IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 24, 2020, accepted April 9, 2020, date of publication April 13, 2020, date of current version April 30, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2987689

Newly Emerging Nature-Inspired Optimization - Algorithm Review, Unified Framework, Evaluation, and Behavioural Parameter Optimization

HUI LI[1], (Member, IEEE), XIAO LIU[1], ZHIGUO HUANG[1], CHENBO ZENG[1], PENG ZOU[1], ZHAOYI CHU[1], AND JUNKAI YI[2]

[1]Department of Computer Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China
[2]Beijing Information Science and Technology University, Beijing 100101, China

Corresponding author: Hui Li (ray@mail.buct.edu.cn)

This work was supported by the National Natural Science Foundation of China under grants of the General Technical Foundation Research Joint Fund under Project U1636208.

ABSTRACT Nature-inspired optimization is a modern technique in the past decades. Researchers report their successful applications in various fields such as manufacturing, biomedical, and environmental engineering, while other researchers doubt its applicability. In this paper, we collect newly emerging nature-inspired optimization algorithms proposed after 2008, present them in a unified way, implement them, and evaluate them on benchmark functions. Moreover, we optimize the behavioural parameters for these algorithms. Since it is impossible to cover all interesting topics regarding nature-inspired optimization, this paper only focuses on the continuous encoding algorithms for single objective global problems, which is fundamental for other related topics.

INDEX TERMS Nature inspired optimization, meta-heuristics, unified framework, parameter optimization, meta-optimization.

I. INTRODUCTION

A. MOTIVATION

Conventional optimization methods such as the Newton-Raphson method and interior-point methods have been implemented in many software packages. However, the well-crafted software packages may still hardly solve some real-world problems. For example, for the nonlinear optimization problem which has large numbers of local optima, conventional optimization algorithms may lose their search ability. Nature-inspired optimization (NIO) algorithms have superior abilities to avoid local optima compared to conventional optimization techniques. Therefore, the NIO algorithm become an important option for solving some challenging real-world problems [1].

Animals and plants naturally develop strategies for millions of years to ensure their survival when resources are scarce [2]. It is understandable that the abundance and success of these strategies merit consideration in the development of algorithms for optimization. Over the past few decades, NIO algorithms such as particle swarm optimization and genetic algorithm have provided a wealth of meta-heuristics for solving desired problems. The No Free Lunch (NFL) theorem states that there is no meta-heuristic best suited for solving all optimization problems [3]. Consequently, researchers constantly propose new NIO algorithms, thereby keeping this field active and spurring steady progress every year. Since the research of NIO algorithms become more active from 2008 than before, in this paper, we focus on the newly emerging NIO algorithms that were proposed after 2008. We believe it is time to investigate whether these new NIO algorithms are powerful in solving challenging problems and how to use them with appropriate parameters.

B. SCOPE AND REVIEW METHODOLOGY

Optimization problems can be divided into two categories: those where solutions are encoded with real valued variables, and those where solutions are encoded with discrete

The associate editor coordinating the review of this manuscript and approving it for publication was Seyedali Mirjalili.

variables [4]. Continuous optimization problems arise for the former category, while we find combinatorial optimization problems for the latter category. We realize that it is really hard to examine and evaluate the NIO algorithms if we consider them both. So we limit our review to the continuous optimization problems. Moreover, we only examine the NIO algorithms for the basic global optimization problems, and do not go into various more complex problems such as multi-modal optimization problems [5], dynamic optimization problems, and multi-objective optimization problems. This is done in order to keep the scope of our analysis in such a breadth where it is possible to examine each NIO algorithm in detail rather than superficially.

Our studies adopt the guidelines for undertaking systematic review as recommended in [6]. The original set of papers is collected from the searchers run on the ''science citation index expanded'' (SCIE) collection of ''web of science'' (WOS) which is a high standard indexing system. The search is performed in 2019, which covers the period between 2008 and 2019. As a general rule, we include in the review the papers which firstly proposed a new NIO algorithm and which was then cited by other authors more than twice in the WOS core collection. Explanation should be added for ''firstly proposed'' that the first public appearance of some algorithms to other journals or conferences beyond the scope of SCIE collection may be earlier than ''firstly proposed'' in this paper. The citations are also limited within the scope of WOS core collection. There is an exception that the black widow optimization algorithm [7] has not been cited yet. The black widow optimization algorithm was just proposed at the end of 2019. We believe it will be cited in the near future. Corresponding resulting set of papers undergo manual reduplication.

The papers which are dedicated to hybrid algorithms, such as [8] which is hybridization between the NIO and non-NIO algorithms, [9] which is hybridization of two NIO algorithms, [10] which is hybridization of three or more NIO algorithms are not included in this review.

Each research paper we collect is closely related to a NIO algorithm. From these papers, we obtain over a dozen NIO original algorithms. We implement all algorithms in Python language, evaluate them on benchmark functions, and perform parameter optimization for them.

C. OTHER LITERATURE REVIEWS

There are some review works in the past decades. Fister *et al.* [11] gave a comprehensive list of nature inspired optimization algorithms from 1992 to 2013. A taxonomy for NIO algorithms was presented in this paper. Yang introduced another taxonomy in his review paper [12]. Kar [13] also summarized the development of 10 nature inspired algorithms from 1970s to 2015, and focused on applications in some specific environments. Valdez *et al.* [14] studied the importance and improvement in performance of the adaptiveness of the parameters of PSO, GSA, and ACO. Agarwal [15] conducted a detailed study of five NIO algorithms including ABC, FA, FPA, CS, and BA by testing the convergence performance on CEC'2014 (Institute of Electrical and Electronic Engineers, Congress on Evolutionary Computation 2014). Nanda and Panda [16] systematically summarized the single-objective NIO algorithm for cluster analysis. Molina *et al.* put forward several recommendations and points of improvement for better methodological practices, after critically analyzing more than three hundred papers of different types of nature and bio-inspired algorithms [17]. Since there are vast excellent NIOs and many topics in this field, it is inevitable that all the review papers to date only focused on a limited scope.

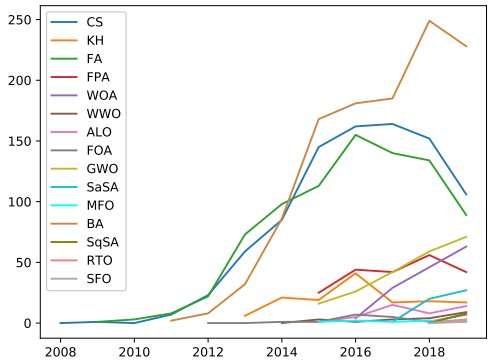


FIGURE 1. The timeline of NIO algorithms.

