering different goals and constraints. For instance, energy cost (or fuel cost) is one of the essential terms that needs to be minimized.

As mentioned in the first chapter, path planning algorithms can be generally classified into three categories: classical, graph-based, and (meta-)heuristic approaches (Koubâa *et al.*, 2018). The classical methods like the artificial potential fields method were popular in the period when the path planning problem was just appearing. They were found to be effective in finding feasible paths, yet the shortcomings became increasingly evident. One of the widely known limitations is that they are time-consuming, as the generated solutions are computationally expensive. Besides, these kinds of methods might fall into local optima (Das & Jena, 2020; Koubâa *et al.*, 2018). As for the second category, graph-based search methods are also popular, such as the A* algorithm (Hart *et al.*, 1968; Hawa, 2013). The feasible paths can be computed in a graph-based environment, such as a grid map. Nonetheless, in challenging environments, the complexity of the computation may increase significantly. Therefore, both types of approaches cannot guarantee fast and efficient path finding when dealing with complex environments. Thus, the demand for intelligence in solving different RPP challenges is increasing today.

To overcome the above limitations, the third class of methods has gained great attention. Like mentioned before, meta-heuristic algorithms are outstanding in solving various optimization problems and are not problem-specific. These features make it one of the ideal methods to accomplish different tasks. The

RPP problems can be solved by transforming them into functional optimization problems. Then, the optimal solution can be found by using meta-heuristic algorithms, which is effective and powerful even in complex environments. In recent decades, PSO (Das *et al.*, 2016a; Das & Jena, 2020; Thabit & Mohades, 2019; Tian *et al.*, 2021), GA (Nazarahari *et al.*, 2019; Tuncer & Yildirim, 2012), ACO (Hasan & Mosa, 2018; Lyridis, 2021; Miao *et al.*, 2021), and ABC (Abbas & Ali, 2014; Bhattacharjee *et al.*, 2011; Contreras-Cruz *et al.*, 2015; Liang & Lee, 2015; Wang *et al.*, 2015; Xu *et al.*, 2020) have all been popular meta-heuristics for such problems.

Particularly, ABC and its variants have been used to solve various RPP problems. Abbas & Ali (2014) proposed an improved ABC algorithm for the off-line autonomous RPP problem. Path length and smoothness were considered in the objective function. Contreras-Cruz *et al.* (2015) introduced a hybrid path planner ABC-EP to solve the RPP problem. The connections between the line segments were optimized via ABC locally. Then, the feasible path was optimized by the evolutionary programming (EP) algorithm in terms of length and smoothness. The performance of ABC-EP planner was proved by comparing it with a classical approach, the probabilistic roadmap method (PRM). A coevolution framework was incorporated in ABC while the global best individual was utilized in the solution search equation (Xu *et al.*, 2020). And the proposed ABC variant was examined on RPP problems aiming at minimizing the path length.

Furthermore, ABC variants were also adopted to achieve multi-robot path planning (MRPP) problems. Bhattacharjee *et al.* (2011) utilized ABC to plan paths for multiple mobile robots. More precisely, the subsequent positions of all the robots were determined via an ABC algorithm aimed at minimizing the path length. Similarly, Liang & Lee (2015) used an effective ABC variant to solve a challenge of on-line path planning for multiple mobile robots. The proposed algorithm was enhanced by using an elite group and solution-sharing strategy. A proper objective function was defined for target, obstacles, and robot collision avoidance. Note that different from the SRPP (single robot path planning) problem, the collisions between teammates also need to be avoided. Wang *et al.* (2015) solved a multi-objective MRPP problem by using an improved multi-objective ABC algorithm. A multi-objective function was defined considering the path length, safety, and smoothness. Then a solution set can be obtained by using the

Pareto dominance relationship. Each path point was positioned by optimizing the angle since the robot step length was constant. Multiple feasible paths were generated through the proposed ABC.

Therefore, different types of RPP problems are concerned in this chapter. Firstly, the proposed four ABC variants are applied to complete the global path planning for a single robot. Different environments with arbitrary obstacles are adopted. The proposed ABC algorithms are compared to ten improved ABC and the standard ABC algorithm as well as four well-known RPP approaches. Secondly, the MRPP task is considered as well. The proposed ABCL algorithm is utilized to solve a local path planning problem for multiple robots. Simulations and comparisons are done to verify the effectiveness of ABCL.

The outline of this chapter is as follows. The single robot path planning (SRPP) problem is achieved in section 6.2. Then, the multi-robot path planning (MRPP) problem is concerned in section 6.3. Both of these sections include the problem formulations, simulations, and comparisons. The conclusion is given in section 6.4.

6.2 Single robot path planning (SRPP)

6.2.1 Problem formulation of SRPP

The RPP environment includes arbitrary obstacles and free space. An example is given in Figure 6.1. The **S** and **T** are the start and target points of the robot. And O_1, O_2, O_3 are static obstacles that need to avoid. The goal is to determine the optimal path without any collisions with obstacles from the start point to its desired target. With this purpose, the RPP problem can be converted to an optimization problem and solved with the proposed improved ABC algorithms.

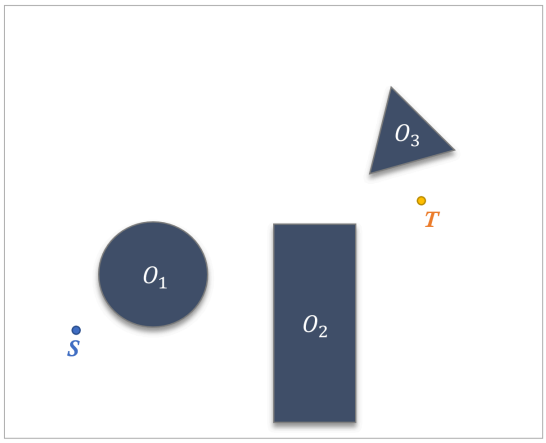


Fig. 6.1. Example of RPP environment

The implementation method is shown in Figure 6.2. Firstly, the start point **S** and the target point **T** are connected as line **ST**. Then a new coordinate system is designed by setting the line **ST** as the new x-axis (x' in Figure 6.2). In other words, **S** becomes the origin of the coordinate system $x'Sy'$. Hence, the new coordination (x_n, y_n) of each point in the original coordinate system XOY can be calculated via Eq.(6.1).

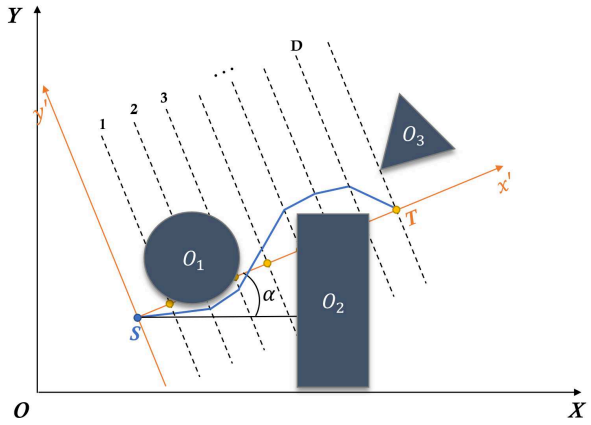


Fig. 6.2. Implementation method of robot path planning

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} \times \left(\begin{bmatrix} X \\ Y \end{bmatrix} - \begin{bmatrix} x_S \\ y_S \end{bmatrix} \right), \tag{6.1}$$

where

$$\alpha = \arcsin \frac{y_T - y_S}{|\mathbf{ST}|}, \tag{6.2}$$

α denotes the rotate angle from original X-axis to line **ST**. (x_n, y_n) is the corresponding coordinate of (X, Y) in new coordinates system and (x_S, y_S) is the coordinate of **S** in XOY original system.

After that, the line **ST** is divided into $(D+1)$ equal parts and the divided points can be obtained via Eq.(6.3).

$$\delta(d) = d \times \frac{|\mathbf{ST}|}{D+1}, \quad d = 1, 2, \cdots, D. \tag{6.3}$$

Vertical lines are drawn on each divided point as well as points **S** and **T**. Then a feasible path can be obtained by selecting one node on each vertical line and connecting the nodes together sequentially. Thus, the RPP problem is transformed into a D-dimensional optimization problem. And a candidate solution (i.e., a candidate path) is in form of $p_i = (\mathbf{S}, p_{i,1}, p_{i,2}, \cdots, p_{i,D}, \mathbf{T})$, $i \in \{1, \cdots, SN\}$.

With the purpose of planning collision-free paths via the meta-heuristic optimization algorithms, two strategies are utilized: firstly, for each candidate path p_i, the feasibilities of all its nodes $p_{i,1}, p_{i,2}, \cdots, p_{i,D}$ are checked; secondly, each section of a candidate path is checked step by step to see if it has hit an obstacle.

To explain the strategies clearly, the corresponding pseudo-code is described in Algorithm 14. Each time the candidate solutions are generated or updated, this mechanism is adopted to avoid collisions. If certain nodes encounter obstacles or leave the workspace, they will be directly replaced by a newly produced feasible nodes (lines 1-7 in Algorithm 14). Hence, it is also necessary to verify if the connections of every two adjacent nodes in a candidate path collide with the obstacles. More specifically, for each line segment (i.e., $\overline{\mathbf{S}p_{i,1}}, \overline{p_{i,1}p_{i,2}}, \cdots, \overline{p_{i,D}\mathbf{T}}$), the inner points are checked one by one with a step increment of 1. And if there are z points encountering the obstacles, a penalty $\lambda = z \times 5000$ will be imposed on the objective function value as shown in Eq.(6.4). In this way, collision-free paths can be obtained not only by ensuring the feasibility of all the nodes, but also by determining whether the node connections come across the obstacles.

The objective function is defined as

$$F = length(p_i) + \lambda, \tag{6.4}$$

where $length(\cdot)$ is the function to calculate the total length of candidate path

p_i. λ is a penalty term, which is equal to 0 when all the nodes and segments lie in the free space. Otherwise, $\lambda = z \times 5000$ when there exist z inner points located at infeasible positions. Hence, the objective function is penalized when the candidate path encounters obstacles.

Algorithm 14 Mechanism for generating collision-free path in SRPP

for each candidate path p_i **do**
  for $j = 1 : D$ **do**
    if node $p_{i,j}$ is outside the map **or** encounters an obstacle **then**
      Reproduce a new feasible node $p_{i,j}^{new}$
      Replace the infeasible $p_{i,j}$ with $p_{i,j}^{new}$
    end if
  end for
  Set $\lambda = 0$
  for all the line segments $\overline{p_{i,j}p_{i,j+1}}$ **do**
    Calculate the slope θ_j and segment length l_j
    for $m = 0 : 1 : l_j$ **do** ▷ Step by increments of 1
      Calculate the inner point $ip = p_{i,j} + m \times [sin(\theta_j), cos(\theta_j)]^T$
      if point ip is infeasible **then**
        $\lambda = \lambda + 5000$
      end if
    end for
  end for
  Evaluate the objective function value with Eq.(6.4)
end for

Remark 6.1 *Note that, like many other heuristic algorithms, the ABC algorithm is initially designed for unconstrained optimization problems, and it can be used to solve constrained optimization problems by adding penalties (Abbas & Ali, 2014). Nonetheless, there also exist constrained optimization algorithms which are established specifically for the constrained optimization problems (Szczepanski et al., 2018).*

Besides, to better describe the process of applying meta-heuristic algorithms to resolve the SRPP problem, a flowchart with ABC_RL algorithm, as an example, is given in Figure 6.3.

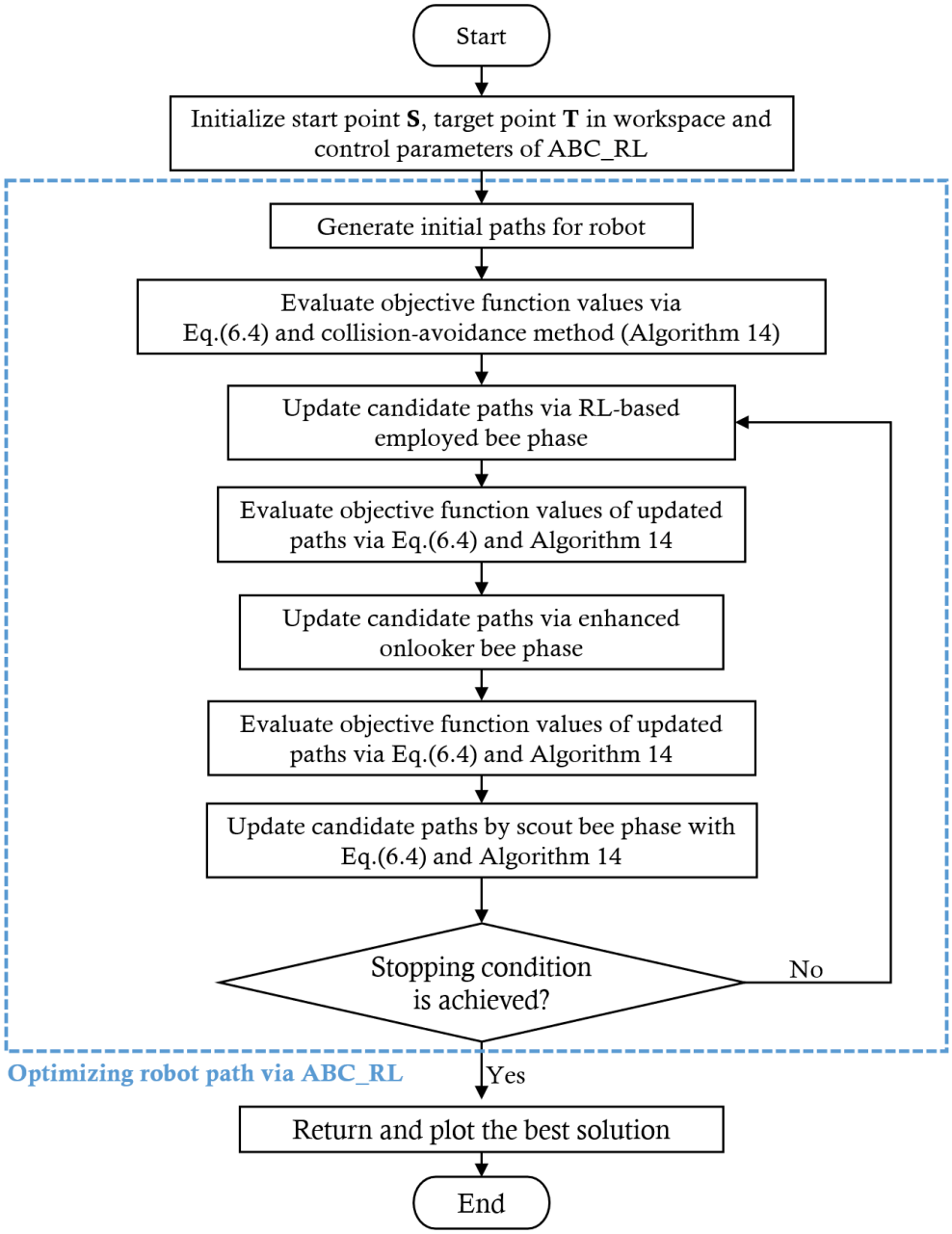


Fig. 6.3. Flowchart of SRPP by the proposed ABC_RL algorithm

6.2.2 Simulation results of SRPP

Simulations and comparisons have been done to investigate whether the proposed algorithms could effectively solve the real-world problems. So in this part, comparisons are done first among those ABC variants that were covered in the previous chapters. Then, comparisons are also carried out with a few popular path planners as presented in subsubsection 6.2.2.2. The simulations are run with MATLAB on Intel Core(TM) i7 processor with 2.30GHz CPU and 16GB of memory.

6.2.2.1 Comparison with ABC variants

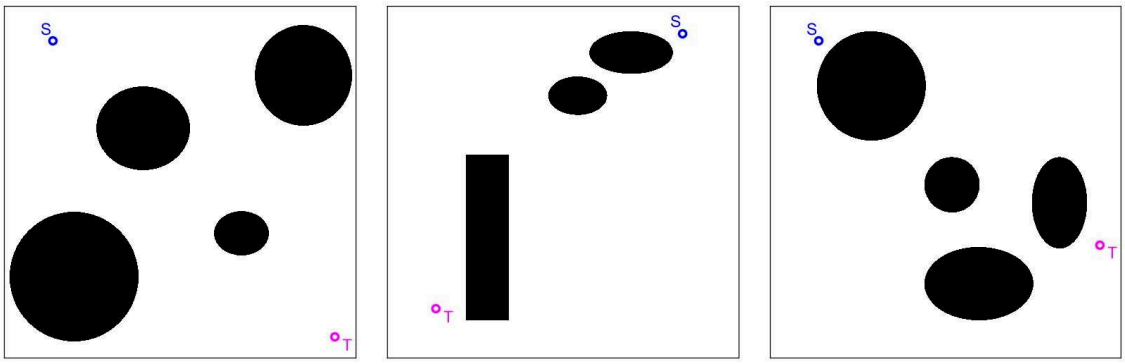
All of the ABC variants that were mentioned in comparisons of previous chapters are used to accomplish the same set of SRPP tasks in order to demonstrate that our proposed ABC algorithms can not only achieve outstanding results on functional optimization problems but also have advantages in actual problems. The involved 15 ABC algorithms are listed as follows: the standard ABC (Karaboga, 2005), NSABC (Wang *et al.*, 2020), *iff*-ABC (Aslan *et al.*, 2020), MGABC (Zhou *et al.*, 2021a), ILTD_ABC (Gao *et al.*, 2019), sdABC (Chen *et al.*, 2019b), DE-ABC (Li & Yin, 2014), APABC (Cui *et al.*, 2017b), ARABC (Cui *et al.*, 2017a), GABC (Zhu & Kwong, 2010), DABC (Abbas & Ali, 2014) and four improved ABC algorithms that we proposed, namely, ABCL, ABCDC, ABC_RL, and FOABC.

With the purpose of achieving SRPP, all the participating algorithms need to solve the problem defined in Eq.(6.4), with SN set to 20, $D = 15$ and the determination condition is set as $max\_FES = 1500 \times D$. The other parameter settings remain the same as in previous chapters. And the efficiency of our proposed algorithms is assessed through six different maps. The initial configurations of six SRPP tasks are presented in Figure 6.4 with corresponding start point **S** and target point **T**. And the complexity of maps varies since they contain different numbers of obstacles of various shapes.

The 15 ABC algorithms have all been independently run 30 times for each map. The mean and standard deviation of the path lengths are calculated and presented in Table 6.1 and Table 6.2 (columns Avg. path length and Std., respectively). The shortest path and the longest path are also listed in the tables. Moreover, in order to make a meaningful comparison, the average running time

of each method is considered as well. The best values for each map are marked in **boldface**.

Remark 6.2 *The parameter settings can be adjusted depending on the map scale, the complexity of SRPP tasks, etc.*



(a) Map 1 **(b)** Map 2 **(c)** Map 3

(d) Map 4 **(e)** Map 5 **(f)** Map 6

Fig. 6.4. The initial configurations of six SRPP workspaces

First of all, it can be observed from Table 6.1 and Table 6.2 that the SRPP problems can be effectively solved utilizing the problem formulated previously. And all the compared algorithms completed the tasks successfully in their 30 individual runs. As a result, the adopted mechanism for searching a collision-free path for a single robot in static environments can be validated.

Among the total 15 ABC algorithms, our proposed algorithms have outstanding performance, especially our proposed FOABC and ABCL algorithms. It is obvious that FOABC achieves the minimum value on the average of the path

Table 6.1: Comparison of 15 ABC algorithms for solving SRPP problems

	ABC algorithms	Avg. path length	Std.	Best path	Worst path	Avg. time
Map 1	ABC	1280.285	434.217	642.888	2400.394	18.697
	NSABC	676.598	64.238	610.543	918.288	15.806
	iff-ABC	603.452	1.694	601.433	609.124	13.732
	MGABC	616.263	53.335	601.228	835.533	15.352
	ILTD_ABC	601.300	0.067	601.224	601.525	10.865
	sdABC	696.852	35.290	635.149	763.268	17.281
	DEABC	688.479	19.459	641.238	718.374	24.348
	APABC	1221.639	223.856	777.256	1671.565	15.605
	ARABC	603.819	1.680	601.633	608.253	15.830
	GABC	603.440	1.695	601.634	608.875	15.922
	DABC	607.034	2.829	602.783	611.509	15.653
	ABCL	602.702	0.995	601.480	606.379	10.859
	ABCDC	621.947	6.591	608.128	635.182	15.061
	ABC_RL	603.408	1.346	601.794	605.932	**10.239**
	FOABC	**601.218**	**0.020**	**601.206**	**601.314**	10.286
Map 2	ABC	1130.682	352.622	609.450	2110.220	13.892
	NSABC	749.101	101.967	617.486	939.424	13.648
	iff-ABC	599.799	3.139	596.055	612.592	14.933
	MGABC	638.709	118.178	595.158	1151.773	15.991
	ILTD_ABC	595.983	0.483	595.186	597.119	9.871
	sdABC	621.204	15.326	603.582	678.450	12.277
	DEABC	659.099	26.484	611.479	705.744	19.338
	APABC	1415.849	220.580	916.688	1924.129	14.589
	ARABC	604.381	5.379	595.791	616.853	14.057
	GABC	600.522	3.398	595.683	607.277	14.058
	DABC	603.945	4.649	596.472	614.497	15.939
	ABCL	599.871	2.776	595.468	606.456	10.066
	ABCDC	618.030	8.389	607.615	634.652	**8.929**
	ABC_RL	603.119	4.381	597.063	614.470	10.772
	FOABC	**595.344**	**0.274**	**595.021**	**596.280**	11.850
Map 3	ABC	1276.584	527.046	627.844	2640.232	16.501
	NSABC	630.923	62.971	561.171	773.659	13.132
	iff-ABC	550.471	0.831	549.268	552.079	12.198
	MGABC	708.433	215.906	551.678	1187.997	14.546
	ILTD_ABC	549.968	0.427	549.300	551.019	10.436
	sdABC	601.282	23.755	565.389	673.608	15.844
	DEABC	630.072	19.302	573.838	663.645	23.576
	APABC	1048.846	150.078	816.072	1427.738	13.562
	ARABC	551.765	1.963	549.581	557.055	14.501
	GABC	550.581	0.856	549.348	553.035	14.752
	DABC	550.953	1.137	549.230	554.250	10.780
	ABCL	549.955	0.536	549.296	551.434	**9.493**
	ABCDC	560.967	6.329	550.466	575.777	12.784
	ABC_RL	550.917	1.945	549.060	558.013	12.112
	FOABC	**549.283**	**0.244**	**548.943**	**550.038**	10.855

Table 6.2: (*continued*) Comparison of 15 ABC algorithms for solving SRPP problems

	ABC algorithms	Avg. path length	Std.	Best path	Worst path	Avg. time
Map 4	ABC	1374.173	751.638	575.924	3825.581	17.695
	NSABC	795.414	214.625	536.574	1339.219	16.123
	iff-ABC	537.987	8.294	525.905	558.539	13.751
	MGABC	562.355	94.386	527.268	1033.116	13.660
	ILTD_ABC	526.093	**1.150**	524.806	**530.745**	11.228
	sdABC	557.500	13.579	532.505	590.765	15.461
	DEABC	819.613	32.598	749.023	877.275	21.789
	APABC	1629.682	318.632	1160.452	2502.570	11.982
	ARABC	544.193	9.119	528.528	570.100	13.914
	GABC	540.554	6.945	527.041	554.018	14.092
	DABC	537.939	3.625	534.510	551.403	15.203
	ABCL	537.731	8.400	525.540	555.251	**10.342**
	ABCDC	554.008	7.225	540.861	569.026	12.316
	ABC_RL	538.458	9.590	525.358	557.892	12.909
	FOABC	**525.535**	1.725	**524.716**	532.401	11.101
Map 5	ABC	1499.821	569.741	759.605	2983.055	16.025
	NSABC	830.759	158.392	638.792	1311.277	15.863
	iff-ABC	620.451	0.796	619.283	622.048	13.533
	MGABC	633.949	53.919	619.095	873.485	15.294
	ILTD_ABC	619.212	**0.099**	619.044	**619.451**	11.668
	sdABC	687.238	41.668	633.728	788.478	19.736
	DEABC	731.605	24.496	662.920	774.724	26.610
	APABC	1419.491	190.415	1035.945	1704.878	16.099
	ARABC	623.769	7.135	619.565	659.830	13.380
	GABC	620.706	0.923	619.272	622.854	13.480
	DABC	621.505	2.164	619.340	631.606	16.790
	ABCL	620.701	1.551	619.175	626.336	**11.186**
	ABCDC	648.908	6.538	632.713	666.528	15.344
	ABC_RL	625.622	25.916	619.162	762.696	13.634
	FOABC	**619.142**	0.137	**619.013**	619.710	12.364
Map 6	ABC	1563.006	781.150	651.615	3581.080	15.364
	NSABC	727.292	152.099	561.464	1076.331	14.519
	iff-ABC	539.038	0.283	538.668	539.683	11.914
	MGABC	772.555	283.193	544.013	1509.287	17.931
	ILTD_ABC	539.054	0.364	538.633	540.301	10.081
	sdABC	668.048	64.326	570.736	836.237	18.051
	DEABC	714.867	27.531	648.334	772.856	24.597
	APABC	1300.744	198.569	1003.654	1738.505	14.852
	ARABC	539.095	0.338	538.560	540.087	12.548
	GABC	539.497	0.630	538.828	541.467	13.108
	DABC	539.998	0.747	538.694	542.101	15.038
	ABCL	539.069	**0.268**	538.572	539.707	**9.230**
	ABCDC	594.741	11.874	575.052	630.126	12.512
	ABC_RL	539.195	0.565	538.552	540.969	11.913
	FOABC	**538.723**	0.315	**538.459**	**539.672**	10.954

length in all six cases. Moreover, its standard deviation is relatively small, which indicates that FOABC performs more consistently and reliably than other ABC variants. Additionally, the comparisons show that the ABCL and ILTD_ABC algorithms also produce outstanding results. It is discovered that these three methods have very comparable results in terms of average path length and average execution time. Besides, it is interesting to see that as the complexity rises, the performance of the *iff*-ABC algorithm becomes more noticeable. Particularly in Map 6, it outperforms the ABCL and ILTD_ABC in terms of average path length. However, it should be noted that the average running time of the *iff*-ABC algorithm is not as impressive as the aforementioned three ABC methods. Meanwhile, the proposed ABCL algorithm has significant advantages in average runtime thanks to its simple structure and the strategy we adopted to simplify the computational complexity.

In fact, these compared algorithms can be generally classified into three tiers based on the comparison results. Firstly, across every concerned situation, the first four algorithms (i.e., FOABC, ABCL, ILTD_ABC, and *iff*-ABC) perform well and remain stable. Following them, there are several algorithms whose outcomes are rather close, namely ABC_RL, GABC, ARABC, DABC, MGABC, and ABCDC algorithms. Among these algorithms, GABC, DABC, and our proposed ABC_RL outperform the others in terms of overall performance, and the difference from the top tier is also relatively small. Moreover, the standard deviation of MGABC is noticeably higher than that of the others, although their mean values are quite similar. In this context, it can be inferred that MGABC is less stable compared to competitors in the same tier. In addition, considering the average runtime, ABC_RL is more dominant in this second group.

Then, the rest of the algorithms fail to stand out in the comparisons, indicating that they are not effective at resolving this kind of SRPP problem. And the last group of algorithms needs a relatively long time to accomplish the tasks.

Furthermore, to give a more visual picture of the compared algorithms' performance, their rankings across the six tasks are plotted based on average path length and average run time in Figure 6.5 and Figure 6.6, respectively. Note that lower bars in both bar graphs denote better outcomes.

Similar conclusions can be seen in Figure 6.5 and Figure 6.6, which demonstrate that FOABC, ABCL, and ILTD_ABC algorithms surpass the other com-

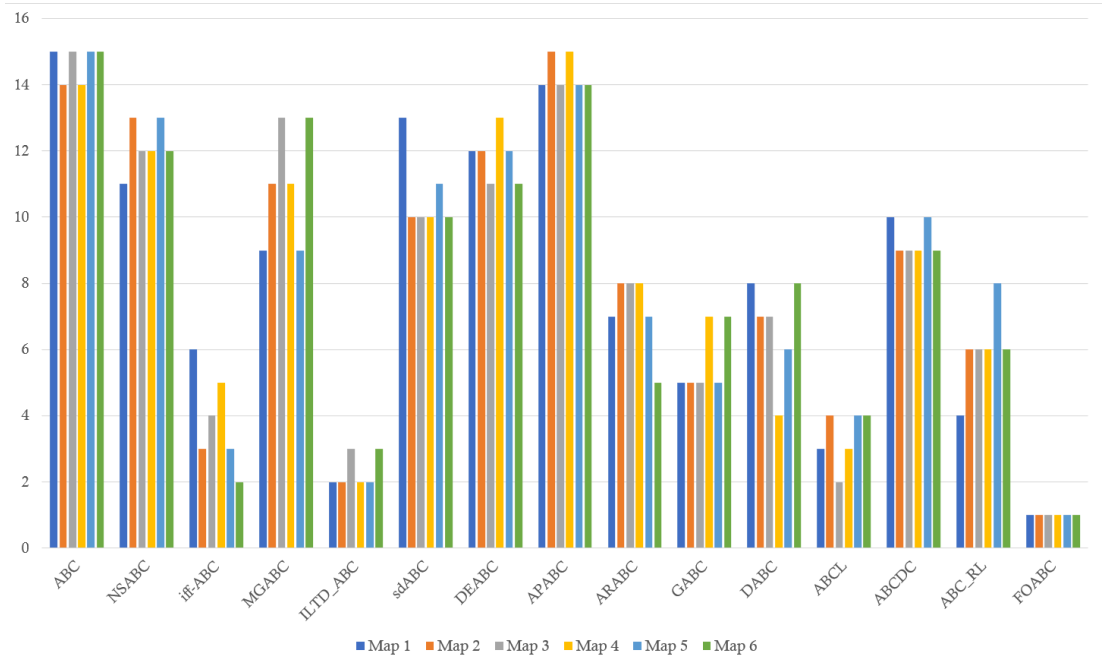


Fig. 6.5. Rankings of compared ABC algorithms based on average path length over 30 independent executions

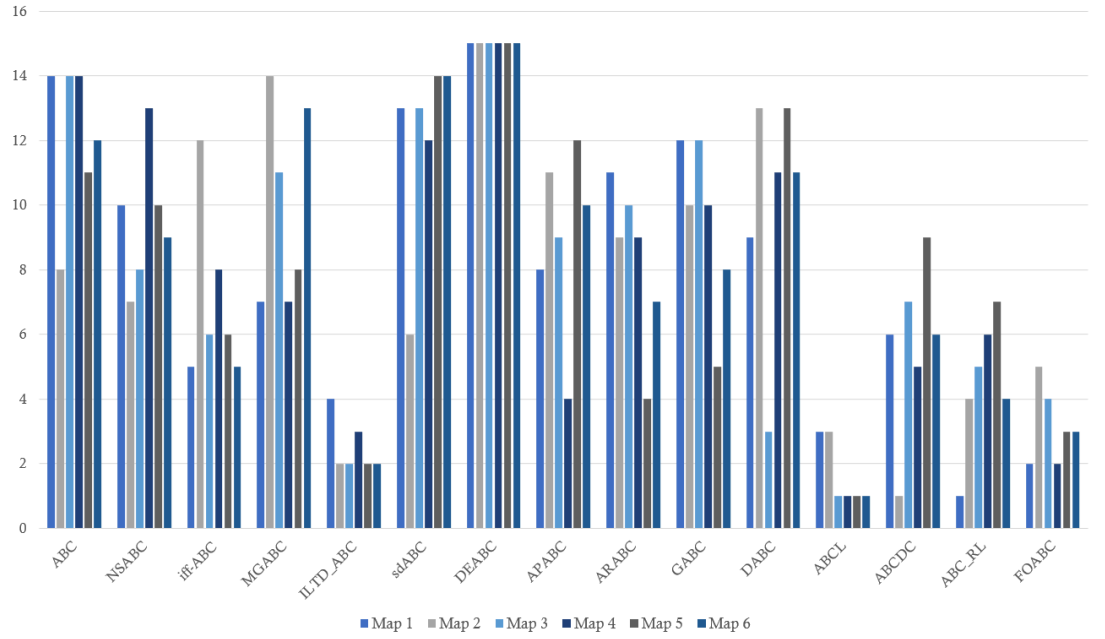


Fig. 6.6. Rankings of compared ABC algorithms based on average running time over 30 independent executions

petitors in terms of overall performance. More precisely, FOABC obtains the first place in all tasks in terms of the average path length. And all of these three winners can successfully find paths with short lengths and fast running times within a limited number of iterations. Due to the average time consumption of *iff*-ABC being higher than these three algorithms, it is not included as a winner.

In addition, more discussions regarding the simulation results can be found below. Actually, the problem scenario concerned above is a uni-modal optimization problem, which means that a unique optimal solution can be found given enough time. Since the primary objective of the defined SRPP is to minimize the path length without collision, the optimal solution in this case refers to a collision-free path with the shortest path length. Therefore, it is evident from the simulations that the three winners are efficient at resolving this type of uni-modal optimization problem. Meanwhile, it is possible that some improvement strategies intended for complete optimization issues do not properly assist the algorithms in finding the optimal solution to this practical problem, and even affect the convergence effect. Therefore, certain algorithms that performed well in the previous chapters were unable to maintain their superiorities in this SRPP problem. For instance, APABC, MGABC, NSABC, and the original ABC algorithm have mediocre performance in the simulations above. Nevertheless, note that these algorithms might be better suited to deal with other real-world issues.

6.2.2.2 Comparison with well-known path planners

In addition, it is also important to compare with other well-known path planning methods. Hence, comparisons are carried out between our proposed ABC algorithms and four well-known global path planners: A* algorithm, Probabilistic Road Map (PRM), Rapidly-exploring Random Trees (RRT), and Bidirectional Rapidly-exploring Random Trees (BRRT), which are widely used to find the shortest path. Note that the code implementation made reference to the resources (Kala, 2014a,b,c,d).

In the following simulations, according to the scale of environments, the number of random points to be selected in PRM is defined as 30. The step sizes of RRT and BRRT are set as 10. The parameter settings of the proposed four ABC algorithms are defined the same as above. Moreover, the comparison results of each method are calculated based on 30 independent executions as presented in

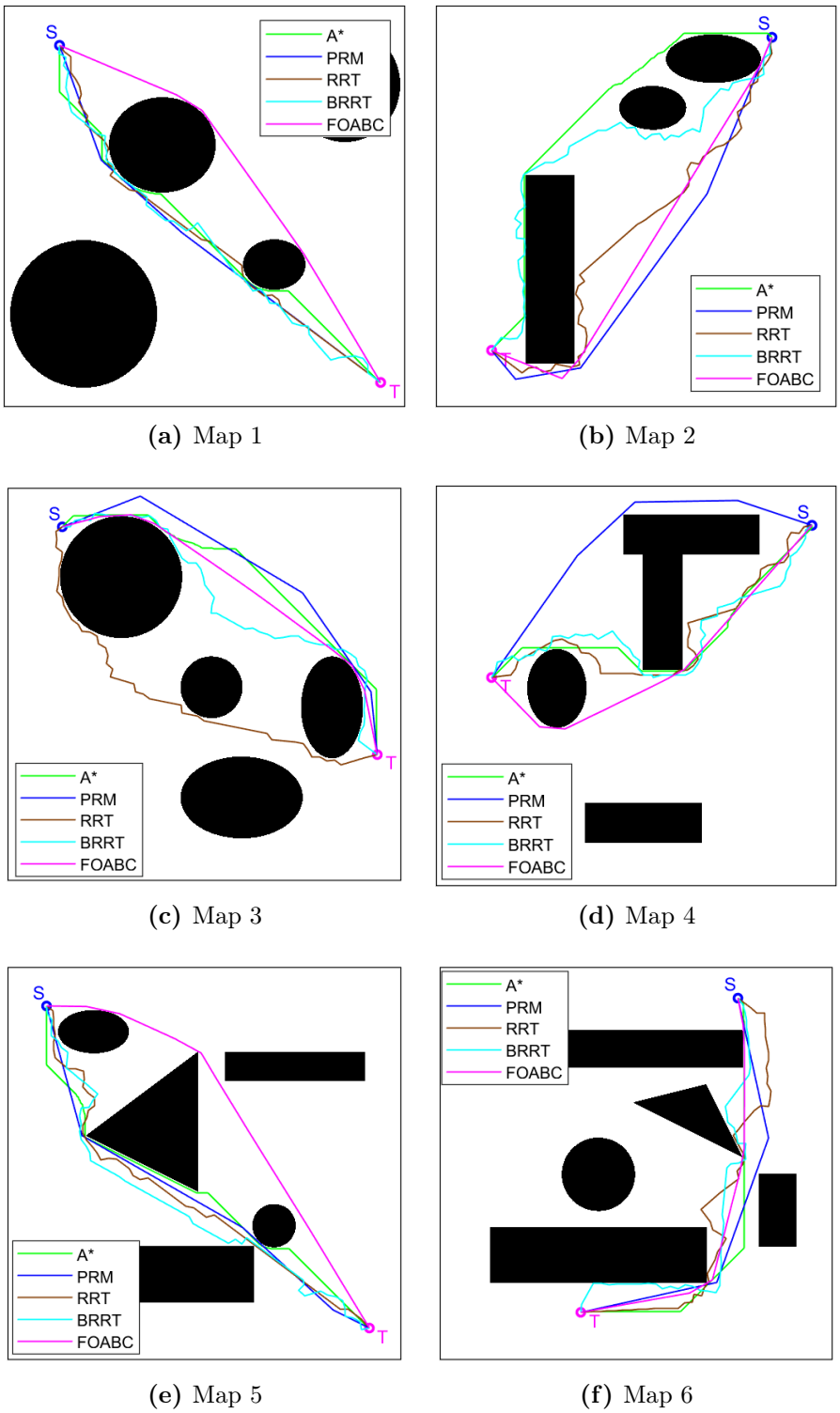
Table 6.3. The best values are marked in **boldface**. Furthermore, the shortest paths planned by the FOABC algorithm and the four well-known path planning approaches are drawn in Figure 6.7.

From Table 6.3, it is encouraging to find that our proposed approaches, especially the FOABC algorithm, are very comparable to those well-known methods. In other words, the performance of proposed algorithms in searching for the optimal collision-free path has been well demonstrated. In terms of average path length, all approaches outperform the other four path planners in Maps 2, 3, and 5. And the average path length obtained by FOABC is the shortest in 5 out of 6 workspaces. As for Maps 1 and 6, excluding ABCDC, the other three improved ABC algorithms are also outstanding. Be aware that ABCDC performs slightly worse than PRM in Map 1 and A* in Map 6, respectively.