D. CONTRIBUTION

We select over a dozen NIO algorithms from articles published after 2008. Fig.1 counts the number of SCIE indexed articles related to each algorithm. The colors of lines differentiate the algorithms. The texts appearing in the legend are the abbreviations for various algorithms (see Table 1). Additionally, the past two years saw so many new competitive NIO algorithms that we can hardly put them all into this review. So, we list some of them in Table 2 without providing more details. The X variable represents the year, while the Y variable is the number of papers which appeared and were indexed in SCIE that year. We omit some papers unintentionally and inevitably since some papers do not fall into the scope of SCIE. Another reason is that titles and keywords do not suggest salient connection with the NIOs for some papers.

When introducing various NIOs, we present a new perspective concerning the number of elites, thus giving a new taxonomy for the NIO algorithms. Furthermore, we make efforts in unifying most NIO algorithms, though it is impossible to unify all NIO algorithms. Such a unified framework allows users to develop a well-structured NIO software. In fact, our implements of the NIO algorithms have already benefited from the unifying and abstraction.

It is always reasonable to ask if some methods are better than others. We selected 41 benchmark functions from [18]–[22] to test the performance of NIOs and compare them fairly and objectively. Even though the benchmark tests indicate that some NIO algorithms really demonstrably perform better on most functions than others, we cannot scientifically answer the above-mentioned question yet.

E. LIMITATIONS

Before we proceed with the description of NIO algorithms, the limitations of the review process should be noted. Firstly, the scope of this review is limited. As mentioned previously, we focused only on research publications dealing primarily with the continuous global optimization problems. The papers regarding combinatorial optimization problems, multi-modal optimization problems, multi-objective optimization problems, and dynamic optimization problems are all excluded. Secondly, the analysis in this paper is done based only on our understanding. We did not contact the authors to verify the correctness of our understanding. Thirdly, for some NIOs, there exists the original algorithm and some amended algorithms. It is impossible to exclude the factor of subjectivity while determining which is the most representative paper for a NIO algorithm. Fourthly, some high standard paper related to a NIO algorithm may does not find their place in this review due to it is not indexed by "Web of Science". Fifthly, we certainly miss some papers when the title, the abstract, or the keywords of the papers did not bring the paper into our collection. Finally, the review is only valid at the time of writing this paper. It is subject to change since NIO is a hot research topic recently. Nevertheless, the literature search method still ensure an acceptable level of the completeness of this review. Hence, we believe that the set of papers is representative and the results of the analysis may be generalized.

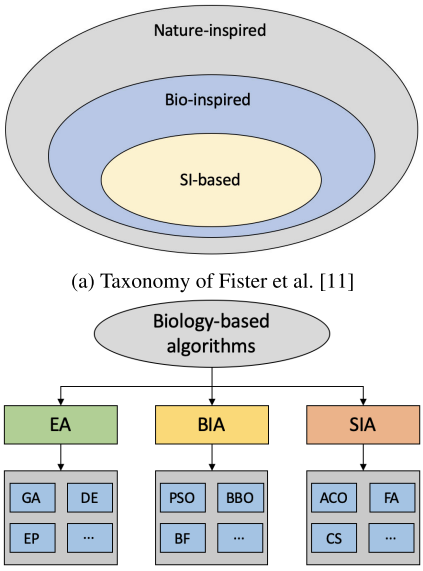
Researchers have a natural desire to provide theoretical basis for NIOs. As a pioneer work, Holland [23] used Schema theory to theoretically explain the genetic algorithm. However, the attempt for such a classical NIO was criticized as being an inadequate theoretical basis [24]. Even worse, it is hard to analyse the computational complexity for any NIO due to the difficulties in stochastic search and fitness landscape analysis [25]. So we take extensively measurements of the cost function evaluation counts instead of the computational complexity estimation.

F. ORGANIZATION

Section II gives the taxonomy and the unified framework for NIOs to be described. We provide some technical notations and abbreviations to facilitate further description and discussion. In section III, the essence of each NIO algorithms is epitomized. This is followed by the evaluation (section IV) and parameter optimization (section V) of the NIO algorithms. Section VI concludes the paper finally.

II. OVERVIEW

In this paper, we focus on the NIO algorithms which are based on various animal habits or nature processes such as foraging and breeding. Table 1 lists these NIO algorithms in descending order of published year. The first reference for each algorithm is the earliest published literature of the NIO algorithm.



(a) Taxonomy of Fister et al. [11]

(b) Taxonomy of Siddique [12]

FIGURE 2. Taxonomy of Fister *et al.* [11] and Siddique [12].

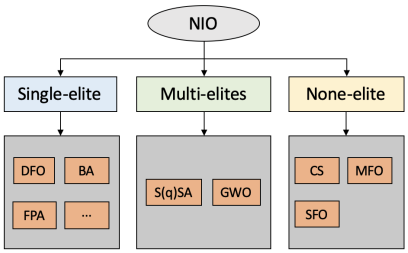


FIGURE 3. Proposed taxonomy.

A. TAXONOMY

It is a tremendous challenge to define a widely accepted taxonomy for nature inspired optimization algorithms due to diversity of the nature. Fister *et al.* [11] classified the NIO algorithms into swarm intelligent (SI) based algorithms, bio-inspired but not swarm intelligence based algorithms, physics and chemistry based algorithms, and other algorithms. Siddique [12] classified the biology-based algorithms into evolutionary algorithms (EA), bio-inspired algorithms (BIA) and swarm intelligence-based algorithms (SIA).

We visualize the taxonomy mentioned previously in Fig.2. As illustrated in Fig.2a, the bio-inspired algorithm is a type of nature inspired algorithm, and the SI-based algorithms belong to the category of bio-inspired algorithm. It is impossible to exhibit all NIO algorithms in Fig.2b due to the space limitation. We do not fully enumerate all algorithms. Only some representative algorithms are exhibited to save space. And we use "..." to represent those NIO algorithms not listed here.

In this paper, we add a taxonomy by classifying the NIO algorithms into single-elite, multi-elite, and non-elite algorithms, based on the number of elites they used in the algorithm. An elite is defined as the individual in a swarm who has the highest fitness, while multi-elites are defined as the

TABLE 1. The collected NIO algorithms.

NIO	Abbr.	Inspiration	Year	Main literature
Sailfish Optimizer	SFO	Sailfish hunting.	2019	[26]
Black Widow Optimization Algorithm	BWOA	Black widow breeding.	2019	[7]
Squirrel Search Algorithm	S(q)SA	Squirrels foraging	2018	[27]
Salp Swarm Algorithm	S(a)SA	Salp swarm foraging	2017	[28] [29]
Whale Optimization Algorithm	WOA	Whale foraging	2016	[30] [31] [32] [33]
Root Tree Optimization	RTO	Tree roots growth	2016	[34] [35]
Ant-lion Optimizer	ALO	The ant-lion hunting process	2015	[36] [37] [38]
Moth Flame Optimization	MFO	Moths navigation mechanism	2015	[39] [31] [40]
Water Wave Optimization	WWO	The shallow water theory	2014	[41]
Grey Wolf Optimizer	GWO	Grey wolf pack hunting	2014	[1] [42] [43] [44] [45]
Dispersive Files Optimisation	DFO	Dispersive flies over foods	2014	[46] [47]
Krill Herd	KH	Krill herd foraging behaviour	2012	[48] [49] [50] [51] [52]
Flower Pollination Algorithm	FPA	Flowers pollination behaviour	2012	[53] [54] [55]
Fruit Fly Algorithm	FOA	Fruit fly foraging	2011	[56] [57]
Bat Algorithm	BA	Bats locating skills through sounds	2010	[58] [59] [60] [61] [62]
Firefly Algorithm	FA	Fireflies chasing the lights	2009	[63] [64] [20] [65]
Cuckoo Search	CS	Cuckoos breeding behaviour	2009	[66] [67] [68] [69] [70]

TABLE 2. Other newly proposed NIO algorithms.