A*, a kind of exact path planning method, is able to find a competitive path among the approaches. It is worth pointing out that it achieves the shortest average path length in Map 4. Nonetheless, A* is significantly time-consuming as it generates computationally expensive solutions. PRM is also comparative in terms of path length and running time. Its execution time is the least among the concerned methods. Additionally, because the degrees of complexity of concerned maps are different, the running times of RRT, BRRT, and A* have been significantly impacted. Meanwhile, it can be found that the increase in difficulty can hardly affect the running time of our proposed ABC algorithms as well as PRM. Therefore, it can be concluded that the proposed algorithms are effective methods for the SRPP problem considering their superiorities in the path length and robustness.

Table 6.3: Comparison of proposed ABC algorithms and four path planners for solving SRPP problems

	Path planners	Avg. path length	Best path	Worst path	Avg. time
Map 1	A*	626.105	626.105	626.105	40.248
	PRM	609.822	**597.298**	634.480	**4.194**
	RRT	675.507	631.113	736.114	6.997
	BRRT	731.488	667.771	784.292	6.927
	ABCL	602.702	601.480	606.379	10.859
	ABCDC	621.947	608.128	635.182	15.061
	ABC_RL	603.408	601.794	605.932	10.239
	FOABC	**601.218**	601.206	**601.314**	10.286
Map 2	A*	620.541	620.541	620.541	127.001
	PRM	649.105	608.643	706.050	**4.736**
	RRT	763.858	681.373	907.245	22.070
	BRRT	776.651	712.855	864.513	8.487
	ABCL	599.871	595.468	606.456	10.066
	ABCDC	618.030	607.615	634.652	8.929
	ABC_RL	603.119	597.063	614.470	10.772
	FOABC	**595.344**	**595.021**	**596.280**	11.850
Map 3	A*	580.583	580.583	580.583	64.371
	PRM	650.791	567.378	825.051	**4.233**
	RRT	686.250	603.656	798.088	6.342
	BRRT	714.202	631.658	821.873	7.188
	ABCL	549.955	549.296	551.434	9.493
	ABCDC	560.967	550.466	575.777	12.784
	ABC_RL	550.917	549.060	558.013	12.112
	FOABC	**549.283**	**548.943**	**550.038**	10.855
Map 4	A*	**515.612**	515.612	**515.612**	70.342
	PRM	570.306	**510.282**	634.658	**5.069**
	RRT	672.524	583.872	879.423	7.940
	BRRT	737.793	605.132	1015.296	7.743
	ABCL	537.731	525.540	555.251	10.342
	ABCDC	554.008	540.861	569.026	12.316
	ABC_RL	538.458	525.358	557.892	12.909
	FOABC	525.535	524.716	532.401	11.101
Map 5	A*	648.950	648.950	648.950	67.680
	PRM	689.412	**612.760**	814.643	**5.743**
	RRT	736.144	660.395	920.081	8.241
	BRRT	800.318	710.789	920.934	9.246
	ABCL	620.701	619.175	626.336	11.186
	ABCDC	648.908	632.713	666.528	15.344
	ABC_RL	625.622	619.162	762.696	13.634
	FOABC	**619.142**	619.013	**619.710**	12.364
Map 6	A*	563.279	563.279	563.279	66.804
	PRM	639.337	568.421	752.795	**5.151**
	RRT	766.370	612.092	1172.381	17.718
	BRRT	770.834	613.458	1130.065	10.796
	ABCL	539.069	538.572	539.707	9.230
	ABCDC	594.741	575.052	630.126	12.512
	ABC_RL	539.195	538.552	540.969	11.913
	FOABC	**538.723**	**538.459**	**539.672**	10.954



(a) Map 1 **(b)** Map 2

(c) Map 3 **(d)** Map 4

(e) Map 5 **(f)** Map 6

Fig. 6.7. The best paths of FOABC algorithm and four well-known path planners in six workspaces

6.3 Multi-robot path planning (MRPP)

Compared to a single robot, a multi-robot system possesses advantages because of the cooperation and interaction inside the team. In this case, multi-robot collaboration has a stronger ability to resolve complex problems and has higher robustness and reliability (Das & Jena, 2020). In this context, multi-robot systems exist in various fields, for instance, intelligent warehouse (Han & Yu, 2019), oilfield inspection (Li *et al.*, 2020a), etc. Hence, it is worth pointing out that multi-robot path planning (MRPP) is a critical technique.

As a result of the aforementioned advantages and broad applications, MRPP problems have been investigated by researchers for decades. The objective of MRPP is to compute collision-free and qualifying paths for a group of robots from an initial configuration to a target configuration via path planners (Han & Yu, 2020). As we have mentioned, the classical and graph-based methods cannot remain effective when dealing with complex situations. These approaches become more limited when targeting MRPP tasks. Correspondingly, the type of algorithm we study is more flexible and unconstrained by the problem. Therefore, as long as an optimization problem can be properly formulated, feasible paths can be found using such meta-heuristic methods.

In this part, the objective is to solve on-line MRPP problems more effectively via the intelligent meta-heuristic algorithm. The task is to generate optimal paths without collisions for a group of robots from their starting positions to the intended targets while taking environmental constraints into account. An objective function is properly defined considering the length and safety of the paths. To solve this established problem, the ABCL algorithm is adopted considering its simple structure and the fact that it consumes the least amount of time in the previous simulations. In fact, ABCL is designed with the ultimate goal of resolving the MRPP problem. Hence, as indicated in Chapter 4, the proposed improvement strategies shouldn't make computations more complex because we need to consider the time spent for computing during on-line planning.

Moreover, a new implementation method is adopted to determine the next positions for all the robots simultaneously. In other words, the candidate solutions of ABCL are formed with the angles to rotate and speeds of all the moving robots. Then the optimal subsequent coordinates are computed by ABCL considering the

path length, distances from the obstacles, and distances between teammates. In this way, the optimal collision-free paths for multiple robots can be obtained.

6.3.1 Problem formulation of MRPP

Similar to the previous case, the environment where the robots move and act is composed of obstacles and free space. The start points and target points of robots are located in the free space. An example of MRPP's environment is shown in Figure 6.8, where O_1, O_2, O_3 are the static obstacles. S_{ri} and T_{ri} are the start and target points of the i^{th} robot ri $(i = 1, 2, 3)$. Note that, for clarity, the start positions and goals of different robots are colored differently.

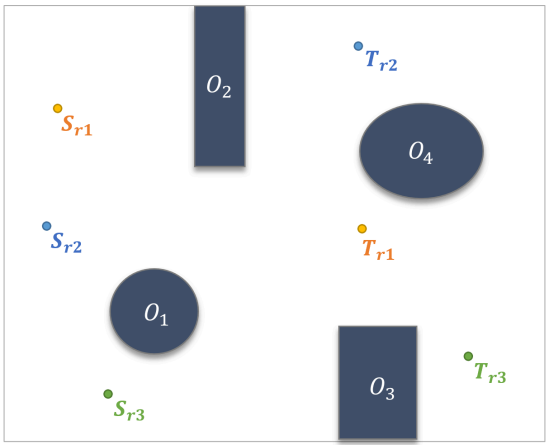


Fig. 6.8. Environment of multi-robot path planning

The objective of MRPP is to find collision-free paths for a group of mobile robots from their start points to their desired goals. To this end, collisions between team members should also be prevented while avoiding the obstacles. Note that, depending on the requirements of different missions, the robots can also reach the same destination. In almost all the path planning problems, there are three aspects that need to be mainly concerned with: efficiency, safety, and accuracy. That is, each robot should save as much energy as possible while finding a safe path in a short amount of time (Koubâa *et al.*, 2018).

In the followings, we focus on the MRPP problem in 2-dimensional environments filled with static obstacles. And several assumptions are given before formulating the problem. Firstly, each robot knows its initial and target positions. Secondly, at each moment, each robot selects its own direction and speed for the

next movement. This process continues until the robots achieve their targets or a collision occurs. Moreover, to get closer to reality, the robots are considered as squares rather than points.

With the purpose of formulating the MRPP problem, it is necessary to define a proper objective function taking into account the three aspects mentioned above. In this context, the function should be utilized to minimize each robot's path length and arrival time while avoiding obstacles (Das & Jena, 2020). In each step, the proposed algorithm is used to compute the next positions from the current positions of the robots. So, the paths can be found by constantly determining the subsequent feasible positions until they reach their corresponding targets. Notice that, when part of the robots arrive at their destinations, they will stay there and wait for the others to move toward their goals. Besides, the proposed implementation method will be explained after introducing the robot kinematic model.

6.3.1.1 Robot kinematic model

The kinematic model of each robot is demonstrated in Figure 6.9. For the robot ri, suppose its position at time t is $(x^{present}, y^{present})$ and $\theta^{present}$ is the angle between its forward direction and the $x-$axis.

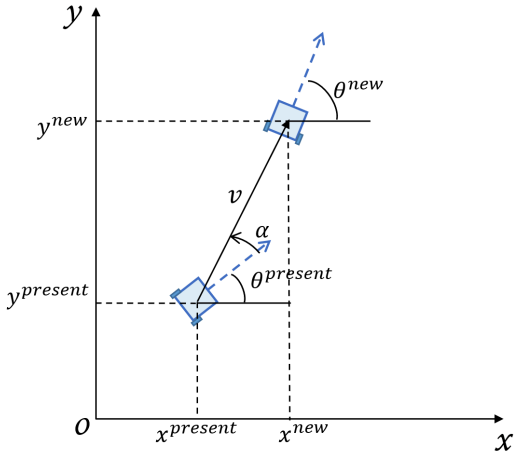


Fig. 6.9. Kinematic illustration of a mobile robot from present to its new position

The variables α and v indicate the rotation angle and robot velocity, respectively, which need to be optimized for the next displacement. In this case, the

next position (x^{new}, y^{new}) that robot will be in at time $(t + \Delta t)$ can be expressed as follows.

$$\begin{cases} x_{ri}^{new} = x_{ri}^{present} + \cos(\alpha_{ri} + \theta_{ri}^{present}) \cdot v_{ri} \cdot \Delta t \\ y_{ri}^{new} = y_{ri}^{present} + \sin(\alpha_{ri} + \theta_{ri}^{present}) \cdot v_{ri} \cdot \Delta t \end{cases}. \tag{6.5}$$

Set $\Delta t = 1$ and $\theta_{ri}^{new} = \alpha_{ri} + \theta_{ri}^{present}$, then the equation can be rewritten as

$$\begin{cases} x_{ri}^{new} = x_{ri}^{present} + \cos\theta_{ri}^{new} \cdot v_{ri} \\ y_{ri}^{new} = y_{ri}^{present} + \sin\theta_{ri}^{new} \cdot v_{ri} \end{cases}. \tag{6.6}$$

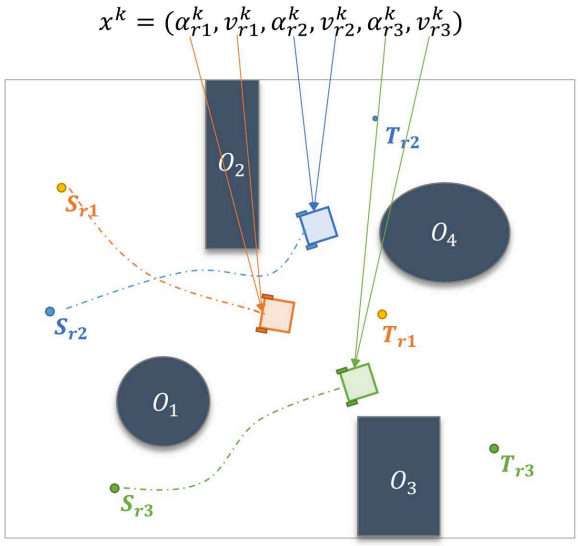
6.3.1.2 Implementation method of MRPP

There exist various manners of employing this kind of algorithm to achieve robot navigation. One common implementation approach is to generate a group of feasible paths as initial solutions, then optimize the paths with the optimization algorithms (Agarwal & Bharti, 2021; Nazarahari *et al.*, 2019; Oleiwi *et al.*, 2014). Besides, some classical path planning methods are used to generate the initial paths before adopting the heuristic algorithms. In the case of global path planning, a complete path from the robot's start point to its target is determined before the robot starts to move. So the environment should be completely known (Sedighi *et al.*, 2004). On the other hand, local path planning methods are able to navigate the robots in dynamic or incomplete workspaces. And the path is generated while the robot is moving towards the destination (Koubâa *et al.*, 2018). Accordingly, some improved meta-heuristic algorithms have been used for local path planning. For instance, Das *et al.* (2016a) proposed a hybrid IPSO–IGSA algorithm for MRPP. The algorithm was utilized to determine the next positions from the current positions of every robot. And for each robot, the algorithm was called after each move by taking its current position and velocity as parameters.

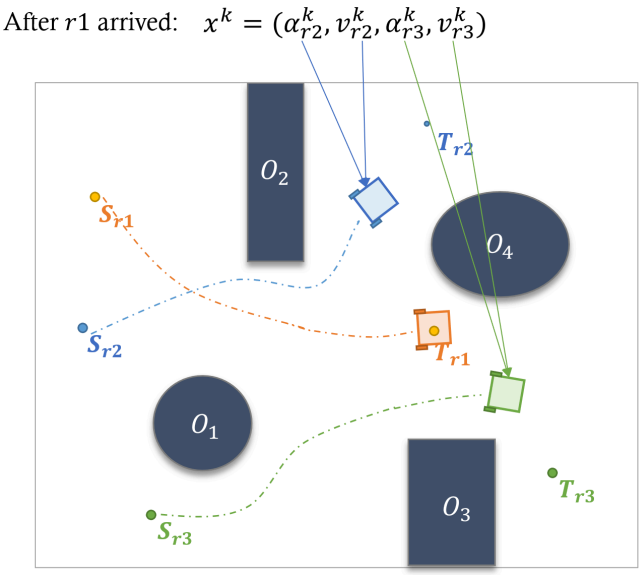
Different from the existing implementation methods, a novel way that can determine the next positions of all the robots simultaneously is designed. In other words, at each step, the algorithm is called only once to find where all the robots are moving to. In this way, the running time can be saved.

The form of candidate solutions is made up of the rotation angles and velocities of all the robots that have not reached their targets yet. The k^{th} solution is formulated as $x^k = (\alpha_{r1}^k, v_{r1}^k, \alpha_{r2}^k, v_{r2}^k, \cdots, \alpha_{rN}^k, v_{rN}^k)$, N is the number of robots

and $k \in \{1, \cdots, SN\}$. Reminding that SN stands for the swarm population, and here it also represents the number of candidate paths. The manner of determining the subsequent positions of all the robots simultaneously is illustrated in Figure 6.10.



(a) All the robots are moving



(b) After part of robots arrived their targets

Fig. 6.10. Illustrations of determining subsequent positions for multiple robots at the same time

The dimension of the candidate solution is set as $D = 2 \times N$ where N is the number of robots. When part of the robots achieve their goals, the dimension of candidate solutions will be decreased accordingly, as shown in Figure 6.10(b). In each generation, all candidate solutions (all planning options for robots' next positions) will be evaluated by objective function F which will be described in the following. A candidate planning with a smaller function value contains more qualified subsequent positions.

By using this implementation method, the efficiency of path searching can be ensured, especially when the number of robots is large. If the robot number increases, only the dimension of candidate solutions will augment when using the proposed implementation method. However, in the case that robots call the algorithm to compute the next positions one by one, the number of times the algorithm is called will increase correspondingly, which will significantly increase the running time. Thus, the novel implementation approach can not only save time but also maximize the benefits of meta-heuristic algorithms when dealing with high-dimensional optimization problems. In addition, the outline of the proposed MRPP implementation method is shown in Algorithm 15.

Algorithm 15 Pseudo-code of the proposed MRPP method

Input Map; Number of robots N; Start points S_{ri} and Target points T_{ri} $(i = 1, \cdots, N)$; Robot size; Search ranges of rotate angles α and velocities v
Initialize parameters and set list of current positions $L\_current = [S_{r1}; S_{r2}; \cdots; S_{rN}]$
List of targets of moving robots $L\_target = [T_{r1}; T_{r2}; \cdots; T_{rN}]$
repeat
    Get the L_target, $L\_current$ of robots haven't arrive their targets
    Calculate the optimal α_{ri} and velocities v_{ri} via the proposed ABCL algorithm
    Move to the next positions and update $L\_current$
    for each moving robot **do**
        if robot ri arrives its target T_{ri} **then**
            Remove position of ri from $L\_current$
            Remove T_{ri} from L_target
        end if
    end for
until all the robots reach the desired targets

6.3.1.3 Objective function of MRPP

The quality of paths is measured by the objective function. In order to find the optimal and safe paths for multiple robots, the objective function is defined in three parts. Function f_1 is used to lead all the robots toward their destinations. And f_2 is designed to prevent collisions between teammates, while f_3 is adopted to avoid the surrounding obstacles.

First of all, the function f_1 is defined to minimize the distances between the N robots and their desired targets.

$$f_1 = \sum_{i=1}^{N} \sqrt{(x_{ri}^{new} - x_{ri}^{T})^2 + (y_{ri}^{new} - y_{ri}^{T})^2}, \tag{6.7}$$

where $(x_{ri}^{new}, y_{ri}^{new})$ with $i = 1, \cdots, N$ represent the next positions of all the robots generated by the algorithm, and (x_{ri}^{T}, y_{ri}^{T}) are the corresponding target points.

Then, the constraints imposed by the environment are tackled via the following objective functions. The second objective function f_2 is formulated in order to avoid collisions between teammates as shown in Eq.(6.8).

$$f_2 = \sum_{i=1}^{N} g_2^{ri}, \tag{6.8}$$

where

$$g_2^{ri} = \begin{cases} 0, & if \quad Dis(ri, rk) > Q, \\ \eta \times (\dfrac{1}{Dis(ri, rk)} - \dfrac{1}{Q}), & otherwise, \end{cases} \tag{6.9}$$

where η is a positive constant and $Dis(ri, rk)$ is the distance between the i^{th} robot and the k^{th} robot, $k = 1, \cdots, N$. For each robot, the distances to all the other robots are calculated with Eq.(6.9), then the security information is concerned together in Eq.(6.8). Q is the safe distance defined as $Q = 2 \times L\_diagonal$, with $L\_diagonal$ is robot's diagonal length.

The f_3 is defined to avoid obstacles in the environment, which is formulated in Eq.(6.11). The nearest obstacle is searched step by step in five specific directions, namely left, forward left diagonal, forward, forward right diagonal, and right as expressed in Eq.(6.10). In order to explain clearer, the way of measuring distances of obstacles is demonstrated in Figure 6.11.

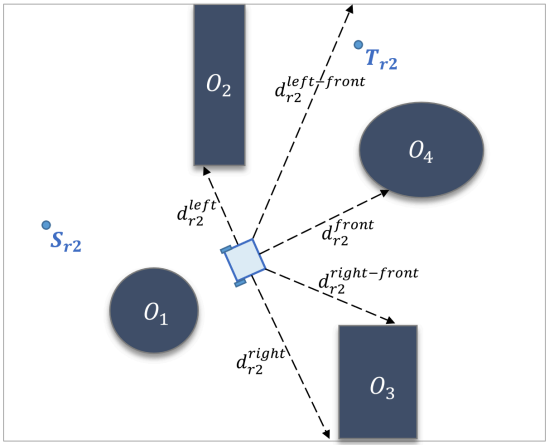


Fig. 6.11. Illustration of measuring distances from obstacles in 5 directions

$$minObs^{ri} = min(d_{ri}^{left}, d_{ri}^{left-front}, d_{ri}^{front}, d_{ri}^{right-front}, d_{ri}^{right}), \tag{6.10}$$

$$f_3 = \sum_{i=1}^{N} g_3^{ri}. \tag{6.11}$$

$$g_3^{ri} = \begin{cases} 0, & if \quad minObs^{ri} > Q, \\ \zeta \times (\dfrac{1}{minObs^{ri}} - \dfrac{1}{Q}), & otherwise, \end{cases} \tag{6.12}$$

where ζ is a positive constant and Q is the safe distance defined above. $minObs^{ri}$ is the minimum distance from robot ri to the obstacles calculated with Eq.(6.10). Note that the safety of all robots is considered simultaneously.

Hence, the MRPP problem can be represented as an optimization problem with an objective function that minimizes the sum of f_1, f_2, and f_3. The function f_1 is defined to attract the robots toward their targets. f_2 and f_3 are designed for avoiding collisions with obstacles and other robots, respectively. Therefore, the overall objective functions can be defined by the sum of these functions which are equally weighted as shown in Eq.(6.13)

$$F = f_1 + f_2 + f_3. \tag{6.13}$$

Then, each time of planning the next positions, the objective function is minimized by the proposed algorithm.

6.3.2 Simulation results of MRPP

Six maps of size 500×500 pixels with different complexity are designed to verify the effectiveness of the ABCL algorithm in MRPP. In the simulations, mobile robots are represented by squares with a side length of 10 pixels and are colored differently. The start points and target points of all the robots are assigned beforehand. In addition, the obstacles are black and in various shapes. The complexity of a workspace is related to the size and number of obstacles.

The performance of ABCL is examined by comparing it with three other ABC algorithms in cases of six and twelve robots. More precisely, the standard ABC, GABC (Zhu & Kwong, 2010), and DABC (Abbas & Ali, 2014) are compared with ABCL. Note that DABC was designed for mobile RPP as well. The parameter settings are the same as those in the previous comparisons.

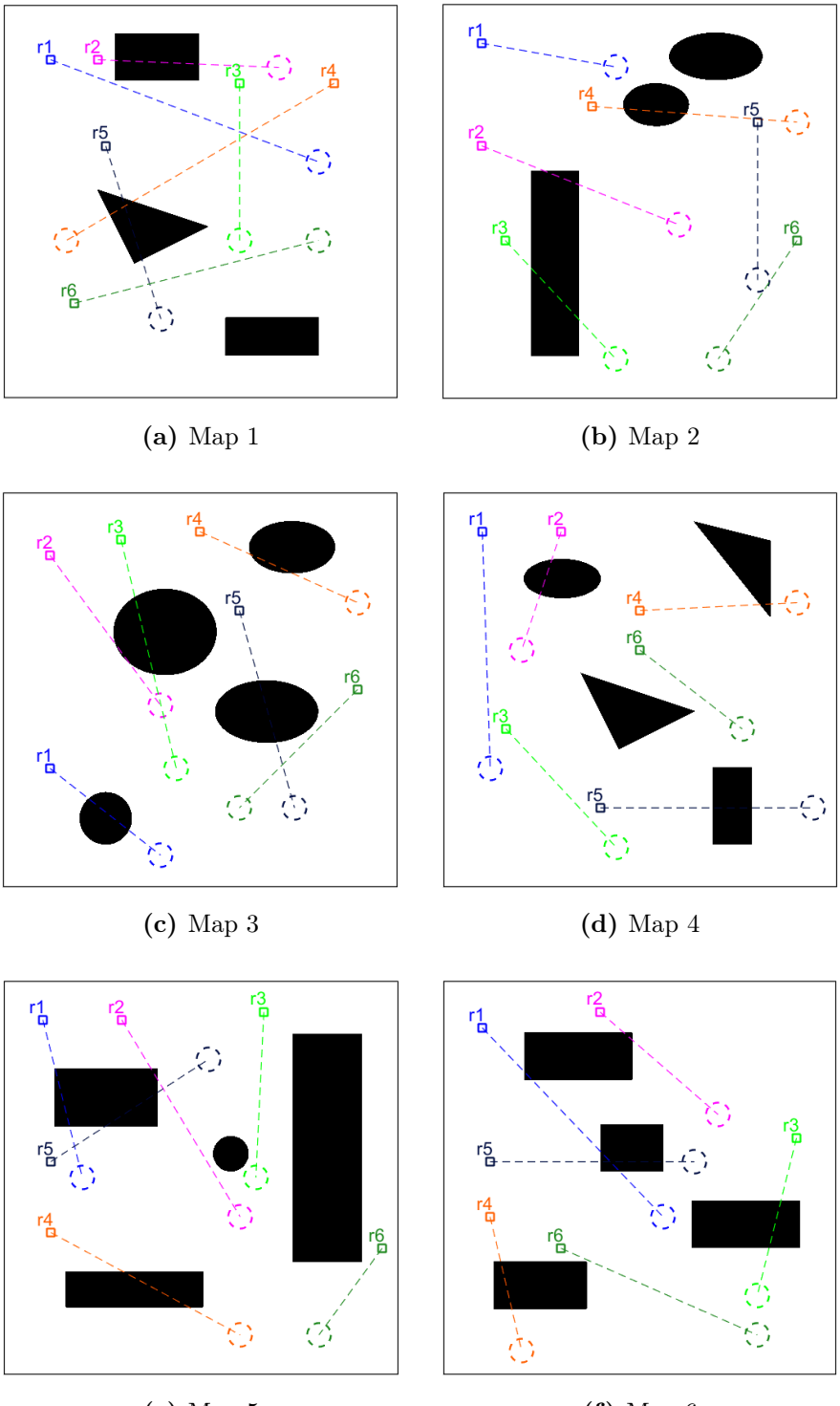
For all the algorithms, the swarm number is $SN = 2 \times nb\_robot$ and the determination condition is set as $max\_FES = 3000 \times D$. And the range of rotate angle α is defined as $[-\pi/2, \pi/2]$ while the robot velocity $v \in [10, 60]$. Notice that D is equal to $2 \times nb\_robot$ according to the proposed implementation manner.

6.3.2.1 Comparative study of six-robot path planning

In this part, the simulations of six robot path planning are implemented. The initial configuration of maps is shown in Figure 6.12. When the number of robots is six, in each workspace, there are six squares color-coded differently, which are the initial positions of the robots. The other six circles drawn with dotted lines represent the target areas. The areas are centered on the predefined goal positions and the path planning is achieved once the robots enter the associated areas. Each robot has a specific color which is the color of its corresponding start point and ending area.

For each map, each compared algorithm is executed 15 independent times and the average values are calculated. In Table 6.4, the average number of steps required to reach the target and the average path length traveled by each robot are presented. The minimum values are marked in **boldface**. Moreover, the average running time is concerned in order to make a more meaningful comparison. The consumed time is presented in Figure 6.14.

In Table 6.4, all the algorithms manage to plan a set of collision-free paths for the six robots. Nonetheless, the effectiveness of ABCL can be proved according



(a) Map 1 **(b)** Map 2

(c) Map 3 **(d)** Map 4

(e) Map 5 **(f)** Map 6

Fig. 6.12. The initial configurations of MRPP workspaces for six robots

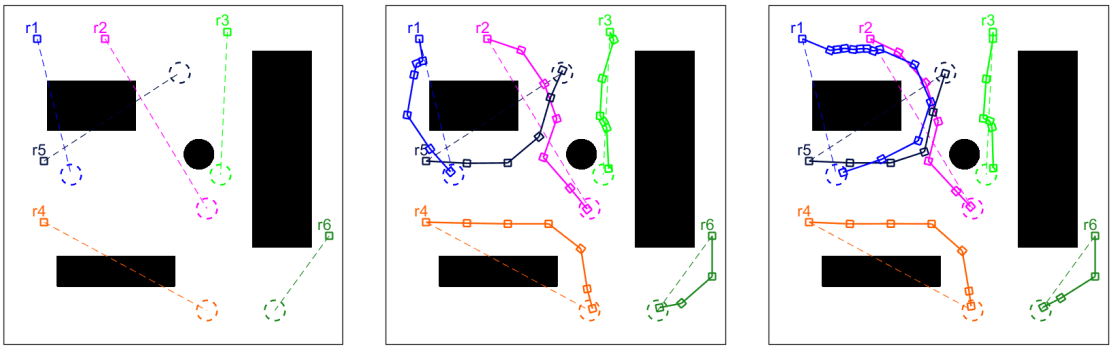
Table 6.4: Comparison of average required steps and average path lengths for six robots

		Average required steps				Average path length			
	Robot	ABC	GABC	DABC	ABCL	ABC	GABC	DABC	ABCL
Map 1	r1	6.73	6.93	6.87	6.67	464.57	468.06	467.51	461.90
	r2	5.00	5.00	5.00	5.00	240.18	239.33	240.20	238.61
	r3	4.00	4.00	4.00	4.00	203.40	205.08	201.68	202.66
	r4	8.00	8.00	8.00	8.00	363.98	363.70	361.82	362.16
	r5	6.20	5.53	6.20	6.07	292.98	286.49	288.53	287.93
	r6	6.07	6.00	6.00	6.00	228.81	229.66	229.58	228.65
	Total	36.00	**35.47**	36.07	35.73	1793.91	1792.32	1789.31	**1781.92**
Map 2	r1	3.00	3.00	3.00	3.00	185.32	187.31	186.51	185.57
	r2	5.20	5.00	5.00	5.00	338.10	346.47	343.84	347.17
	r3	8.33	12.87	7.60	8.00	286.56	365.90	283.34	285.07
	r4	7.20	6.93	8.07	5.00	293.82	292.56	309.29	277.27
	r5	4.00	4.00	4.00	4.00	199.07	201.46	199.90	199.72
	r6	3.00	3.13	3.20	3.00	69.15	77.11	65.34	69.31
	Total	30.73	34.93	30.87	**28.00**	1372.02	1470.82	1388.22	**1364.11**
Map 3	r1	4.53	5.00	4.20	4.53	243.48	254.73	242.15	242.77
	r2	5.00	5.00	5.00	5.00	335.59	336.74	337.04	336.46
	r3	12.40	10.67	12.53	9.93	416.57	413.74	449.06	396.29
	r4	4.67	4.27	4.60	4.40	283.42	282.21	285.77	279.95
	r5	6.53	6.60	6.67	6.67	319.28	318.54	318.96	318.85
	r6	4.00	5.20	4.00	4.00	117.64	143.27	115.26	116.77
	Total	37.13	36.73	37.00	**34.53**	1715.98	1749.23	1748.25	**1691.08**
Map 4	r1	5.00	5.00	5.07	5.00	306.84	307.88	305.76	305.80
	r2	7.00	6.93	7.00	6.73	214.65	212.69	218.25	215.38
	r3	4.00	4.00	4.00	4.00	287.76	287.27	286.36	286.94
	r4	4.07	4.07	4.20	4.00	198.51	194.42	203.83	193.94
	r5	11.20	10.67	9.87	9.73	325.53	327.65	316.88	300.90
	r6	3.00	3.00	3.00	3.00	223.99	225.86	225.30	223.31
	Total	34.27	33.67	33.13	**32.47**	1557.28	1555.75	1556.38	**1526.27**
Map 5	r1	8.67	6.60	8.87	9.40	308.948	263.986	308.93	307.224
	r2	7.53	7.40	6.67	6.93	398.236	408.086	400.047	399.518
	r3	6.27	6.60	6.00	6.40	223.052	215.56	211.214	209.014
	r4	6.00	6.00	6.00	6.07	367.657	368.641	367.339	368.188
	r5	5.07	5.13	5.13	5.20	208.825	202.747	217.093	209.991
	r6	3.00	9.47	3.00	3.00	89.4593	215.795	87.761	84.1803
	Total	36.53	41.20	**35.67**	37.00	1596.18	1674.81	1592.38	**1578.11**
Map 6	r1	6.67	7.20	6.33	7.27	468.25	478.65	460.41	471.77
	r2	4.00	4.00	4.00	4.00	281.43	278.02	279.16	279.92
	r3	12.87	16.93	13.67	13.13	340.01	386.93	358.04	320.07
	r4	10.60	11.60	12.80	8.93	316.46	315.82	357.54	265.03
	r5	6.13	7.93	5.33	5.93	281.86	302.74	271.76	272.44
	r6	5.00	5.00	5.00	5.00	356.11	355.46	356.46	356.30
	Total	45.27	52.67	47.13	**44.27**	2044.13	2117.62	2083.37	**1965.52**

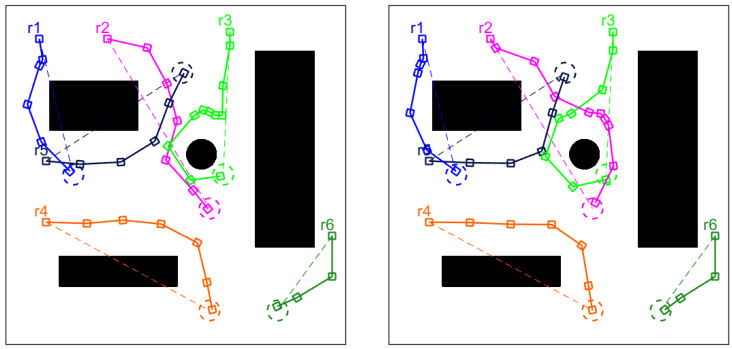
to the comparison results. The proposed algorithm outperforms the other competitors in terms of the average total path length in all the maps. Moreover, the steps required to reach the destinations when utilizing ABCL are also competitive. Actually, because the robot's speed is variable, having more steps does not necessarily result in a longer journey. So, in Map 1 and Map 5, certain robots move very short in some places with ABCL. Hence, in these two cases, the path length planned by ABCL is the shortest but more steps are used. And in Map 1, the GABC algorithm needs the least number of steps to get to the target points while DABC surpasses the others in Map 5. Nevertheless, ABCL can still keep a nice balance between the number of steps the robots require and the total length of paths.

Furthermore, another difficulty in MRPP is how to choose the direction when there exist obstacles in a robot's moving direction, especially in a narrow area. A longer detour may occur if the path planner fails to determine an optimal direction. Examples are given in Figure 6.13 to demonstrate clearly the detours that may take place.

In Figure 6.13(a), the initial configuration of Map 5 is presented. It can be found that in order to reach the desired target points, all the robots will move very close to the obstacles. Particularly, for robots $r1$ and $r3$, the path lengths of bypassing the two sides are significantly different when encountering obstacles. A case where all the robots avoid obstacles successfully and do not take detours, as shown in (b). The (c)-(e) in Figure 6.13 demonstrate different detour cases. For robot $r1$, it is better to pass the obstacle from the left side of the map. But as shown in Figure 6.13(c), it moves through the other side from the start so that it has to travel much longer. It should be pointed out that in this case, the routes of $r1$, $r2$, and $r5$ are close to each other, but there is no collision because the robots take different steps to arrive there. In other words, the robots pass by at separate moments.



(a) Map 5 Initial state **(b)** Case without obvious detour **(c)** Detour case 1



(d) Detour case 2 **(e)** Detour case 3

Fig. 6.13. The examples of detours in Map 5 planned by ABC algorithm

In Figure 6.13(d), $r3$ takes a detour when it encounters the circular obstacle in the center of the workspace. Besides, both $r2$ and $r3$ travel a bit longer than they do in (b). In fact, if an algorithm is able to distribute the candidate solutions more comprehensively and extract the promising information throughout iterations, it is more capable of choosing the right directions for the robots. In this context, the proposed algorithm is more effective to plan reasonable paths for robots, especially in complicated cases.

In addition, the running time is an essential metric in path planning problems. Hence, the average consumed time of each algorithm over 15 executions is calculated and shown in Figure 6.14.

In Figure 6.14, it can be observed that the ABCL algorithm takes the shortest time in all designed workspaces. As mentioned before, the complexities of Maps 5 and 6 are higher than the other maps. So, in simpler cases, the execution times

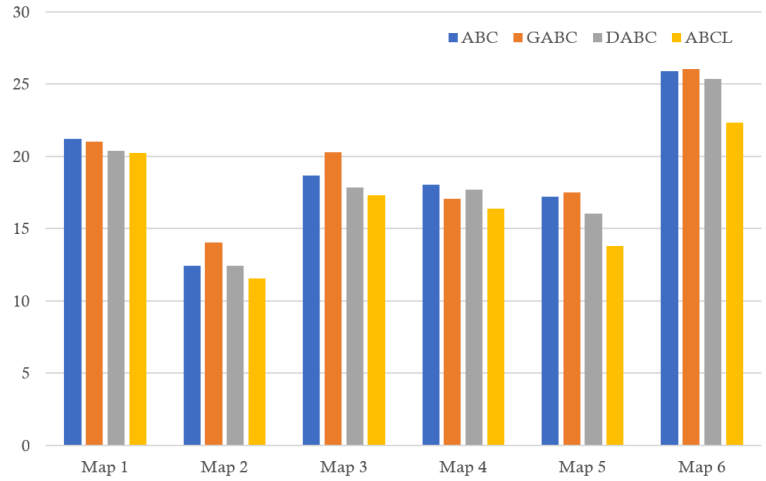


Fig. 6.14. Comparison of running time in six robots path planning problems

of all the algorithms are close. And the advantages of ABCL are more significant in the last two cases in terms of time efficiency.

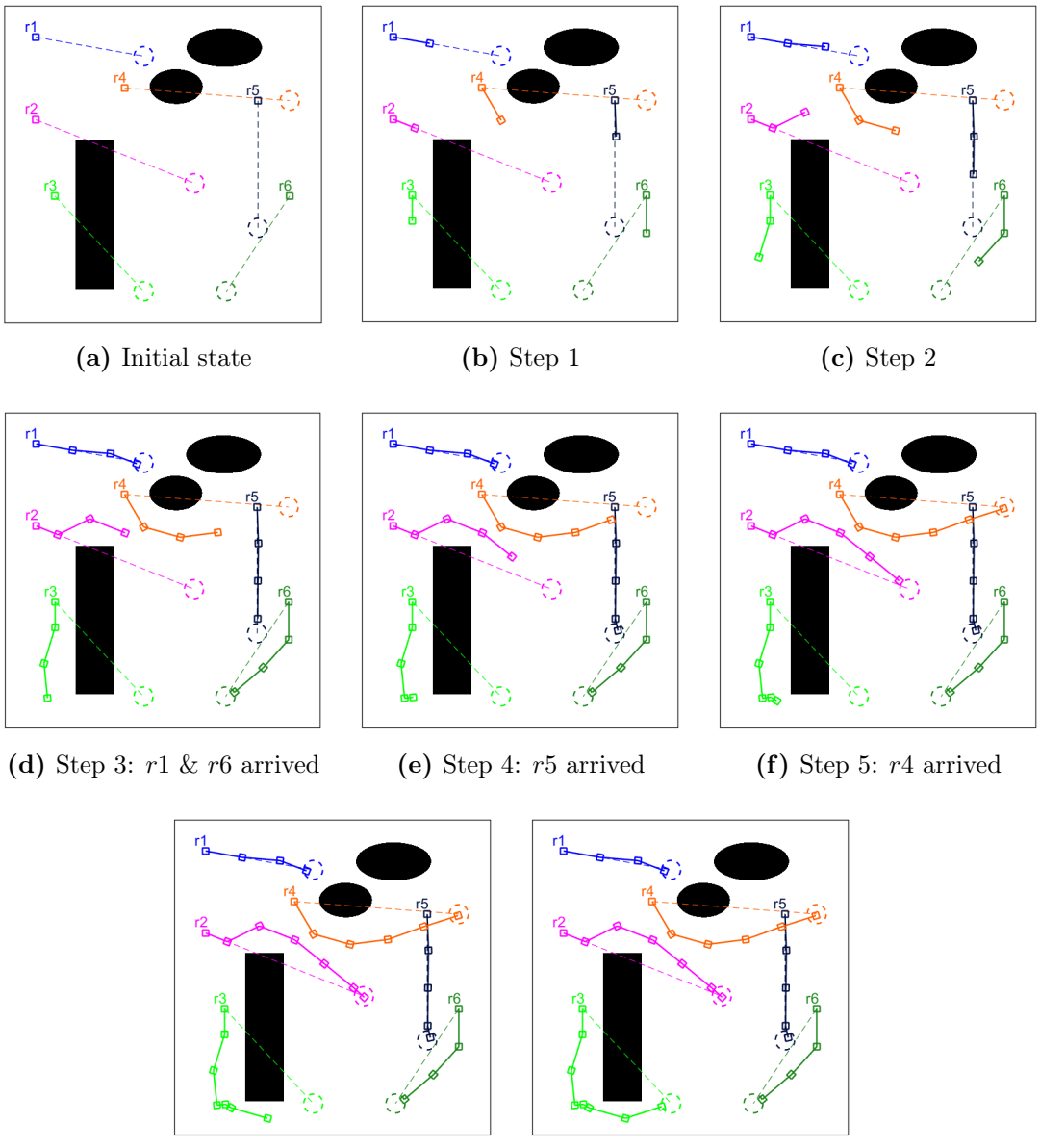
6.3.2.2 MRPP process of six robots

To display the process of MRPP clearly, examples of using ABCL in two maps are shown in Figure 6.15 and Figure 6.16. Note that, each subfigure corresponds to the path planning situation at each moment.