NIO	Abbr.	Inspiration	Year	Main literature
Barnacles Mating Optimizer	BMO	Barnacles mating behavior.	2019	[71]
Poor and rich optimization algorithm	PRO	Efforts of the poor and the rich people to achieve wealth.	2019	[72]
Pathfinder algorithm	PFA	The leadership hierarchy of swarms to find best food area or prey.	2019	[73]
Falcon Optimization Algorithm	F(a)OA	The hunt behavior of falcons.	2019	[74]
Meerkats-inspired Algorithm	MEA	Meerkats social organizations.	2018	[75]
Farmland fertility	FF	Farmland fertility in nature.	2018	[76]
Coyote Optimization Algorithm	COA	The canis latrans species.	2018	[77]
Owl Optimization Algorithm	OOA	The decoy behavior of owls	2018	[78]
Interactive search algorithm	ISA	Modifies and combines iPSO and TLBO.	2018	[79]
Cheetah Based Optimization Algorithm	CBA	The social behavior of cheetah.	2018	[80]
Galactic Swarm Optimization	GCO	The motion of stars, galaxies and superclusters of galaxies.	2015	[81]

best several individuals. Elites guide other individuals during the course of searching for optima. The classification of the collected NIO algorithms in this paper is illustrated in Fig.3.

B. UNIFIED FRAMEWORK

It is beneficial for implementation to develop a unified framework in terms of algorithm structure. Over the past several years, vast literature provides valuable insights into how to understand the employed strategies and what the general characteristic of the NIOs is ([82] and [83]). The framework we present here inevitably uses these concepts as the basis of the description and reconstructed them to form more detailed principles and practices for scientific research or programming.

Almost all NIO algorithms mentioned in this paper share a common flowchart as Fig.4. In the framework, each NIO process can be divided into four steps. The widely used stopping criteria encompass fitness evaluation bound, generation bound, and specified precision.

Furthermore, we describe the detailed procedures of each step in Table 3. Each sub-procedure is labelled in ascending order. We list all contained sub-procedures for each NIO algorithm in Table 4. It should be noted that the steps in Table 4 are optional and few NIO algorithms contain all sub-procedures. More details for each NIO algorithm will be described in the next section.

C. TECHNICAL NOTATIONS

To facilitate further description and discussion, we give some technical notations and abbreviations in Table 5. Also a symbol list is provided in Table 6. With Table 6 in hand, we give a pseudo code version of the unified framework as Algorithm 1.

TABLE 3. Specification of steps and sub-steps in the NIO algorithms.

Step	Sub-steps	Description
Step 1	1-1	Uniformly initialize the swarm in search domain
	1-2	Sort solutions by CFV value
Step 2	2-1	Update the best solution in the population
	2-2	Update the best n solutions in the population.
Step 3	3-1	Generate new solution guided by nearby neighbours
	3-2	Generate new solution guided by the elites
	3-3	Generate new solution from previous position
	3-4	Randomly generate new solution
	3-5	Generate new solution guided by a randomly selected individual
Step 4	4-1	Replace the old solution anyway
	4-2	Replace the old solution if the new solution is better
	4-3	Accept new solutions with a certain probability if the new solution is worse

TABLE 4. Steps or sub-steps contained in collected NIO algorithms.

NIO	Step 1		Step 2		Step 3					Step 4		
	1-1	1-2	2-1	2-2	3-1	3-2	3-3	3-4	3-5	4-1	4-2	4-3
SFO	✓	✓	✓	-	-	✓	✓	✓	-	-	✓	-
BWOA	✓	✓	-	-	-	-	✓	-	-	-	✓	-
S(q)SA	✓	✓	-	✓	-	✓	✓	✓	-	-	✓	-
RTO	✓	✓	✓	-	-	✓	✓	-	✓	✓	-	-
S(a)SA	✓	-	✓	-	✓	-	✓	-	-	-	✓	-
WOA	✓	-	✓	-	-	✓	✓	-	✓	✓	-	-
WWO	✓	-	✓	-	-	✓	✓	-	-	✓	✓	-
ALO	✓	✓	✓	-	-	✓	✓	-	✓	-	✓	-
MFO	✓	✓	✓	-	-	✓	✓	-	-	✓	-	-
GWO	✓	-	-	✓	-	✓	✓	-	-	✓	-	-
DFO	✓	-	✓	-	✓	✓	✓	-	-	-	✓	✓
KH	✓	-	✓	-	✓	✓	✓	✓	-	✓	-	-
FPA	✓	-	✓	-	-	✓	✓	-	✓	✓	✓	-
FOA	✓	-	✓	-	-	✓	-	-	-	-	✓	-
BA	✓	-	✓	-	-	✓	✓	-	-	-	✓	✓
FA	✓	✓	✓	-	-	✓	✓	-	-	✓	-	-
CS	✓	-	✓	-	-	-	-	✓	-	-	✓	✓

TABLE 5. Technical notations and abbreviations.

Notation or abbreviation	Description
CFV	Cost Function Value
Gen	Generation
D, d, Dim	Dimension of the search space
$U(a, b)$	Uniform distribution in $[a, b]$
$L(\lambda)$	Lévy distribution
$N(\mu, \sigma^2)$	Normal distribution
$Lévy(\lambda)$	Lévy flight

III. INTENSIVE REVIEW OF NIO ALGORITHMS

A. SAILFISH OPTIMIZER

SFO is inspired by the attack-alternation strategy of the group hunting sailfishes (predators) and a school of sardines (prey). This strategy saves the energy of the predators while other predators are injuring the prey.