In Figure 6.15, the robots $r1$ and $r6$ reach their target points in three steps since their planning tasks are easier than the others. The robots that have already arrived will stay there and wait for the other teammates. Then, $r5$ arrives at its goal in the next step. $r4$ and $r2$ achieve their desired destinations and successfully avoid obstacles after the next two movements. Finally, the whole MRPP is accomplished in seven steps. As for Map 4, a similar process can be found in Figure 6.16. All robots are able to avoid obstacles efficiently. Meanwhile, when there are no teammates or obstacles nearby, the robots can move in almost the shortest straight line toward the destinations, $r1$ and $r3$ as examples.

6.3.2.3 Comparative study of twelve-robot path planning

More challenging tasks are designed to further test the ABCL's competence in solving MRPP and the efficacy of the proposed implementation manner. Hence, the number of robots is doubled (i.e., twelve robots), and the initial configurations

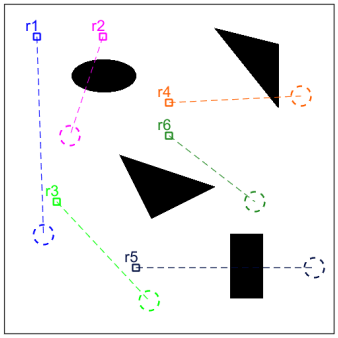


(a) Initial state **(b)** Step 1 **(c)** Step 2

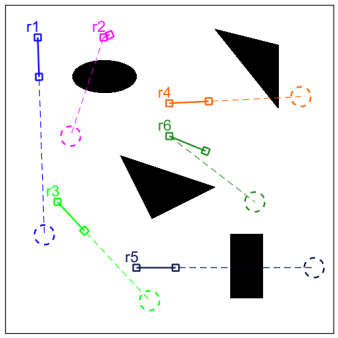
(d) Step 3: $r1$ & $r6$ arrived **(e)** Step 4: $r5$ arrived **(f)** Step 5: $r4$ arrived

(g) Step 6: $r2$ arrived **(h)** Step 7: MRPP accomplished

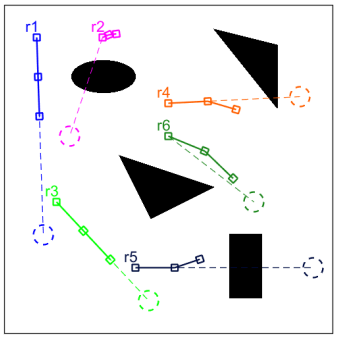
Fig. 6.15. MRPP process for six robots via ABCL algorithm in Map 2



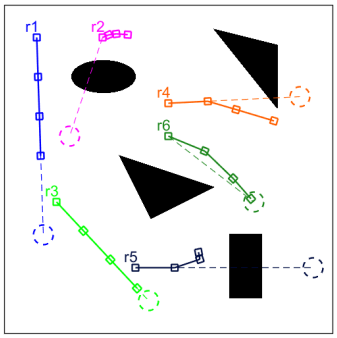
(a) Initial state



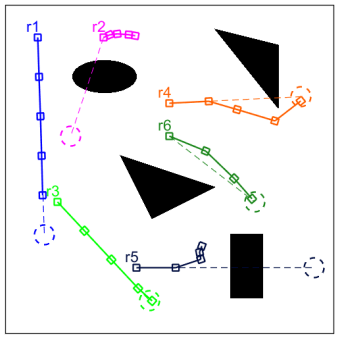
(b) Step 1



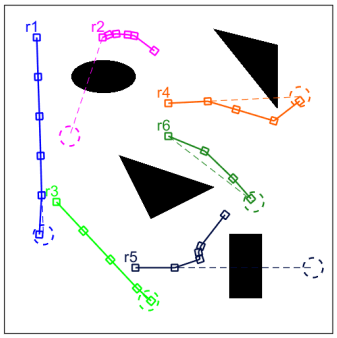
(c) Step 2



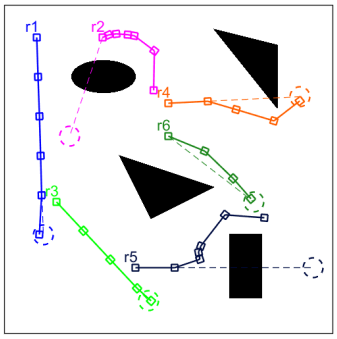
(d) Step 3: $r6$ arrived



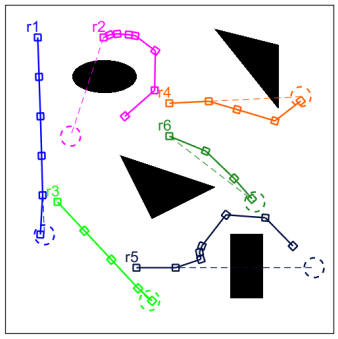
(e) Step 4: $r3$ & $r4$ arrived



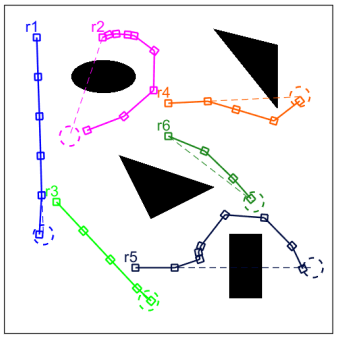
(f) Step 5: $r1$ arrived



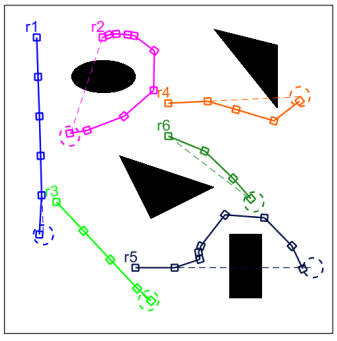
(g) Step 6



(h) Step 7



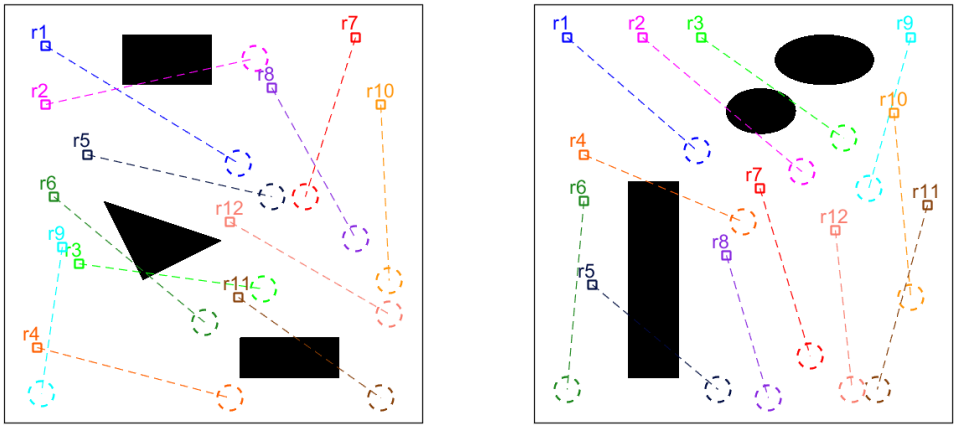
(i) Step 8: $r5$ arrived



(j) Step 9: MRPP accomplished

Fig. 6.16. MRPP process for six robots via ABCL algorithm in Map 4

of two tested maps are given in Figure 6.17. The settings of the termination condition and variable search ranges are defined the same as before. Notice that Map 1 and Map 2 are identical with the previous simulations, however, the problem became harder as the workspace became more crowded.



(a) Map 1 **(b)** Map 2

Fig. 6.17. The initial configurations of MRPP workspaces for twelve robots

The average required steps, average path lengths, and running time are compared with ABC, GABC, and DABC over 15 independent runs as shown in Table 6.5.

Based on the comparison in Table 6.5, the MRPP missions of twelve robots are accomplished by all the involved algorithms without collisions. And ABCL evidently outperforms the other methods in terms of average path lengths as well as execution time. Meanwhile, the steps required to reach the predefined goals via GABC are a bit smaller than the other competitors. It can be found that the proposed ABCL algorithm enables the robots to avoid long detours by taking more small steps in some complex situations. In this way, the average path lengths planned by ABCL are the shortest in comparison while it requires a bit more steps to achieve the targets compared with GABC. Besides, the average running time can be found in the last row of each case in the table. Although the execution times of all the algorithms do not have a large difference, ABCL is still the most efficient. Therefore, its effectiveness can be further proved through these complex tasks MRPP.

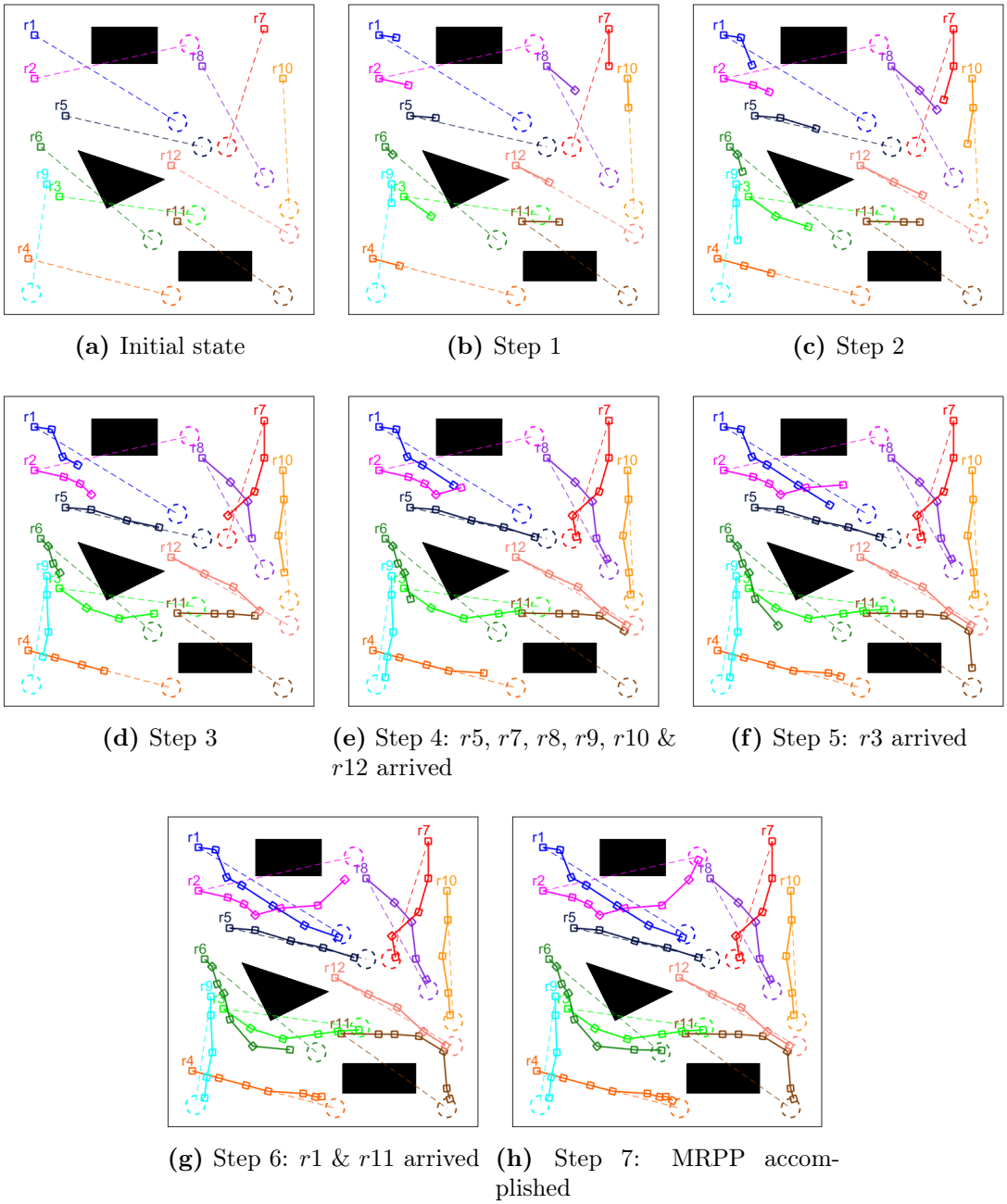
Table 6.5: Comparison of average required steps, average path lengths and running time for twelve robots

		Average required steps				Average path length			
	Robot	ABC	GABC	DABC	ABCL	ABC	GABC	DABC	ABCL
Map 1	r1	6.73	5.40	6.07	6.33	367.20	368.08	364.37	368.60
	r2	7.33	6.07	6.27	6.80	279.54	253.97	258.85	257.79
	r3	5.40	5.20	5.60	5.73	244.82	249.06	249.16	248.31
	r4	6.80	6.87	6.87	6.33	279.67	279.11	277.13	270.77
	r5	4.67	4.07	4.40	5.13	261.44	265.09	261.88	270.40
	r6	7.13	5.67	6.80	6.60	334.41	326.76	329.10	328.00
	r7	4.87	4.07	4.67	4.60	154.24	142.09	152.68	139.62
	r8	5.20	4.07	4.80	4.40	278.54	274.67	281.49	267.90
	r9	4.47	3.87	4.27	4.40	157.61	152.90	152.63	155.05
	r10	5.33	4.33	5.40	5.13	213.15	216.24	221.94	216.98
	r11	6.27	4.93	5.80	5.73	278.62	274.98	275.79	276.54
	r12	5.00	4.60	4.60	4.87	288.74	289.69	287.10	287.09
	Total	69.20	**59.13**	65.53	66.07	3137.99	3092.66	3112.11	**3087.05**
		Average	running	time		47.24	58.91	49.33	**43.35**
Map 2	r1	5.20	4.53	4.87	5.00	292.31	283.73	284.43	280.90
	r2	8.47	6.53	8.00	7.93	363.06	353.66	369.01	354.01
	r3	13.53	14.47	10.80	12.27	341.81	351.09	336.03	330.28
	r4	9.33	6.67	9.00	5.27	334.70	298.18	334.43	260.97
	r5	12.00	12.73	10.00	10.13	348.23	331.10	319.22	320.92
	r6	6.27	4.73	6.00	6.07	227.25	207.85	222.06	216.36
	r7	4.40	4.00	4.40	4.53	252.04	254.15	255.61	251.67
	r8	5.87	4.00	5.47	5.47	228.89	209.06	233.64	227.60
	r9	4.33	3.80	4.40	4.47	137.03	134.03	131.65	137.00
	r10	4.93	4.20	4.87	5.47	234.02	234.55	233.24	240.54
	r11	5.27	4.67	5.47	5.47	181.16	172.14	168.68	169.73
	r12	5.20	3.73	5.07	4.53	210.81	197.01	216.47	205.58
	Total	84.80	**74.07**	78.33	76.60	3151.31	3026.55	3104.46	**2995.57**
		Average	running	time		41.84	58.03	45.81	**39.20**

6.3.2.4 MRPP process of twelve robots

The process of planning paths for twelve robots simultaneously is demonstrated step by step in Figure 6.18. For robots $r4$, $r5$, $r7$, $r8$, $r10$, and $r12$, connecting lines from their respective start points to the end points do not cross any static obstacles, which means that the planned paths are preferred if they are closer to straight lines. Correspondingly, it can be found that the proposed MRPP method can plan almost straight paths step by step for the aforementioned robots. Meanwhile, the other robots have more difficult tasks, but they are also capable of avoiding obstacles and maintaining safe distances from them. The change in directions of robots in their midway is because the planning algorithm needs to consider the distance between all the robots and the distance from the obstacles surrounding them while minimizing the total path length.

In fact, Figure 6.18 is found more complicated because of the increase in the number of robots. Due to the starting and ending points we set, it is hard to avoid that some robots' paths cross together. Actually, certain intersections of robots' connection lines between corresponding points **S** and **T** can be found from the initial configuration in Figure 6.17. Nevertheless, as mentioned before, different robots arrive at the same area at different moments. And it is worth mentioning that no robot collisions were actually detected in the simulations.



(a) Initial state **(b)** Step 1 **(c)** Step 2

(d) Step 3 **(e)** Step 4: $r5$, $r7$, $r8$, $r9$, $r10$ & $r12$ arrived **(f)** Step 5: $r3$ arrived

(g) Step 6: $r1$ & $r11$ arrived **(h)** Step 7: MRPP accomplished

Fig. 6.18. MRPP process for twelve robots via ABCL algorithm in Map 1

6.4 Conclusion

In this chapter, problems with more practical implications are concerned, namely the global path planning and local path planning (corresponding to the SRPP and MRPP problems we constructed, respectively). The main objective is to provide more effective solutions for these RPP problems by using our proposed ABC algorithms.

In the first part, after formulating the SRPP problem, two groups of comparisons are carried out. For a more comprehensive comparison, all the ABC variants involved in the previous chapters' comparisons were included in the first group. According to the comparisons, the proposed FOABC and ABCL algorithms have been found outstanding among the 15 algorithms in terms of path length and execution time. ABC_RL and ABCDC are also comparable. Then, the proposed ABC algorithms are also compared to four well-known path planners. The comparison results demonstrate the advantages of our proposed ABCs in average path length while the A* and PRM are also competitive. In addition, when the situation becomes complex, the running time of our proposed algorithms is not affected much. So this difference in running time is acceptable.

Secondly, the proposed ABCL algorithm is adopted to solve on-line MRPP problems in static environments considering its superiority in runtime. Meanwhile, a new implementation method is designed. In each step, ABCL is adopted to determine the subsequent positions for all the robots. The effectiveness of ABCL is proved by achieving MRPP tasks for six robots and twelve robots. The simulation results demonstrate the efficiency of ABCL in accomplishing MRPP in terms of path length, safety, and arrival time.

As a result, implementation methods for solving different RPP problems via meta-heuristic algorithms have been proposed. And the effectiveness of our proposed ABC algorithms in practical problems is also demonstrated through simulations and comparisons.

6. ROBOT PATH PLANNING VIA IMPROVED ABC ALGORITHMS

Conclusions and Perspectives

Summary of main results

This thesis focuses on a class of optimization algorithms represented by the ABC algorithm. Several novel improved ABC variants are proposed through a systematic study of the basic theory and research status of meta-heuristic algorithms with theoretical studies of RL, FOC, and heavy-tailed distributions. Different types of improvement strategies are explored to make the algorithms more successful at handling various functional optimization problems based on investigations of the structure and characteristics of ABC. In parallel, multi-robot systems, as another research interest, are used to test and verify the effectiveness of the proposed ABC algorithms in real-world applications.

Firstly, in Chapter 2, the impact of population composition of ABC is investigated in addition to certain widely mentioned improvement strategies. It has been found that in basic ABC, the invariable population composition of the bee colony cannot satisfy the needs of different search stages. In this case, improving the effectiveness of ABC by adjusting the population composition is developed. Therefore, ABC with dynamic population composition, namely ABCDC is proposed. The main contributions are as follows: firstly, the Symmetric Latin Hypercube Design (SLHD) is adopted in initialization to improve the population diversity. Secondly, a novel mechanism is proposed to adjust the colony population's composition according to the searching experiences. More precisely, the division of labor between bees with different functions is clearer, so that global optimum can be obtained more efficiently under their cooperation. And experimental studies on functional optimization problems are done to verify the performance of ABCDC. The comparisons show that ABCDC has better solution precision and a faster convergence rate.

In Chapter 3, a particular improvement strategy is investigated considering that it is difficult to define the parameter value appropriately for all kinds of problems. In fact, the control parameters are usually constant or updated with predetermined adaptation methods, which heavily rely on the designer's experience. In this context, a RL-based ABC algorithm is proposed, named ABC_RL. The RL method is utilized to vary a newly introduced control parameter, i.e., number of dimensions to be updated in the solution search equation. The reward value of RL is defined based on the update results. In this case, more information can be learned appropriately from the previous experience. To sum up, the main contributions can be summarized as follows: firstly, RL is adopted to enlarge and adjust the frequency of perturbation of employed bee phase intelligently. Secondly, two enhanced solution search equations are utilized being inspired by the operators of DE. Thirdly, a type of heavy-tailed distribution is used to generate the scale factors for increasing the randomness. The proposed ABC_RL is validated through comparisons with other improved ABC variants on CEC 2017 benchmark problems. The comparison results demonstrated that the proposed algorithm outperformed the other competitors in terms of solution accuracy and overall performance.

Thirdly, in Chapter 4, enhancing the performance of ABC while avoiding it becoming too complex is studied. Hence, a learning-based ABC (ABCL) algorithm is proposed. Since one of the most essential goals of improving such algorithms is to solve more practical problems, then its practicality and complexity must be considered. Therefore, to improve the performance of algorithm without overcomplicating it, the following strategies are utilized: firstly, the global best solution is adopted in the employed bee and scout bee phases to better guide the colony. Secondly, the learning phase of the TLBO algorithm is embedded in the onlooker bee phase to improve the exploitation ability and simplify the computational complexity. Furthermore, comparisons with other ABC variants are carried out on functional optimization problems in different dimensions cases. And the effectiveness of ABCL can be validated according to the results.

In Chapter 5, different from the proposed ABC variants in previous chapters, the FOC is incorporated into the ABC algorithm considering the memory properties of FOC. In the proposed FOABC algorithm, each time generating a new candidate solution, the previous foraging behaviors stored in memory are referred

to. The main contributions of this chapter are as follows. The FOC is incorporated into the onlooker bee phase to make full use of the historical experiences. Meanwhile, a differential search strategy is utilized in the employed bee phase to reinforce the exploration ability. And the scale factors of these search equations are generated via Lévy distribution. Finally, according to the experimental results, FOABC outperforms the other ABC algorithms in terms of solution accuracy and robustness. Moreover, it is very competitive in comparison with the other effective meta-heuristics.

Last but not the least, in Chapter 6, after investigating different improvement strategies to improve the effectiveness of the ABC algorithm, we attempted to apply them to more meaningful applications. Therefore, the proposed algorithms are adopted to find better solutions for different types of RPP problems. Firstly, the single robot path planning problem is established. Different environments with arbitrary obstacles are considered. Secondly, since multi-robot systems are demonstrating their advantages in more and more fields, the MRPP problems are concerned. For all the path planning challenges, the proposed ABC algorithms are compared to other well-known path planners in terms of path length and execution time. As a result, implementation methods for solving different RPP problems via meta-heuristic algorithms have been proposed. And the effectiveness of our proposed algorithms in practical problems is also demonstrated through simulations and comparisons.

Perspectives

The following directions will be explored in the future

- There is no doubt that the development of more efficient optimization algorithms will continue to be a popular research direction in the future. Improving the adaptive capabilities of meta-heuristic algorithms and combining them with other powerful techniques (e.g. machine learning) are two promising approaches. Adaptation methods or machine learning methods can make meta-heuristic algorithms more intelligent and thus be capable to deal with more complex problems. Moreover, as mentioned before, the classical deterministic optimization algorithms perform well in local search. Therefore, in future research, the principles of traditional methods can be

integrated with ABC to better balance its global and local search abilities. Nevertheless, it is a challenging task to effectively incorporate these methods into ABC while making sure that neither the original complexity nor its capacity to solve problems will decrease.

- It can be found that the search efficiency and accuracy of the ABC algorithm are relatively high. Therefore, applying the improved ABC algorithms effectively and extensively to more problems in various fields is another essential future task. As for the path planning problem, we can expand to more types of environments in the future, as it is tough to cover all possible situations in this thesis. Moreover, it is very meaningful to apply the proposed methods to real robots.

- Since its introduction, the ABC algorithm's structure and search equations have been modified in different ways, which have significantly increased search accuracy and convergence speed. However, the effectiveness of most meta-heuristics is verified through numerical experiments. And the theoretical study of such algorithms is still in the exploratory stage. In this context, it is one of the important tasks to theoretically prove the convergence, time complexity, and convergence speed of the ABC algorithm.

References

Abbas, N.H. & Ali, F.M. (2014). Path planning of an autonomous mobile robot using directed artificial bee colony algorithm. *International Journal of Computer Applications*, **96**. 21, 103, 149, 153, 155, 173

Abdel-Basset, M., Abdel-Fatah, L. & Sangaiah, A.K. (2018). Metaheuristic algorithms: A comprehensive review. In *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*, 185–231. 4, 6

Agarwal, D. & Bharti, P.S. (2021). Implementing modified swarm intelligence algorithm based on Slime moulds for path planning and obstacle avoidance problem in mobile robots. *Applied Soft Computing*, **107**, 107372. 168

Akay, B. & Karaboga, D. (2012). A modified artificial bee colony algorithm for real-parameter optimization. *Information Sciences*, **192**, 120–142. 13, 64

Alizadegan, A., Asady, B. & Ahmadpour, M. (2013). Two modified versions of artificial bee colony algorithm. *Applied Mathematics and Computation*, **225**, 601–609. 34, 35

Anuar, S., Selamat, A. & Sallehuddin, R. (2016). A modified scout bee for artificial bee colony algorithm and its performance on optimization problems. *Journal of King Saud University - Computer and Information Sciences*, **28**, 395–406. 94

Arora, S. & Majumdar, A. (2022). Machine learning and soft computing applications in textile and clothing supply chain: Bibliometric and network analyses to delineate future research agenda. *Expert Systems with Applications*, 117000. 66

REFERENCES

Aslan, S. (2022). An immune plasma algorithm with a modified treatment schema for UCAV path planning. *Engineering Applications of Artificial Intelligence*, **112**, 104789. 21

Aslan, S., Karaboga, D. & Badem, H. (2020). A new artificial bee colony algorithm employing intelligent forager forwarding strategies. *Applied Soft Computing*, **96**, 106656. 130, 155

Awad, N., Ali, M.Z. & Reynolds, R.G. (2015). A differential evolution algorithm with success-based parameter adaptation for CEC2015 learning-based optimization. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, 1098–1105, Sendai, Japan. 34

Awad, N.H., Ali, M.Z., Suganthan, P.N., Liang, J.J. & Qu, B.Y. (2017). Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective real-parameter numerical optimization. 34. 77

Aydoğdu, İ., Akin, A. & Saka, M.P. (2016). Design optimization of real world steel space frames using artificial bee colony algorithm with Levy flight distribution. *Advances in engineering software*, **92**, 1–14. 69

Babaoglu, I. (2015). Artificial bee colony algorithm with distribution-based update rule. *Applied Soft Computing*, **34**, 851 – 861. 14, 38

Badem, H., Basturk, A., Caliskan, A. & Yuksel, M.E. (2018). A new hybrid optimization method combining artificial bee colony and limited-memory BFGS algorithms for efficient numerical optimization. *Applied Soft Computing*, **70**, 826–844. 17

Banharnsakun, A., Achalakul, T. & Sirinaovakul, B. (2011). The best-so-far selection in artificial bee colony algorithm. *Applied Soft Computing*, **11**, 2888–2901. 13, 18, 65, 98

Bhattacharjee, P., Rakshit, P., Goswami, I., Konar, A. & Nagar, A.K. (2011). Multi-robot path-planning using artificial bee colony optimization algorithm. In *2011 Third World Congress on Nature and Biologically Inspired Computing*, 219–224, Salamanca, Spain. 21, 149

Brest, J., Greiner, S., Boskovic, B., Mernik, M. & Zumer, V. (2006). Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation*, **10**, 646–657. 41

Chen, M., Chen, J., Zeng, G., Lu, K. & Jiang, X. (2019a). An improved artificial bee colony algorithm combined with extremal optimization and Boltzmann selection probability. *Swarm and Evolutionary Computation*, **49**, 158–177. 17, 18

Chen, R., Yang, B., Li, S. & Wang, S. (2020). A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem. *Computers & Industrial Engineering*, **149**, 106778. 65, 66, 73

Chen, X., Xu, B., Mei, C., Ding, Y. & Li, K. (2018). Teaching–learning–based artificial bee colony for solar photovoltaic parameter estimation. *Applied Energy*, **212**, 1578–1588. 16, 99

Chen, X., Tianfield, H. & Li, K. (2019b). Self-adaptive differential artificial bee colony algorithm for global optimization problems. *Swarm and Evolutionary Computation*, **45**, 70–91. 15, 64, 80, 98, 116, 130, 155

Chen, Y., Luo, G., Mei, Y., Yu, J. & Su, X. (2016). UAV path planning using artificial potential field method updated by optimal control theory. *International Journal of Systems Science*, **47**, 1407–1420. 20

Chen, Y., Pi, D. & Wang, B. (2019c). Enhanced global flower pollination algorithm for parameter identification of chaotic and hyper-chaotic system. *Nonlinear Dynamics*, **97**, 1343–1358. 8

Chopra, N. & Ansari, M.M. (2022). Golden jackal optimization: A novel nature-inspired optimizer for engineering applications. *Expert Systems with Applications*, **198**, 116924. 3, 4

Châari, I., Koubaa, A., Bennaceur, H., Trigui, S. & Al-Shalfan, K. (2012). smartPATH: A hybrid ACO-GA algorithm for robot path planning. In *2012 IEEE congress on evolutionary computation*, 1–8. 20

Contreras-Cruz, M.A., Ayala-Ramirez, V. & Hernandez-Belmonte, U.H. (2015). Mobile robot path planning using artificial bee colony and evolutionary programming. *Applied Soft Computing*, **30**, 319–328. 18, 21, 149

Couceiro, M.S., Rocha, R.P., Ferreira, N.M.F. & Machado, J.A.T. (2012). Introducing the fractional-order darwinian PSO. *Signal, Image and Video Processing*, **6**, 343–350. 112, 117, 118, 136

Cui, L., Li, G., Lin, Q., Du, Z., Gao, W., Chen, J. & Lu, N. (2016). A novel artificial bee colony algorithm with depth-first search framework and elite-guided search equation. *Information Sciences*, **367-368**, 1012–1044. 14

Cui, L., Li, G., Wang, X., Lin, Q., Chen, J., Lu, N. & Lu, J. (2017a). A ranking-based adaptive artificial bee colony algorithm for global numerical optimization. *Information Sciences*, **417**, 169–185. 16, 80, 155

Cui, L., Li, G., Zhu, Z., Lin, Q., Wen, Z., Lu, N., Wong, K.C. & Chen, J. (2017b). A novel artificial bee colony algorithm with an adaptive population size for numerical function optimization. *Information Sciences*, **414**, 53–67. 16, 35, 45, 48, 94, 155

Cui, L., Zhang, K., Li, G., Wang, X., Yang, S., Ming, Z., Huang, J.Z. & Lu, N. (2018). A smart artificial bee colony algorithm with distance-fitness-based neighbor search and its application. *Future Generation Computer Systems*, **89**, 478–493. 35, 45

Cui, L., Deng, J., Zhang, Y., Tang, G. & Xu, M. (2020). Hybrid differential artificial bee colony algorithm for multi-item replenishment-distribution problem with stochastic lead-time and demands. *Journal of Cleaner Production*, **254**, 119873. 16, 64

Cui, Y., Hu, W. & Rahmani, A. (2022). Improved artificial bee colony algorithm with dynamic population composition for optimization problems. *Nonlinear Dynamics*, **107**, 743–760. 64, 65

Das, P., Behera, H. & Panigrahi, B. (2016a). A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning. *Swarm and Evolutionary Computation*, **28**, 14–28. 21, 149, 168

Das, P.K. & Jena, P.K. (2020). Multi-robot path planning using improved particle swarm optimization algorithm through novel evolutionary operators. *Applied Soft Computing*, **92**, 106312. 8, 20, 21, 148, 149, 165, 167

Das, S., Mullick, S.S. & Suganthan, P.N. (2016b). Recent advances in differential evolution – an updated survey. *Swarm and Evolutionary Computation*, **27**, 1 – 30. 8

Dennis, J.E., Jr & Moré, J.J. (1977). Quasi-Newton methods, motivation and theory. *SIAM review*, **19**, 46–89. 2

Deshmukh, A.B. & Usha Rani, N. (2019). Fractional-Grey Wolf optimizer-based kernel weighted regression model for multi-view face video super resolution. *International Journal of Machine Learning and Cybernetics*, **10**, 859–877. 112

Dijkstra, E.W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, **1**, 269–271. 20

Dillen, W., Lombaert, G. & Schevenels, M. (2021). A hybrid gradient-based/metaheuristic method for eurocode-compliant size, shape and topology optimization of steel structures. *Engineering Structures*, **239**, 112137. 4

Doering, J., Kizys, R., Juan, A.A., Fitó, A. & Polat, O. (2019). Metaheuristics for rich portfolio optimisation and risk management: Current state and future trends. *Operations Research Perspectives*, **6**, 100121. 6

Dorigo, M. (1992). Optimization, learning and natural algorithms. *Ph. D. Thesis, Politecnico di Milano*. 7

Dorigo, M., Birattari, M. & Stutzle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine*, **1**, 28–39. 7

Emary, E., Zawbaa, H.M. & Grosan, C. (2017). Experienced gray wolf optimization through reinforcement learning and neural networks. *IEEE transactions on neural networks and learning systems*, **29**, 681–694. 64

Farah, A. & Belazi, A. (2018). A novel chaotic Jaya algorithm for unconstrained numerical optimization. *Nonlinear Dynamics*, **93**, 1451–1480. 45

Feng, T., Wang, C., Zhang, J., Wang, B. & Jin, Y.F. (2022). An improved artificial bee colony-random forest (IABC-RF) model for predicting the tunnel deformation due to an adjacent foundation pit excavation. *Underground Space*, **7**, 514–527. 17

Fletcher, R. (2013). *Practical methods of optimization*. 2

Flor-Sánchez, C.O., Reséndiz-Flores, E.O. & Altamirano-Guerrero, G. (2022). Kernel-based gradient evolution optimization method. *Information Sciences*, **602**, 313–327. 3, 4

Formica, G. & Milicchio, F. (2020). Kinship-based differential evolution algorithm for unconstrained numerical optimization. *Nonlinear Dynamics*, **99**, 1341–1361. 35

Fragapane, G., De Koster, R., Sgarbossa, F. & Strandhagen, J.O. (2021). Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda. *European Journal of Operational Research*, **294**, 405–426. 148

Gao, H., Shi, Y., Pun, C.M. & Kwong, S. (2019). An improved artificial bee colony algorithm with its application. *IEEE Transactions on Industrial Informatics*, **15**, 1853–1865. 18, 48, 80, 130, 155

Gao, K., He, Z., Huang, Y., Duan, P. & Suganthan, P.N. (2020). A survey on meta-heuristics for solving disassembly line balancing, planning and scheduling problems in remanufacturing. *Swarm and Evolutionary Computation*, **57**. 8

Gao, W., Liu, S. & Huang, L. (2012). A global best artificial bee colony algorithm for global optimization. *Journal of Computational and Applied Mathematics*, **236**, 2741–2753. 13, 17, 65, 98

Gao, W., Liu, S. & Huang, L. (2013). A novel artificial bee colony algorithm based on modified search equation and orthogonal learning. *IEEE Transactions on Cybernetics*, **43**, 1011–1024. 17

Gao, W., Liu, S. & Huang, L. (2014). Enhancing artificial bee colony algorithm using more information-based search equations. *Information Sciences*, **270**, 112–133. 14, 15, 98

Gao, W., Chan, F.T., Huang, L. & Liu, S. (2015a). Bare bones artificial bee colony algorithm with parameter adaptation and fitness-based neighborhood. *Information Sciences*, **316**, 180–200. 14, 16, 65

Gao, W., Huang, L., Liu, S. & Dai, C. (2015b). Artificial bee colony algorithm based on information learning. *IEEE Transactions on Cybernetics*, **45**, 2827–2839. 15

Gao, W., Huang, L., Wang, J., Liu, S. & Qin, C. (2016). Enhanced artificial bee colony algorithm through differential evolution. *Applied Soft Computing*, **48**, 137 – 150. 16, 17, 36, 98

Gendreau, M. & Potvin, J.Y. (2005). Metaheuristics in combinatorial optimization. *Annals of Operations Research*, **140**, 189–213. 6

Genovesi, S., Mittra, R., Monorchio, A. & Manara, G. (2006). Particle swarm optimization for the design of frequency selective surfaces. *IEEE Antennas and Wireless Propagation Letters*, **5**, 277–279. 25

Gholizadeh, S., Danesh, M. & Gheyratmand, C. (2020). A new Newton metaheuristic algorithm for discrete performance-based design optimization of steel moment frames. *Computers & Structures*, **234**, 106250. 4

Gu, Y., Yu, Y. & Wang, H. (2017). Synchronization-based parameter estimation of fractional-order neural networks. *Physica A: Statistical Mechanics and its Applications*, **483**, 351–361. 117

Han, S.D. & Yu, J. (2019). Effective heuristics for multi-robot path planning in warehouse environments. In *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, 10–12. 21, 165

Han, S.D. & Yu, J. (2020). DDM: Fast near-optimal multi-robot path planning using diversified-path and optimal sub-problem solution database heuristics. *IEEE Robotics and Automation Letters*, **5**, 1350–1357. 22, 165

Harfouchi, F., Habbi, H., Ozturk, C. & Karaboga, D. (2018). Modified multiple search cooperative foraging strategy for improved artificial bee colony optimization with robustness analysis. *Soft Computing*, **22**, 6371–6394. 15