In the SFO, catching prey occurs when the sardine becomes fitter than its corresponding sailfish. Then the position of sailfish substitutes with the latest position of the hunted sardine to increase the chance of hunting new prey. The elite sailfish is assumed to be the best position among sailfishes. A sailfish updates its position according to current position, the position of the elite sailfish, and the position of the injured sardine, which is formulated as:

$$\begin{aligned} X\_sf_{i,d}^{t+1} &= X_{best\_sf}^{t} - \lambda \\ &\quad \times (r \times \frac{X_{best\_sf}^{t} + X_s}{2} - X\_sf_{i,d}^{t}), \\ \lambda &= 2 \times r \times PD - PD, \quad PD = 1 - \frac{N_{sf}}{N_{sf} + N_s}, \\ r &\sim U(0, 1), \end{aligned} \tag{1}$$

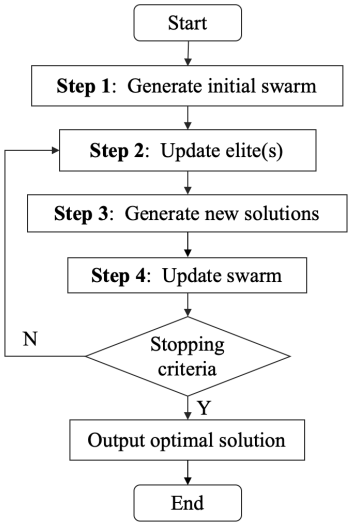


FIGURE 4. Flowchart of the NIOs.

TABLE 6. Symbol definitions.

Symbol	Description	Data type
t	Generation	Integer
T	Number of iterations	Integer
n	Swarm size	Integer
X	Position of an individual in n dim	Vector
X^t	Individual position in generation t	Vector
X_{best}	The best individual	Vector
X_{nghb_b}	The best individual around its neighbour	Vector
$f(X)$	Fitness of an individual	Real
b_u	Upper bound of search domain	Vector
b_l	Lower bound of search domain	Vector
Elite(s)	Optimal individual(s) in the swarm	Vector
Leader	First n individuals in a certain order	Vector

where PD (prey density) is used to control the number of sardines in each iteration; X_sf stands for the position of sailfish; X_{best_sf} stands for the best individual of the sailfish group; N_{sf} stands for the number of sailfishes; N_s stands for the number of sardines.

Sardines update their position based on their position and the AP (attack power):

$$X\_s_{i,d}^{t+1} = r \cdot (X_{best\_s,d}^{t} - X\_s_{i,d}^{t} + AP),$$
$$AP = A_0 \times (1 - 2 \times t \times \epsilon), \quad r \sim U(0,1), \tag{2}$$

where ϵ is the coefficient that reduces the attack power AP linearly from A_0 to 0. The position of sailfish substitutes with the position of the hunted sardine:

$$X\_sf_{i,d}^{t} \leftarrow X\_s_{i,d}^{t} \quad if(f(X\_s_{i,d}^{t}) > f(X\_sf_{i,d}^{t})). \tag{3}$$

Algorithm 1 Pseudo Code of the Unified NIO Framework

Input: The optimization problem with the search domain

Output: The optimal solution and its fitness

begin

Define $f(X)$, $X = (x_1, x_2, \cdots, x_D)$
Generate initial swarm X within the search domain
Set algorithmic parameters
Evaluate the swarm with $f(X)$, and obtain elite(s)
while not meeting stopping criteria **do**
  Generate a new solution x_i^{t+1} from x_i^t and elite(s)
  Re-evaluate the swarm with $f(X)$, and obtain elite(s)
end while
Output the optimal solution and its CFV.

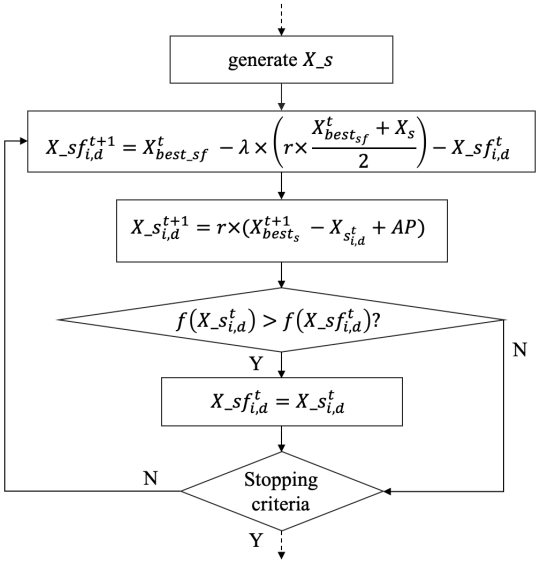


FIGURE 5. The flowchart of SFO.

Fig. 5 illustrates the SFO algorithm. The sailfishes intensify the search around the best solution so far, while the sardines expand the search space. SFO is a single-elite algorithm.

B. BLACK WIDOW OPTIMIZATION ALGORITHM

BWOA is inspired by the mating behavior of black widow spiders. The female spider consumes the male during mating. After eggs are hatched, the offspring engage in sibling cannibalism.

In each generation of BWOA, the best nr individuals are selected as parents X_{p1}, X_{p2} according to procreating rate. Each pair of children can be formulated as:

$$y_1 = \alpha \times X_{p1} + (1 - \alpha) \times X_{p2},$$
$$y_2 = \alpha \times X_{p2} + (1 - \alpha) \times X_{p1},$$
$$\alpha \sim U(0,1), \tag{4}$$

y_1 and y_2 are children. Since each pair of parents produces several pairs of children in a generation, the population size

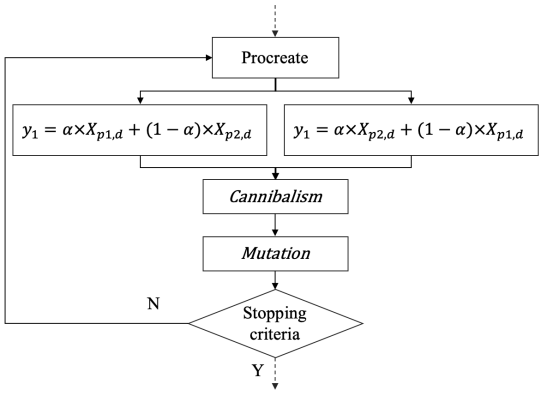


FIGURE 6. The flowchart of BWOA.

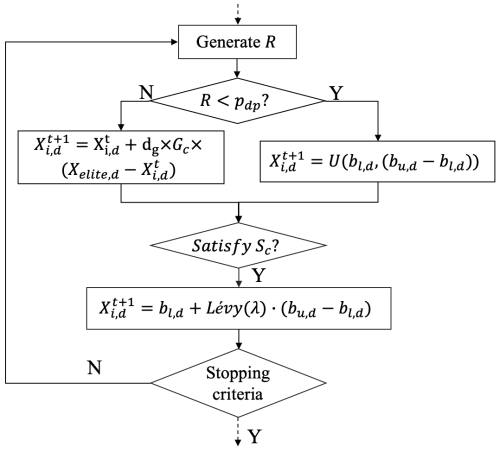


FIGURE 7. The flowchart of S(q)SA.

increases after mating. The children and mother are put to an array and sorted by their fitness value. Then the population size is restored through cannibalism or competition. Finally, individuals are randomly selected for mutation according to the mutation rate.

Fig. 6 illustrates the BWOA algorithm. In BWOA, the optimal individual does not impact the update of other individuals in the population, hence BWOA is a non-elite algorithm.

C. SQUIRREL SEARCH ALGORITHM

S(q)SA is inspired by the foraging strategy and gliding mechanism of flying squirrels. The flying squirrels do not have the ability to fly. Instead they slide to save energy.

In autumn, the squirrels search for food resources by gliding from one tree to the others. They eat acorns in oak trees immediately upon finding them. They also search for the hickory tree as optimal food source. Storing hickory nuts helps them survive in the winter.

In S(q)SA, there are four elites including one squirrel on the hickory tree and three squirrels on oak trees. Three types of gliding correspond to exploitation. Specifically, flying squirrels which are on oak trees may move towards the hickory tree. Flying squirrels on normal trees move towards oak trees or the hickory tree. The glidings can be formulated

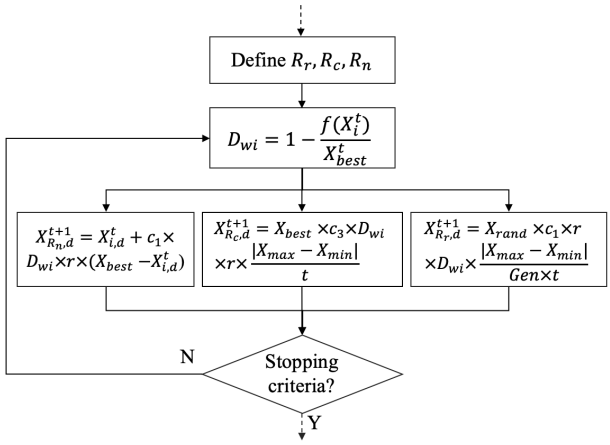


FIGURE 8. The flowchart of RTO.

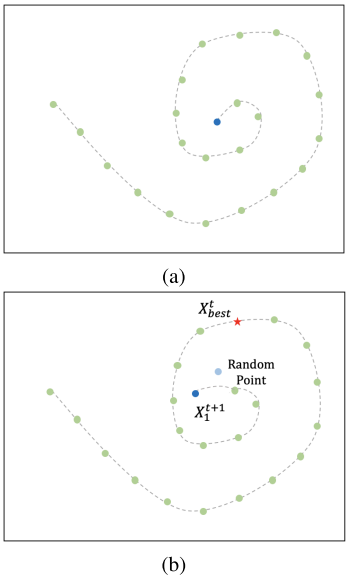


FIGURE 9. The salp chain and the update strategy of the leader.

as:

$$X_{i,d}^{t+1} = \begin{cases} X_{i,d}^{t} + d_g \cdot G_c \cdot (X_{elite,d} - X_{i,d}^{t}), & r \geq p_{dp}, \\ U(b_{l,d}, (b_{u,d} - b_{l,d})), & otherwise, \end{cases} \quad (5)$$

where p_{dp} is the predator presence probability, G_c is the gliding constant, and d_g is a random gliding distance. An exploration operation is conducted mimicking the random search at the end of winter:

$$X_{i,d}^{t+1} = b_{l,d} + Lévy(\lambda) \cdot (b_{u,d} - b_{l,d}), \ \lambda = 1.5, \quad (6)$$

The four elites are updated in each iteration by reordering fitness among the population.

Fig. 7 is the flowchart of the S(q)SA.

D. ROOTED TREE OPTIMIZATION

Tree roots play an essential role in the processes of plant growth, especially in the search for water resources. The roots are prone to grow in the water-rich place. Rooted Tree Optimization (RTO) mimics the behaviour of

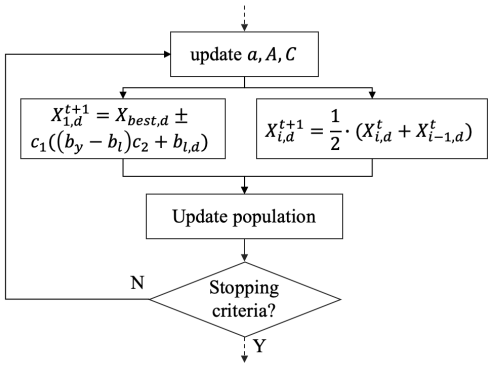


FIGURE 10. The flowchart of S(a)SA.

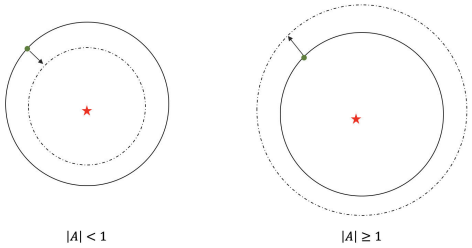


FIGURE 11. The A value in WOA/GWO (WOA/GWO performs exploration when $|A| > 1$).

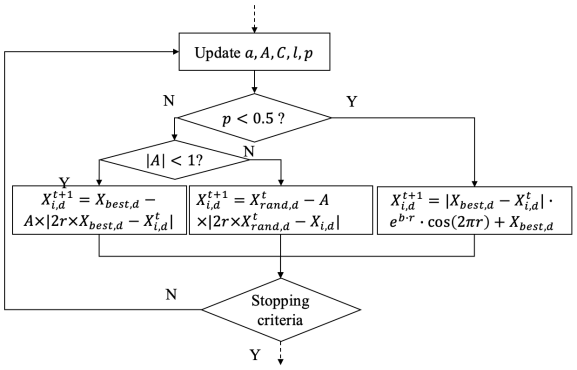


FIGURE 12. The flowchart of WOA.

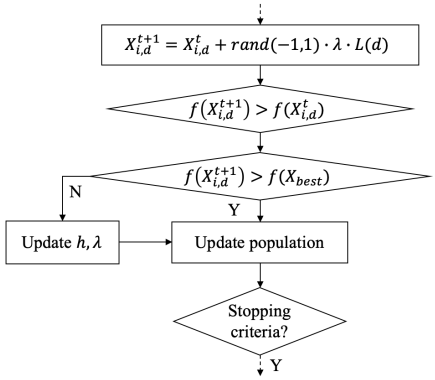


FIGURE 13. The flowchart of WWO.

tree roots. RTO groups its root (the swarm) into three sub-swarms. The sub-swarm far from the water source R_r will randomly choose the location to grow, which is formulated

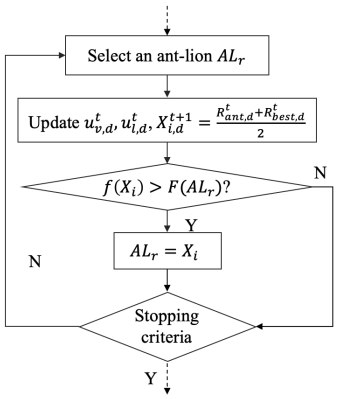


FIGURE 14. The flowchart of ALO.

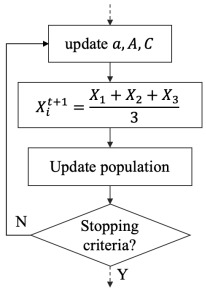


FIGURE 15. The flowchart of GWO.