Hart, P., Nilsson, N. & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, **4**, 100–107. 20, 148

Hasan, A.H. & Mosa, A.M. (2018). Multi-robot path planning based on max–min ant colony optimization and D* algorithms in a dynamic environment. In *2018 International Conference on Advanced Science and Engineering (ICOASE)*, 110–115. 21, 22, 149

Hawa, M. (2013). Light-assisted A∗ path planning. *Engineering Applications of Artificial Intelligence*, **26**, 888–898. 20, 148

Heidari, A.A. & Pahlavani, P. (2017). An efficient modified grey wolf optimizer with Lévy flight for optimization tasks. *Applied Soft Computing*, **60**, 115–134. 69

Holland, J. (1975). Adaptation in natural and artificial systems. 6

Holland, J.H. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press. 6

Houssein, E.H., Emam, M.M., Ali, A.A. & Suganthan, P.N. (2021). Deep and machine learning techniques for medical imaging-based breast cancer: A comprehensive review. *Expert Systems with Applications*, **167**, 114161. 66

Hu, W., Yu, Y. & Zhang, S. (2015). A hybrid artificial bee colony algorithm for parameter identification of uncertain fractional-order chaotic systems. *Nonlinear Dynamics*, **82**, 1441–1456. 17, 18, 35, 36

Hu, W., Yu, Y. & Gu, W. (2018). Parameter estimation of fractional-order arbitrary dimensional hyperchaotic systems via a hybrid adaptive artificial bee colony algorithm with simulated annealing algorithm. *Engineering Applications of Artificial Intelligence*, **68**, 172 – 191. 18

Hu, W., Wen, G., Rahmani, A. & Yu, Y. (2019). Parameters estimation using mABC algorithm applied to distributed tracking control of unknown nonlinear fractional-order multi-agent systems. *Communications in Nonlinear Science and Numerical Simulation*, **79**, 104933. 8

Hu, Z., Gong, W. & Li, S. (2021). Reinforcement learning-based differential evolution for parameters extraction of photovoltaic models. *Energy Reports*, **7**, 916–928. 65, 67, 71, 77

Huillet, T.E. (2016). On Mittag-Leffler distributions and related stochastic processes. *Journal of Computational and Applied Mathematics*, **296**, 181–211. 69

Hussein, S., Kandel, P., Bolan, C.W., Wallace, M.B. & Bagci, U. (2019). Lung and pancreatic tumor characterization in the deep learning era: Novel supervised and unsupervised learning approaches. *IEEE Transactions on Medical Imaging*, **38**, 1777–1787. 66

Imanian, N., Shiri, M.E. & Moradi, P. (2014). Velocity based artificial bee colony algorithm for high dimensional continuous optimization problems. *Engineering Applications of Artificial Intelligence*, **36**, 148–163. 14

Jadon, S.S., Tiwari, R., Sharma, H. & Bansal, J.C. (2017). Hybrid artificial bee colony algorithm with differential evolution. *Applied Soft Computing*, **58**, 11–24. 16, 64, 65

Jensi, R. & Jiji, G.W. (2016). An enhanced particle swarm optimization with levy flight for global optimization. *Applied Soft Computing*, **43**, 248–261. 69

Ji, J., Song, S., Tang, C., Gao, S., Tang, Z. & Todo, Y. (2019). An artificial bee colony algorithm search guided by scale-free networks. *Information Sciences*, **473**, 142–165. 17

Jiang, M., Leung, K., Lyu, Z. & Huang, G.Q. (2020). Picking-replenishment synchronization for robotic forward-reserve warehouses. *Transportation Research Part E: Logistics and Transportation Review*, **144**, 102138. 18

Jose, K. & Pratihar, D.K. (2016). Task allocation and collision-free path planning of centralized multi-robots system for industrial plant inspection using heuristic methods. *Robotics and Autonomous Systems*, **80**, 34–42. 22

Kala, R. (2014a). Code for robot path planning using A* algorithm. Indian Institute of Information Technology Allahabad. 161

Kala, R. (2014b). Code for robot path planning using bidirectional rapidly-exploring random trees. Indian Institute of Information Technology Allahabad. 161

Kala, R. (2014c). Code for robot path planning using probabilistic roadmap. Indian Institute of Information Technology Allahabad. 161

Kala, R. (2014d). Code for robot path planning using rapidly-exploring random trees. Indian Institute of Information Technology Allahabad. 161

Kalantzis, G., Shang, C., Lei, Y. & Leventouri, T. (2016). Investigations of a GPU-based levy-firefly algorithm for constrained optimization of radiation therapy treatment planning. *Swarm and Evolutionary Computation*, **26**, 191–201. 69

Kang, F., Li, J. & Ma, Z. (2011). Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions. *Information Sciences*, **181**, 3508 – 3531. 17, 45, 103

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. 10. 6, 7, 8, 48, 80, 130, 155

Karaboga, D. & Gorkemli, B. (2014). A quick artificial bee colony (qABC) algorithm and its performance on optimization problems. *Applied Soft Computing*, **23**, 227–238. 15, 65

Kennedy, J. & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, 1942 – 1948. 6, 7, 24

Kenny, Q.Y., Li, W. & Sudjianto, A. (2000). Algorithmic construction of optimal symmetric latin hypercube designs. *Journal of statistical planning and inference*, **90**, 145 – 159. 36

Khan, A.T., Li, S. & Cao, X. (2021). Control framework for cooperative robots in smart home using bio-inspired neural network. *Measurement*, **167**, 108253. 18

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. In I.J. Cox & G.T. Wilfong, eds., *Autonomous Robot Vehicles*, 396–404, New York, NY. 20

Kiran, M.S., Hakli, H., Gunduz, M. & Uguz, H. (2015). Artificial bee colony algorithm with variable search strategy for continuous optimization. *Information Sciences*, **300**, 140–157. 15

Kirkpatrick, S., Gelatt Jr, C.D. & Vecchi, M.P. (1983). Optimization by simulated annealing. *science*, **220**, 671–680. 6, 7

Kong, D., Chang, T., Dai, W., Wang, Q. & Sun, H. (2018). An improved artificial bee colony algorithm based on elite group guidance and combined breadth-depth search strategy. *Information Sciences*, **442-443**, 54–71. 14, 15

Koubâa, A., Bennaceur, H., Chaari, I., Trigui, S., Ammar, A., Sriti, M.F., Alajlan, M., Cheikhrouhou, O. & Javed, Y. (2018). *Robot path planning and cooperation: foundations, algorithms and experimentations*, vol. 772. 18, 19, 20, 148, 166, 168

Kozubowski, T.J. & Rachev, S.T. (1999). Univariate geometric stable laws. *Journal of Computational Analysis and Applications*, **1**, 177–217. 70

Kuang, F., Jin, Z., Xu, W. & Zhang, S. (2014). A novel chaotic artificial bee colony algorithm based on tent map. *2014 IEEE Congress on Evolutionary Computation (CEC)*, 235 – 241. 17

Kiran, M.S. & Findik, O. (2015). A directed artificial bee colony algorithm. *Applied Soft Computing*, **26**, 454 – 462. 14

Kiran, M.S. & Gündüz, M. (2013). A recombination-based hybridization of particle swarm optimization and artificial bee colony algorithm for continuous optimization problems. *Applied Soft Computing*, **13**, 2188–2203. 16

Lee, C.Y. & Yao, X. (2004). Evolutionary programming using mutations based on the Lévy probability distribution. *IEEE Transactions on Evolutionary Computation*, **8**. 114

Lei, D., Cui, Z. & Li, M. (2022). A dynamical artificial bee colony for vehicle routing problem with drones. *Engineering Applications of Artificial Intelligence*, **107**, 104510. 18

Lévy, P. (1938). Théorie de l'addition des variables aléatoires. *Bulletin of the American Mathematical Society*, **44**, 19–20. 7, 26, 114

Li, B. & Jiang, W. (2000). A novel stochastic optimization algorithm. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, **30**, 193–198. 3

Li, J. & Pan, Q. (2015). Solving the large-scale hybrid flow shop scheduling problem with limited buffers by a hybrid artificial bee colony algorithm. *Information Sciences*, **316**, 487 – 502. 17

Li, K., Ge, F., Han, Y., Wang, Y. & Xu, W. (2020a). Path planning of multiple UAVs with online changing tasks by an ORPFOA algorithm. *Engineering Applications of Artificial Intelligence*, **94**, 103807. 21, 165

Li, P., Yang, H., Li, H. & Liang, S. (2022). Nonlinear ESO-based tracking control for warehouse mobile robots with detachable loads. *Robotics and Autonomous Systems*, **149**, 103965. 18

Li, S., Chen, H., Wang, M., Heidari, A.A. & Mirjalili, S. (2020b). Slime mould algorithm: A new method for stochastic optimization. *Future Generation Computer Systems*, **111**, 300–323. 8

Li, X. & Yin, M. (2014). Parameter estimation for chaotic systems by hybrid differential evolution algorithm and artificial bee colony algorithm. *Nonlinear Dynamics*, **77**, 61–71. 8, 16, 48, 116, 155

Li, X., Shi, J., Dong, X. & Yu, J. (2019). A new conjugate gradient method based on Quasi-Newton equation for unconstrained optimization. *Journal of Computational and Applied Mathematics*, **350**, 372–379. 2

Li, Z., Wang, W., Yan, Y. & Li, Z. (2015). PS–ABC: A hybrid algorithm based on particle swarm and artificial bee colony for high-dimensional optimization problems. *Expert Systems with Applications*, **42**, 8881–8895. 16, 65

Liang, J.H. & Lee, C.H. (2015). Efficient collision-free path-planning of multiple mobile robots system using efficient artificial bee colony algorithm. *Advances in Engineering Software*, **79**, 47–56. 18, 21, 149

Liang, J.J., Qin, A.K., Suganthan, P.N. & Baskar, S. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation*, **10**, 281–295. 136

Liang, Z., Hu, K., Zhu, Q. & Zhu, Z. (2017). An enhanced artificial bee colony algorithm with adaptive differential operators. *Applied Soft Computing*, **58**, 480–494. 98

Lin, M.H., Tsai, J.F. & Yu, C.S. (2012). A review of deterministic optimization methods in engineering and management. *Mathematical Problems in Engineering*, **2012**. 3

Lin, Q., Zhu, M., Li, G., Wang, W., Cui, L., Chen, J. & Lu, J. (2018). A novel artificial bee colony algorithm with local and global information interaction. *Applied Soft Computing*, **62**, 702–735. 15, 65

Liu, F., Sun, Y., Wang, G. & Wu, T. (2018). An artificial bee colony algorithm based on dynamic penalty and lévy flight for constrained optimization problems. *Arabian Journal for Science and Engineering*, **43**, 7189–7208. 114, 116

Liu, X. (2016). Optimization design on fractional order PID controller based on adaptive particle swarm optimization algorithm. *Nonlinear Dynamics*, **84**, 379–386. 8

Lu, R., Hu, H., Xi, M., Gao, H. & Pun, C.M. (2019). An improved artificial bee colony algorithm with fast strategy, and its application. *Computers & Electrical Engineering*, **78**, 79–88. 18

Lyridis, D.V. (2021). An improved ant colony optimization algorithm for unmanned surface vehicle local path planning with multi-modality constraints. *Ocean Engineering*, **241**, 109890. 21, 149

Ma, Y., Zhang, X., Song, J. & Chen, L. (2021). A modified teaching–learning-based optimization algorithm for solving optimization problem. *Knowledge-Based Systems*, **212**, 106599. 100

Mantegna, R.N. (1994). Fast, accurate algorithm for numerical simulation of levy stable stochastic processes. *Physical Review E*, **49**, 4677. 114

Maqsood, S., Xu, S., Tran, S., Garg, S., Springer, M., Karunanithi, M. & Mohawesh, R. (2022). A survey: From shallow to deep machine learning approaches for blood pressure estimation using biosensors. *Expert Systems with Applications*, **197**, 116788. 66

Meike, D. & Ribickis, L. (2011). Energy efficient use of robotics in the automobile industry. In *2011 15th International Conference on Advanced Robotics (ICAR)*, 507–511, Tallinn, Estonia. 18

Miao, C., Chen, G., Yan, C. & Wu, Y. (2021). Path planning optimization of indoor mobile robot based on adaptive ant colony algorithm. *Computers & Industrial Engineering*, **156**, 107230. 21, 149

Mousavi, Y. & Alfi, A. (2018). Fractional calculus-based firefly algorithm applied to parameter estimation of chaotic systems. *Chaos, Solitons & Fractals*, **114**, 202–215. 112, 117, 118, 136

Nazarahari, M., Khanmirza, E. & Doostie, S. (2019). Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm. *Expert Systems with Applications*, **115**, 106–120. 20, 21, 22, 148, 149, 168

Nikolaev, A.G. & Jacobson, S.H. (2010). Simulated annealing. In *Handbook of metaheuristics*, 1–39. 7

Oleiwi, B.K., Al-Jarrah, R., Roth, H. & Kazem, B.I. (2014). Multi objective optimization of trajectory planning of non-holonomic mobile robot in dynamic environment using enhanced GA by fuzzy motion control and A. In

International Conference on Neural Networks and Artificial Intelligence, 34–49. 168

Öztürk, c., Ahmad, R. & Akhtar, N. (2020). Variants of artificial bee colony algorithm and its applications in medical image processing. *Applied Soft Computing*, **97**, 106799. 18

Park, C., Kyung, J.H., Choi, T.Y., Do, H.M., Kim, B.I. & Lee, S.H. (2012). Design of an industrial dual arm robot manipulator for a human-robot hybrid manufacturing. In *2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, 616–618, Daejeon, Korea (South). 18

Pearl, J. (1984). *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley Longman Publishing Co., Inc. 3

Plaksina, I., Chistokhina, G. & Topolskiy, D. (2018). Development of a transport robot for automated warehouses. In *2018 International Multi-Conference on Industrial Engineering and Modern Technologies)*, 1–4, Vladivostok. 18

Podlubny, I. (1999). An introduction to fractional derivatives, fractional differential equations, to methods of their solution and some of their applications. *Mathematics in Science and Engineering*, **198**, 340. 117

Qin, A., Huang, V. & Suganthan, P. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, **13**, 398–417. 23, 64

Qin, A.K. & Suganthan, P.N. (2005). Self-adaptive differential evolution algorithm for numerical optimization. In *2005 IEEE Congress on Evolutionary Computation*, vol. 2, 1785–1791, Edinburgh, Scotland, UK. 34, 41

Qu, B.Y., Suganthan, P.N. & Das, S. (2013). A distance-based locally informed particle swarm model for multimodal optimization. *IEEE Transactions on Evolutionary Computation*, **17**, 387–402. 136

Rao, R.V., Savsani, V.J. & Vakharia, D. (2011). Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. *Computer-aided design*, **43**, 303–315. 96

Regis, R.G. & Shoemaker, C.A. (2004). Local function approximation in evolutionary algorithms for the optimization of costly functions. *IEEE Transactions on Evolutionary Computation*, **8**, 490 – 505. 36

Rekaby, A., Youssif, A.A. & Eldin, A.S. (2013). Introducing adaptive artificial bee colony algorithm and using it in solving traveling salesman problem. *Science and Information Conference 2013*, 502 – 506. 18

Sahu, B., Kumar Das, P. & Kabat, M.r. (2022). Multi-robot cooperation and path planning for stick transporting using improved Q-learning and democratic robotics PSO. *Journal of Computational Science*, **60**, 101637. 22

Salimi, H. (2015). Stochastic fractal search: A powerful metaheuristic algorithm. *Knowledge-Based Systems*, **75**, 1–18. 7

Samma, H., Mohamad-Saleh, J., Suandi, S.A. & Lahasan, B. (2020). Q-learning-based simulated annealing algorithm for constrained engineering design problems. *Neural Computing and Applications*, **32**, 5147–5161. 65

Sedighi, K.H., Ashenayi, K., Manikas, T.W., Wainwright, R.L. & Tai, H.M. (2004). Autonomous local path planning for a mobile robot using a genetic algorithm. In *Proceedings of the 2004 congress on evolutionary computation (IEEE Cat. No. 04TH8753)*, vol. 2, 1338–1345. 168

Shahrabi, J., Adibi, M.A. & Mahootchi, M. (2017). A reinforcement learning approach to parameter estimation in dynamic job shop scheduling. *Computers & Industrial Engineering*, **110**, 75–82. 73

Sharma, H., Bansal, J.C., Arya, K.V. & Yang, X.S. (2016). Lévy flight artificial bee colony algorithm. *International Journal of Systems Science*, **47**, 2652–2670. 17, 114, 115

Shlesinger, M.F. (1989). Levy flights: Variations on a theme. *Physica D: Nonlinear Phenomena*, **38**, 304–309. 7, 26, 114

Solteiro Pires, E.J., Tenreiro Machado, J.A., de Moura Oliveira, P.B., Boaventura Cunha, J. & Mendes, L. (2010). Particle swarm optimization with fractional-order velocity. *Nonlinear Dynamics*, **61**, 295–301. 112

Song, X., Yan, Q. & Zhao, M. (2017). An adaptive artificial bee colony algorithm based on objective function value information. *Applied Soft Computing*, **55**, 384–401. 15, 65

Song, X., Zhao, M., Yan, Q. & Xing, S. (2019). A high-efficiency adaptive artificial bee colony algorithm using two strategies for continuous optimization. *Swarm and Evolutionary Computation*, **50**, 100549. 6, 14

Storn, R. & Price, K. (1997). Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, **11**, 341–359. 6, 7, 22

Su, H., Zhao, D., Yu, F., Heidari, A.A., Zhang, Y., Chen, H., Li, C., Pan, J. & Quan, S. (2022). Horizontal and vertical search artificial bee colony for image segmentation of COVID-19 X-ray images. *Computers in Biology and Medicine*, **142**, 105181. 18

Sutton, R.S. & Barto, A.G. (2018). *Reinforcement learning: An introduction*. MIT press. 64, 66, 67, 72

Szczepanski, R., Tarczewski, T., Erwinski, K. & Grzesiak, L.M. (2018). Comparison of constraint-handling techniques used in artificial bee colony algorithm for auto-tuning of state feedback speed controller for PMSM. In *ICINCO (1)*, 279–286. 153

Tao, X., Pan, Q. & Gao, L. (2022). An efficient self-adaptive artificial bee colony algorithm for the distributed resource-constrained hybrid flowshop problem. *Computers & Industrial Engineering*, **169**, 108200. 18

Thabit, S. & Mohades, A. (2019). Multi-robot path planning based on multi-objective particle swarm optimization. *IEEE Access*, **7**, 2138–2147. 21, 22, 149

Tian, S., Li, Y., Kang, Y. & Xia, J. (2021). Multi-robot path planning in wireless sensor networks based on jump mechanism PSO and safety gap obstacle avoidance. *Future Generation Computer Systems*, **118**, 37–47. 21, 149

Tu, J., Chen, H., Wang, M. & Gandomi, A.H. (2021). The colony predation algorithm. *Journal of Bionic Engineering*, **18**, 674–710. 8

Tuncer, A. & Yildirim, M. (2012). Dynamic path planning of mobile robots with improved genetic algorithm. *Computers & Electrical Engineering*, **38**, 1564–1572. 21, 149

Wang, G.G., Deb, S. & Cui, Z. (2019). Monarch butterfly optimization. *Neural Computing and Applications*, **31**, 1995–2014. 8