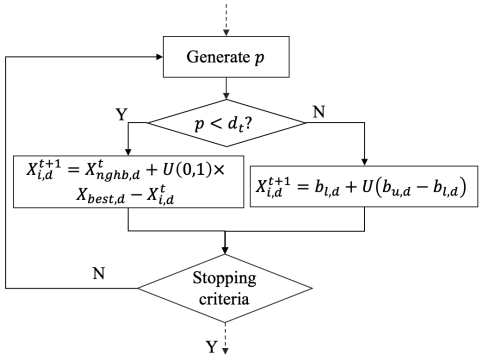


FIGURE 16. The flowchart of DFO.

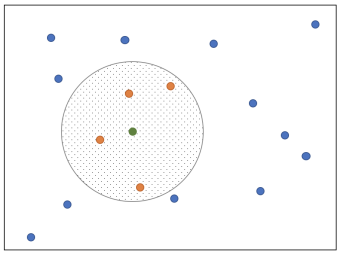


FIGURE 17. Determining neighbourhood by the sensing distance in KH.

as

$$X_{i,d}^{t+1} = X_{rand} + c_1 \times D_{wi} \times r \times \frac{|X_{max} - X_{min}|}{Gen}, \quad r \sim U(0,1), \quad c_1 \sim U(0,1). \tag{7}$$

The sub-swarm not far from the water source (termed as continuous roots) R_c will move forward from the previous

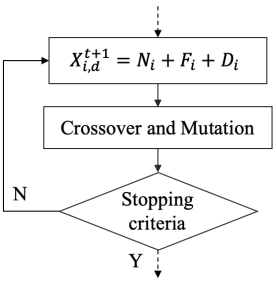


FIGURE 18. The flowchart of KH.

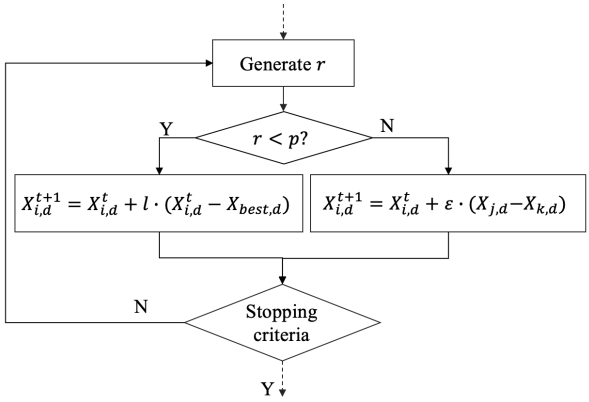


FIGURE 19. The flowchart of FPA.

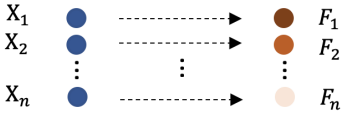


FIGURE 20. The moths and flames in MFO.

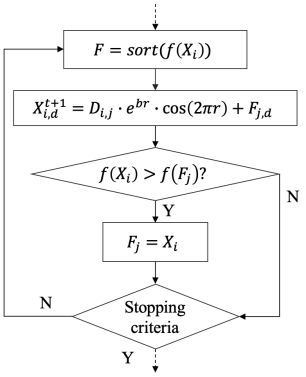


FIGURE 21. The flowchart of MFO.

generation to the best root so far, which is formulated as

$$X_{i,d}^{t+1} = X_{best} + c_3 \times D_{wi} \times r \times \frac{|X_{max} - X_{min}|}{Gen \times t}, \quad r \sim U(0,1), \quad c_3 \sim U(0,1). \tag{8}$$

The roots which are near to the water source R_n update their positions stochastically around the best root, which is formulated as

$$X_{i,d}^{t+1} = X_{i,d}^{t} + c_1 \times D_{wi} \times r \times (X_{best} - X_{i,d}^{t}), \quad r \sim U(0,1), \tag{9}$$

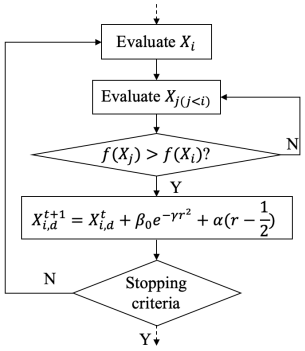


FIGURE 22. The flowchart of FA.

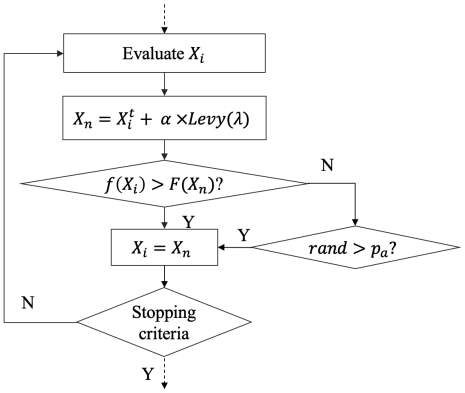


FIGURE 23. The flowchart of CS.

where D_{wi} is the wetness degree of each root, which can be calculated as

$$D_{wi} = 1 - \frac{f(X_i^t)}{X_{best}^t}. \tag{10}$$

RTO is a single-elite algorithm. It relates the update rules of R_c and R_n to the elite X_{best}. Fig.8 illustrates the RTO algorithm.

E. SALP SWARM ALGORITHM

S(a)SA is inspired by swarming behaviours of the salp chains (see Fig.9a). The leader which is the salp at the front of the chain guides the followers to search foods in the sea. S(a)SA updates the position of the leader in the following way:

$$X_{1,d}^{t+1} = \begin{cases} X_{best,d} + c_1((b_{u,d} - b_{l,d})c_2 + b_{l,d}), & c_3 \geq 0.5, \\ X_{best,d} - c_1((b_{u,d} - b_{l,d})c_2 + b_{l,d}), & c_3 < 0.5, \\ c_1 = 2e^{-(\frac{4t}{T})^2}, \quad c_2, c_3 \sim U(0,1), \end{cases} \tag{11}$$

where c_1 is a coefficient used to balance the exploration and the exploitation. The leader tends to move toward the best found solution, as shown in Fig.9b. The followers are then updated using the equation:

$$X_{i,d}^{t+1} = \frac{1}{2} \cdot (X_{i,d}^{t} + X_{i-1,d}^{t}). \tag{12}$$

The new position of each follower is between the current position and the position of its front neighbour. Therefore,

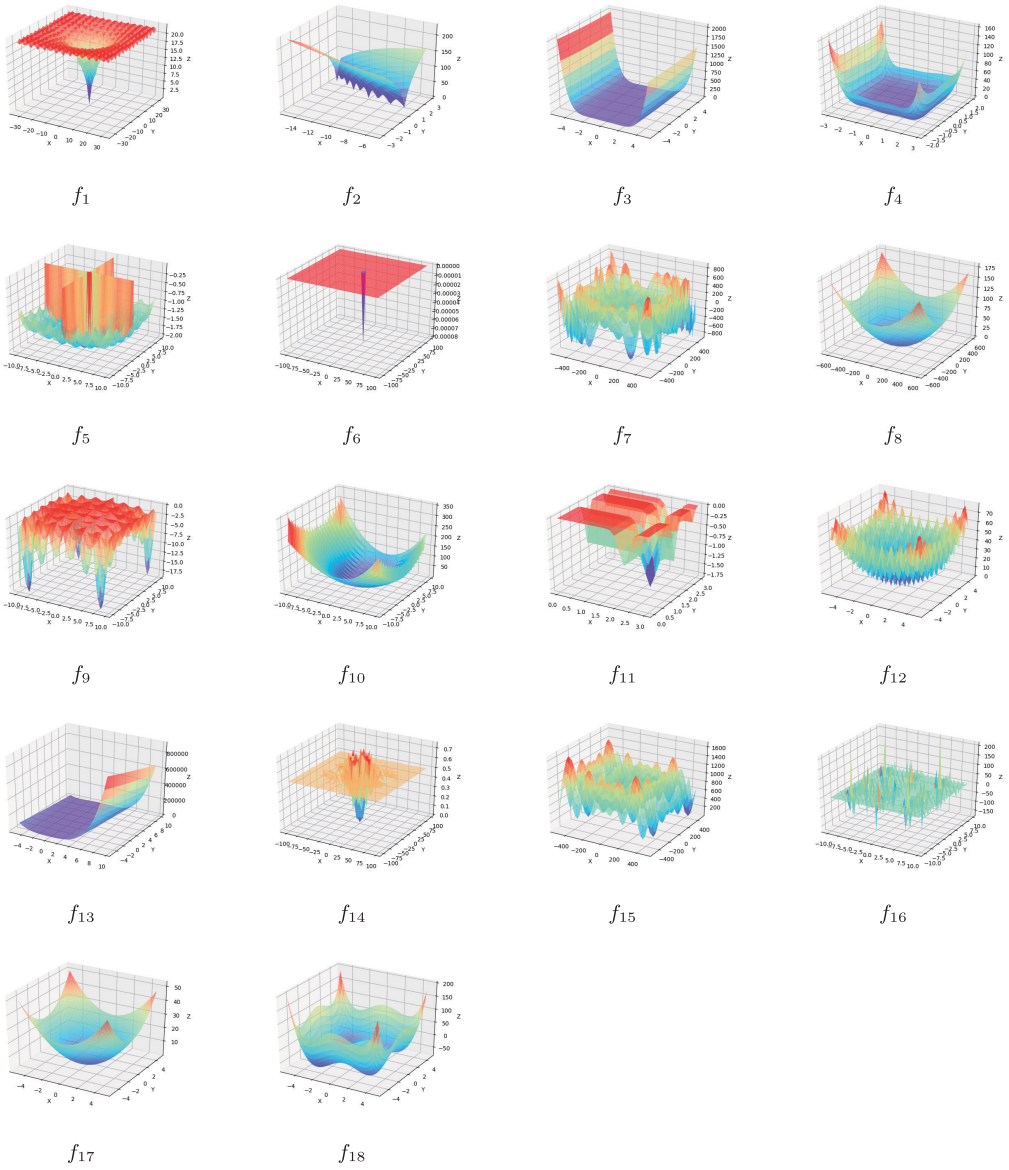


FIGURE 24. The images of the 2-D benchmark functions defined in Table 9.

the salp swarm keeps the chain shape during the course of search. The flowchart of S(a)SA is presented in Fig. 10

F. WHALE OPTIMIZATION ALGORITHM

WOA is inspired by the foraging behaviour of the humpback whales which is called bubble-net feeding method. The shrinking encircling and spiral attacking are two essential mechanisms in the bubble-net feeding method. The mechanisms can be mathematically modelled as:

$$\begin{aligned} X_{i,d}^{t+1} &= X_{best,d} - A \times |2r \times X_{best} - X_{i,d}^{t}|, \\ A &= 2a \cdot r - a, \quad r \sim U(0,1), \end{aligned} \tag{13}$$

for shrinking encircling where a decreases linearly from 2 to 0 with iteration, and

$$\begin{aligned} X_{i,d}^{t+1} &= |X_{best,d} - X_{i,d}^{t}| \cdot e^{b \cdot r} \cdot \cos(2\pi r) + X_{best,d}, \\ r &\sim U(-1,1), \end{aligned} \tag{14}$$

for spiral attacking, where b is a constant defining the shape of the spiral. The shrinking encircling and spiral attacking are performed randomly with the probability of 0.5 in each iteration. The exploration process of WOA can be modelled as:

$$X_{i,d}^{t+1} = X_{rand,d}^{t} - A \times |2r \times X_{rand,d}^{t} - X_{i,d}^{t}|. \tag{15}$$

The exploration happens when the individual is far away from the best solution ($|A| > 1$), as illustrated in Fig. 11.

WOA is a single-elite NIO algorithm. The flowchart of WOA is shown in Fig.12.

G. WATER WAVE OPTIMIZATION

WWO is inspired by the shallow water wave theory. Each individual of WWO is a wave with position X, wave length λ and amplitude h. There are three operations in WWO