Wang, H., Wang, J. & Huang, M. (2013). Building a smart home system with WSN and service robot. In *2013 Fifth International Conference on Measuring Technology and Mechatronics Automation*, 353–356, Hong Kong. 18

Wang, H., Wang, W., Xiao, S., Cui, Z., Xu, M. & Zhou, X. (2020). Improving artificial bee colony algorithm using a new neighborhood selection mechanism. *Information Sciences*, **527**, 227–240. 15, 16, 35, 45, 48, 65, 98, 130, 155

Wang, S., Hu, W., Riego, I. & Yu, Y. (2022). Improved surrogate-assisted whale optimization algorithm for fractional chaotic systems' parameters identification. *Engineering Applications of Artificial Intelligence*, **110**, 104685. 114

Wang, Z., Li, M., Dou, L., Li, Y., Zhao, Q. & Li, J. (2015). A novel multi-objective artificial bee colony algorithm for multi-robot path planning. In *2015 IEEE International Conference on Information and Automation*, 481–486, Lijiang, China. 21, 149

Watkins, C.J.C.H. & Dayan, P. (1992). Q-learning. *Machine Learning*, **8**, 279–292. 67

Wei, J., Chen, Y., Yu, Y. & Chen, Y. (2019). Optimal randomness in swarm-based search. *Mathematics*, **7**, 828. 69, 70

Wilson, G., Pereyda, C., Raghunath, N., de la Cruz, G., Goel, S., Nesaei, S., Minor, B., Schmitter-Edgecombe, M., Taylor, M.E. & Cook, D.J. (2019). Robot-enabled support of daily activities in smart home environments. *Cognitive Systems Research*, **54**, 258–272. 18

Wu, G., Mallipeddi, R., Suganthan, P., Wang, R. & Chen, H. (2016). Differential evolution with multi-population based ensemble of mutation strategies. *Information Sciences*, **329**, 329–345. 64

Xiang, W. & An, M. (2013). An efficient and robust artificial bee colony algorithm for numerical optimization. *Computers & Operations Research*, **40**, 1256–1265. 14, 17, 94

Xiang, W., Li, Y., Meng, X., Zhang, C. & An, M. (2017). A grey artificial bee colony algorithm. *Applied Soft Computing*, **60**, 1–17. 71

Xiang, W., Li, Y., He, R. & An, M. (2021). Artificial bee colony algorithm with a pure crossover operation for binary optimization. *Computers & Industrial Engineering*, **152**, 107011. 15, 71

Xiang, Y., Peng, Y., Zhong, Y., Chen, Z., Lu, X. & Zhong, X. (2014). A particle swarm inspired multi-elitist artificial bee colony algorithm for real-parameter optimization. *Computational Optimization and Applications*, **57**, 493–516. 14

Xiang, Y., Zhou, Y. & Liu, H. (2015). An elitism based multi-objective artificial bee colony algorithm. *European Journal of Operational Research*, **245**, 168–193. 14

Xiao, S., Wang, H., Wang, W., Huang, Z., Zhou, X. & Xu, M. (2021). Artificial bee colony algorithm based on adaptive neighborhood search and Gaussian perturbation. *Applied Soft Computing*, **100**, 106955. 15, 98

Xing, G. (2021). Motion control method of multi degree of freedom industrial robot for intelligent manufacturing. In *2021 2nd International Conference on Intelligent Design (ICID)*, 6–9, Xi'an, China. 18

Xu, F., Li, H., Pun, C.M., Hu, H., Li, Y., Song, Y. & Gao, H. (2020). A new global best guided artificial bee colony algorithm with application in robot path planning. *Applied Soft Computing*, **88**, 106037. 18, 21, 70, 149

Xu, R., Wang, Q., Shi, L. & Chen, L. (2011). Design of multi-robot path planning system based on hierarchical fuzzy control. *Procedia Engineering*, **15**, 235–239. 22

Xue, Y., Jiang, J., Zhao, B. & Ma, T. (2018). A self-adaptive artificial bee colony algorithm based on global best for global optimization. *Soft Computing*, **22**, 2935–2952. 34

Yameni Noupoue, Y.Y., Tandoğdu, Y. & Awadalla, M. (2019). On numerical techniques for solving the fractional logistic differential equation. *Advances in Difference Equations*, **2019**, 108. 118

Yang, X.S. (2009). Firefly algorithms for multimodal optimization. In O. Watanabe & T. Zeugmann, eds., *Stochastic Algorithms: Foundations and Applications*, vol. 5792, 169–178, Berlin, Heidelberg. 6, 54

Yang, X.S. (2020). *Nature-inspired optimization algorithms*. 2, 3, 27, 69, 114

Yang, X.S. & Deb, S. (2009). Cuckoo search via Lévy flights. In *2009 World congress on nature & biologically inspired computing (NaBIC)*, 210–214. 7, 26, 69, 114

Yavuz, G. & Aydin, D. (2019). Improved self-adaptive search equation-based artificial bee colony algorithm with competitive local search strategy. *Swarm and Evolutionary Computation*, **51**, 100582. 15

Yin, S., Ji, W. & Wang, L. (2019). A machine learning based energy efficient trajectory planning approach for industrial robots. *Procedia CIRP*, **81**, 429–434. 18

Yousri, D. & Mirjalili, S. (2020). Fractional-order cuckoo search algorithm for parameter identification of the fractional-order chaotic, chaotic with noise and hyper-chaotic financial systems. *Engineering Applications of Artificial Intelligence*, **92**, 103662. 112, 136

Yousri, D., Abd Elaziz, M. & Mirjalili, S. (2020). Fractional-order calculus-based flower pollination algorithm with local search for global optimization and image segmentation. *Knowledge-Based Systems*, **197**, 105889. 112, 117, 121

Yousri, D., Abd Elaziz, M., Abualigah, L., Oliva, D., Al-qaness, M.A. & Ewees, A.A. (2021). COVID-19 X-ray images classification based on enhanced fractional-order cuckoo search optimizer using heavy-tailed distributions. *Applied Soft Computing*, **101**, 107052. 69

Zabihi, F. & Nasiri, B. (2018). A novel history-driven artificial bee colony algorithm for data clustering. *Applied Soft Computing*, **71**, 226 – 241. 18

Zhang, J. & Sanderson, A.C. (2009). JADE: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, **13**, 945–958. 34, 41, 42, 64

Zhang, M., Tian, N., Palade, V., Ji, Z. & Wang, Y. (2018). Cellular artificial bee colony algorithm with Gaussian distribution. *Information Sciences*, **462**, 374–401. 14

Zhang, Y., Cheng, S., Shi, Y., Gong, D.w. & Zhao, X. (2019). Cost-sensitive feature selection using two-archive multi-objective artificial bee colony algorithm. *Expert Systems with Applications*, **137**, 46–58. 18

Zhao, H. & Zhang, C. (2020). A decomposition-based many-objective artificial bee colony algorithm with reinforcement learning. *Applied Soft Computing*, **86**, 105879. 64

Zhao, Z., Yang, J., Hu, Z. & Che, H. (2016). A differential evolution algorithm with self-adaptive strategy and control parameters based on symmetric Latin hypercube design for unconstrained optimization problems. *European Journal of Operational Research*, **250**, 30 – 45. 36

Zhou, X., Lu, J., Huang, J., Zhong, M. & Wang, M. (2021a). Enhancing artificial bee colony algorithm with multi-elite guidance. *Information Sciences*, **543**, 242–258. 64, 80, 130, 155

Zhou, Y., Zhang, W., Kang, J., Zhang, X. & Wang, X. (2021b). A problem-specific non-dominated sorting genetic algorithm for supervised feature selection. *Information Sciences*, **547**, 841 – 859. 8

Zhu, G. & Kwong, S. (2010). Gbest-guided artificial bee colony algorithm for numerical function optimization. *Applied Mathematics and Computation*, **217**, 3166 – 3173. 13, 16, 35, 45, 65, 97, 98, 99, 100, 103, 155, 173

Zorarpaci, E. & Özel, S.A. (2016). A hybrid approach of differential evolution and artificial bee colony for feature selection. *Expert Systems with Applications*, **62**, 91–103. 18, 116

Zou, F., Chen, D. & Xu, Q. (2019). A survey of teaching–learning-based optimization. *Neurocomputing*, **335**, 366–383. 96

Šeda, M. (2007). Roadmap methods vs. cell decomposition in robot motion planning. In *Proceedings of the 6th WSEAS international conference on signal processing, robotics and automation*, 127–132, World Scientific and Engineering Academy and Society (WSEAS) Athens, Greece. 20

Résumé Etendu

En tant que classe d'algorithmes stochastiques, les algorithmes méta-heuristiques sont efficaces pour résoudre des problèmes d'optimisation grâce à un compromis entre la randomisation et la recherche locale. De tels algorithmes sont généralement conçus en modélisant le comportement intelligent de certaines espèces. Ils se sont avérés efficaces et simples à comprendre. Ainsi, ils ont été utilisés pour résoudre de nombreux problèmes d'optimisation dans une variété de domaines. Néanmoins, il y a encore des pistes d'amélioration, comme la facilité d'être piégé dans les optimums locaux ou l'accélération de la vitesse de convergence. Et la gestion précise du compromis entre l'exploration et l'exploitation pour toutes les tâches d'optimisation est toujours un défi. Dans ce contexte, cette thèse traite une classe d'algorithmes méta-heuristiques représentée par l'algorithme ABC (Artificial Bee Colony). Une série de variantes améliorées en analysant les caractéristiques et les faiblesses de l'algorithme ABC a été proposée. De plus, des problèmes ayant une valeur d'application plus élevée sont également pris en compte. Les algorithmes ABC améliorés proposés ont résolu avec succès différents types de problèmes d'optimisation pratiques, y compris des tâches de planification de trajectoire pour un seul robot et multi-robots dans divers environnements.

Dans le premier chapitre, le contexte et les motivations de cette thèse sont présentés, suivis d'un aperçu des algorithmes méta-heuristiques. Ensuite, un bref résumé du problème de planification de trajectoire de robot est fourni. Enfin, les préliminaires de l'algorithme ABC original et quelques notions de base sont également donnés.

L'algorithme ABC est efficace pour explorer l'espace de recherche, en revanche, sa capacité à rechercher des régions prometteuses est limitée, ce qui entraîne un taux de convergence lent. Il contient trois phases fonctionnelles : la phase d'abeille employée, la phase d'abeille spectatrice et la phase d'abeille éclaireuse. La mission

d'exploration est principalement accomplie par les abeilles employées tandis que les abeilles spectatrices sont chargés d'exploiter dans certaines régions.

En plus de ces stratégies d'amélioration largement mentionnées, l'impact de la composition de la population est étudié dans le chapitre 2. En effet, la composition invariable de la population de la colonie d'abeilles ne peut pas satisfaire les besoins des différentes étapes de recherche. Dans ce contexte, l'amélioration de l'efficacité de l'ABC en ajustant la composition de la population est développée. Ainsi, un algorithme ABC avec composition dynamique de la population, à savoir ABCDC, est proposé. Les principales contributions de ce chapitre sont les suivantes : premièrement, le Symmetric Latin Hypercube Design (SLHD) est adoptée dans l'initialisation pour améliorer la diversité de la population. Ensuite, un nouveau mécanisme est proposé pour ajuster la composition de la population en fonction des expériences de recherche. Plus précisément, le nombre d'abeilles employées diminue périodiquement tandis que la taille des abeilles spectatrices augmente afin d'apporter plus d'énergie pour exploiter l'optimum global au stade moyen-tardif du processus de recherche. Dans l'ABCDC, la division du travail entre les différentes abeilles est plus claire, de sorte que l'optimum global peut être obtenu plus efficacement grâce à leur coopération. De plus, l'ABCDC maintient un bon équilibre entre la diversification et l'intensification. Des études expérimentales sur des problèmes d'optimisation fonctionnelle sont réalisées pour vérifier la performance d'ABCDC. Les comparaisons montrent que l'ABCDC a une meilleure précision de solution et un taux de convergence plus rapide.

En fait, il est difficile de définir les valeurs des paramètres de contrôle de manière appropriée pour tous les types de problèmes. Ainsi, ces paramètres de contrôle sont généralement constants ou mis à jour avec des méthodes d'adaptation prédéterminées, comme celle adoptée dans l'ABCDC. Cependant, les démarches d'adaptation reposent encore largement sur l'expérience du concepteur. Dans ce contexte, différent de la littérature existante, une nouvelle façon de définir les valeurs des paramètres est proposée dans le chapitre 3. Un algorithme ABC basé sur l'apprentissage par renforcement (RL) est proposé, nommé ABC_RL. La méthode RL est utilisée pour faire varier le nombre de dimensions à mettre à jour dans l'équation de recherche de solution. La valeur de récompense de RL est définie en fonction des résultats de la mise à jour. Dans ce cas, plus d'informations

peuvent être apprises de manière appropriée à partir de l'expérience de mise à jour précédente.

Les principales contributions du chapitre 3 peuvent être résumées comme suit : premièrement, RL est adopté pour élargir et ajuster intelligemment la fréquence de perturbation de la phase d'abeille employée en tenant compte de la récompense immédiate des résultats de mise à jour de la solution. Deuxièmement, deux équations de recherche de solution améliorées sont utilisées. De plus, un type de distribution à queue lourde, la distribution de Mittag-Leffler, est utilisé pour générer des facteurs d'échelle des équations de recherche. Enfin, l'ABC_RL proposé est comparé à d'autres algorithmes ABC améliorés sur un groupe de fonctions de référence et montre ses performances exceptionnelles..

Dans le chapitre 4, la praticité et la complexité de l'algorithme sont considérées comme plus importantes puisque l'un des objectifs les plus essentiels de l'amélioration de ces algorithmes est de résoudre des problèmes pratiques. Bien que de nombreuses stratégies de modification soient efficaces pour résoudre les problèmes d'optimisation fonctionnelle, elles n'aident pas toujours à obtenir rapidement la solution optimale dans les applications pratiques. Par conséquent, il est également important d'améliorer les performances de l'algorithme sans le compliquer à l'excès. C'est pourquoi le chapitre 4 étudie l'amélioration des performances de l'ABC tout en évitant qu'il ne devienne trop complexe. Dans ce contexte, un algorithme ABC basé sur l'apprentissage (ABCL) est proposé. Ainsi, il est possible d'économiser de l'énergie et du temps lors de la résolution de problèmes tels que la planification de chemins locaux.

Les principales contributions de ce chapitre sont les suivantes : premièrement, la meilleure solution globale est adoptée dans les phases d'abeille employée et d'abeille éclaireuse pour guider l'essaim dans une direction de recherche prometteuse. Deuxièmement, la phase d'apprentissage de l'algorithme TLBO est intégrée à la phase d'abeille spectatrice pour améliorer la capacité d'exploitation et simplifier la complexité de calcul. Les résultats expérimentaux et les simulations du dernier chapitre démontrent l'efficacité de l'ABCL tout en garantissant des solutions de haute qualité.

Dans les variantes de l'ABC proposées aux chapitres 2 à 4, les équations de recherche de solutions sont améliorées en élargissant le nombre de dimensions à mettre à jour et en augmentant la quantité d'informations pouvant être obtenues

à partir de la colonie. Cependant, ce type d'amélioration ignore en fait certaines informations utiles sur l'expérience de recherche historique. Il convient de souligner que, par rapport à la dérivée d'ordre entier, la dérivée d'ordre fractionnaire contient la mémoire entière de ses événements précédents. Par conséquent, contrairement aux résultats existants, le calcul d'ordre fractionnaire (FOC) est incorporé dans l'algorithme ABC en tenant compte des propriétés de mémoire du FOC. Dans l'algorithme FOABC proposé, chaque fois qu'une nouvelle solution candidate est générée, les comportements de fourrage précédents stockés en mémoire sont pris en compte.

Les principales contributions de ce chapitre incluent l'incorporation du concept de FOC dans la phase d'abeille spectatrice pour améliorer la capacité de recherche locale. Parallèlement, une stratégie de recherche différentielle est utilisée dans la phase d'abeille employée pour renforcer la capacité d'exploration. Et les facteurs d'échelle de ces équations de recherche sont générés via la distribution de Lévy pour augmenter le caractère aléatoire du FOABC. Des comparaisons sont effectuées sur un ensemble de problèmes pour valider les performances de FOABC.

Dans la dernière partie, après avoir étudié différentes stratégies d'amélioration pour accroître l'efficacité de l'algorithme ABC, nous avons appliqué ces algorithmes ABC améliorés pour résoudre différents types de problèmes de planification de trajectoire. Les algorithmes proposés sont adoptés pour trouver de meilleures solutions en un temps limité après avoir transformé ces tâches pratiques en problèmes d'optimisation.

Dans le chapitre 6, premièrement, ces méthodes sont utilisées pour compléter la planification globale de trajectoire pour un seul robot. Différents environnements avec des obstacles arbitraires sont considérés. Ensuite, nous avons considéré le problème de planification de trajectoire multi-robots. La tâche consiste à générer des trajectoires optimales sans collisions pour un groupe de robots depuis leurs positions de départ jusqu'aux points cibles prévus, tout en tenant compte des contraintes de l'environnement dans lequel ils évoluent. Pour tous les défis de planification de chemin, les algorithmes ABC proposés sont comparés à d'autres planificateurs de chemin bien connus en termes de longueur de chemin et de temps d'exécution.

Titre: Algorithmes améliorés de colonies d'abeilles artificielles pour la planification de la trajectoire des robots

Résumé: En tant que classe d'algorithmes stochastiques, les algorithmes méta-heuristiques sont efficaces pour résoudre des problèmes d'optimisation grâce à un compromis entre la randomisation et la recherche locale. De tels algorithmes sont avérés efficaces et simples à comprendre. Néanmoins, il y a encore des pistes d'amélioration, comme la facilité d'être piégé dans les optimums locaux ou l'accélération de la vitesse de convergence. Et la gestion précise du compromis entre l'exploration et l'exploitation pour toutes les tâches d'optimisation est toujours un défi. Dans ce contexte, cette thèse traite une classe d'algorithmes méta-heuristiques représentée par l'algorithme ABC (Artificial Bee Colony). Une série de variantes améliorées en analysant les caractéristiques et les faiblesses de l'algorithme ABC a été proposée. De plus, des problèmes ayant une valeur d'application plus élevée sont également pris en compte. Les algorithmes ABC améliorés proposés ont résolu avec succès différents types de problèmes d'optimisation pratiques, y compris des tâches de planification de trajectoire pour un seul robot et multi-robots dans divers environnements.

Mots-clés: Algorithme méta-heuristique, algorithme de colonies d'abeilles artificielles, planification de la trajectoire des robots, apprentissage par renforcement, calcul fractionnaire, optimisation globale.

Title: Improved artificial bee colony algorithms for robot path planning

Abstract: As a class of stochastic algorithms, meta-heuristic algorithms are effective for solving optimization problems through a trade-off between randomization and local search. Such algorithms have been found to be effective and simple to understand. Nevertheless, there is still room for improvement, such as easy to be trapped in local optimums or slow convergence speed. And precisely handling the trade-off between exploration and exploitation for all optimization tasks is always challenging. In this context, this thesis focused on a class of meta-heuristic algorithms represented by the Artificial Bee Colony (ABC) algorithm and proposed a series of improved variants by analyzing the characteristics and weaknesses of the ABC algorithm. Furthermore, problems with a higher application value are also taken into account. The proposed enhanced ABC versions have successfully solved different types of optimization problems, including robot path planning tasks for single and multiple robots in various environments.

Keywords: Meta-heuristic algorithm, artificial bee colony algorithm, robot path planning, reinforcement learning, fractional calculus, global optimization.