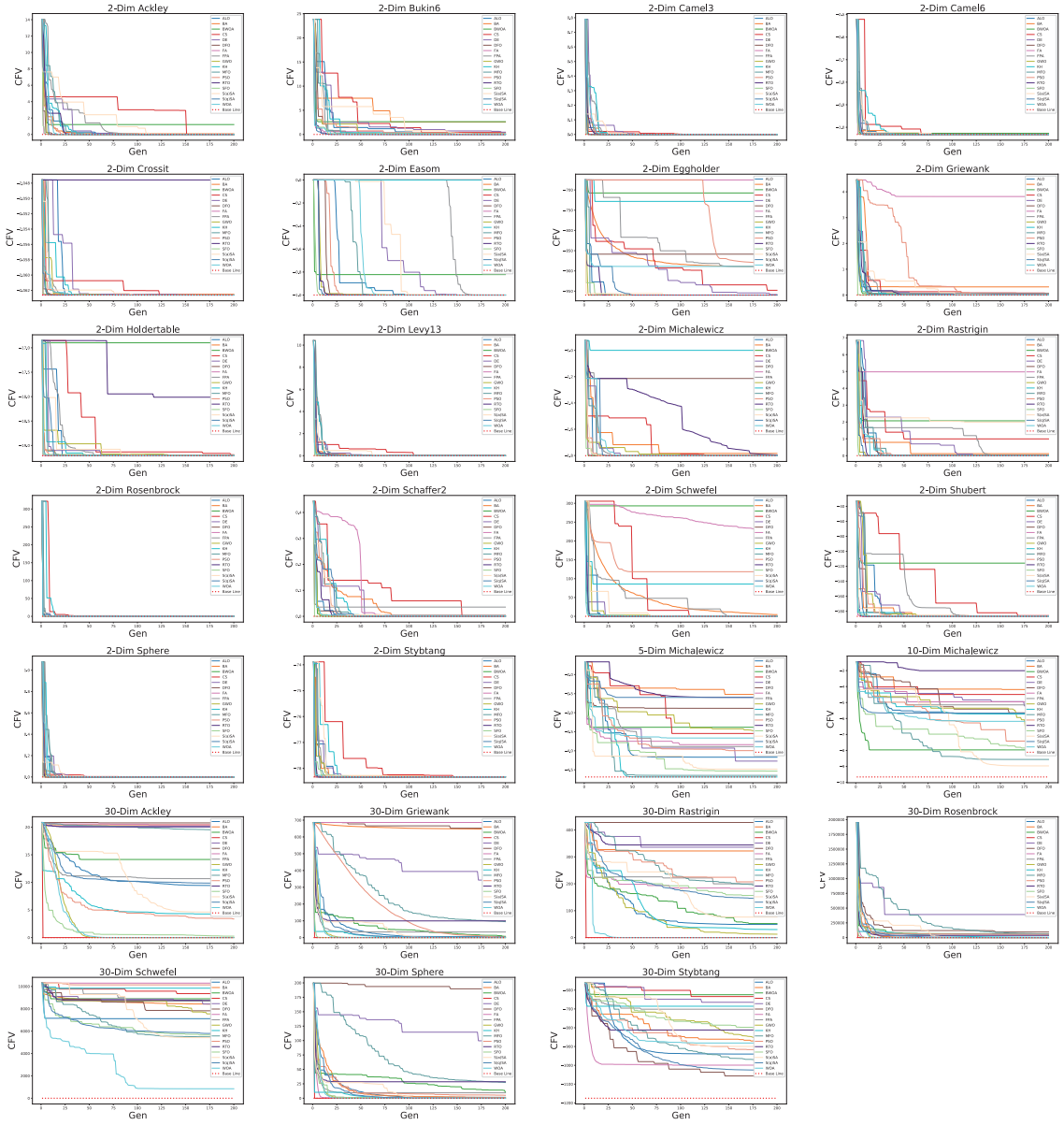


FIGURE 25. Convergence curves.

including Propagation, Refraction, and Breaking. Propagation creates a new position by shifting each dimension d of the original wave as:

$$X_{i,d}^{t+1} = X_{i,d}^{t} + r \cdot \lambda \cdot D_d, \quad r \sim U(-1, 1), \tag{16}$$

where D_d is the length of the d^{th} dimension of the search space. If the new position is better than the old one, it replaces the old one with itself, and thereafter the new solution updates its amplitude as h_{max}. Otherwise, the wave length keeps unchanged, but its height h is decreased by one. WWO performs refraction on waves whose heights decrease to zero, and calculates the new position:

$$X_{i,d}^{t+1} = r \cdot (\frac{X_{best,d}}{2}, \frac{|X_{best,d} - X_{i,d}^{t}|}{2}), \quad r \sim N(\mu, \theta). \tag{17}$$

After refraction, the amplitude of the wave is set to h_{max}, and the wave length is set to:

$$\lambda' = \lambda \frac{f(X^t)}{f(X^{t+1})}. \tag{18}$$

When WWO finds an enough optimal solution which exceeds the threshold value β, the wave breaks. In breaking, WWO performs local search near the optimal solution:

$$X_{i,d}^{t+1} = X_{i,d}^{t} + r \cdot \beta \cdot D_d, \quad r \sim N(0, 1). \tag{19}$$

By the end of each iteration, the wavelength of each wave is updated as follows:

$$\lambda \leftarrow \lambda \cdot \alpha^{-\frac{f(X) - f_{min} + \epsilon}{f_{max} - f_{min} + \epsilon}}, \tag{20}$$

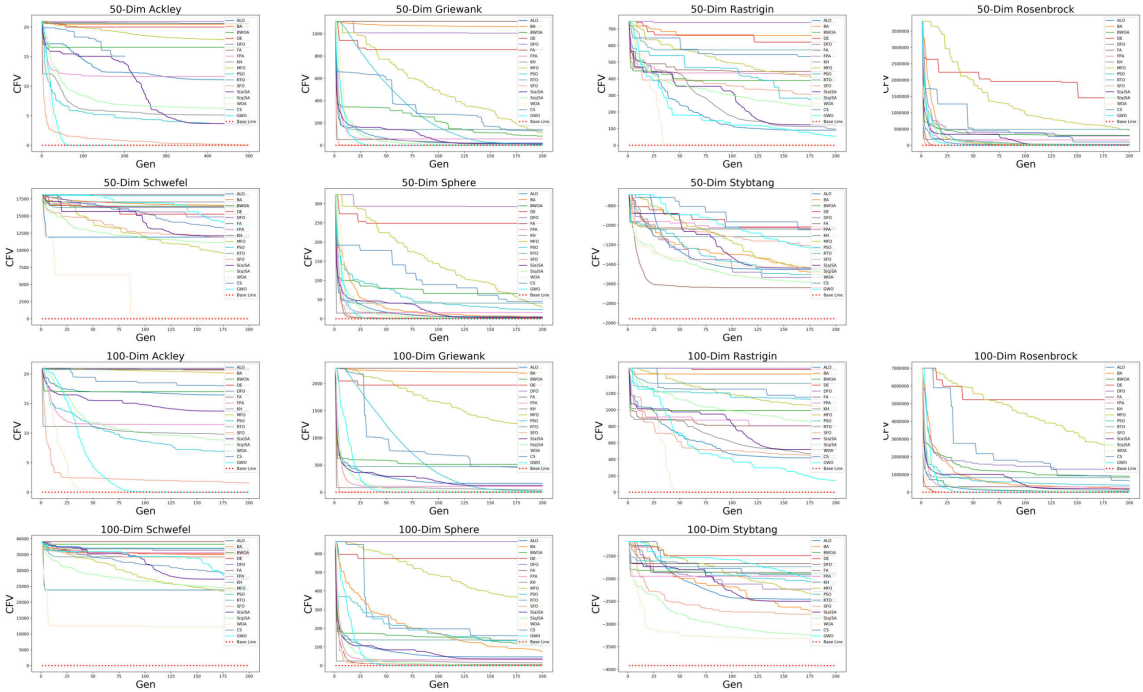


FIGURE 26. Convergence curves of the 50D and 100D functions.

where α is the wavelength reduction coefficient and ϵ is a positive infinitesimal. f_{max} and f_{min} are the maximum and minimum CFV in current swarm.

Fig. 13 is the flowchart of the WWO. WWO is a single-elite NIO. Only the individual with best fitness has impact on other individuals.

H. ANT LION OPTIMIZATION

ALO mimics the hunting mechanism of ant lions. It maintains two fitness matrix for antlions and ants. The global search is implemented mimicking the random walk of ants. Each ant randomly selects a trap built by an antlion, and walks around it:

$$R^t_{ant,d} = \frac{(X^t_{i,d} - b_{l,d}) \cdot (b_{u,d} - v^t_{l,d})}{v^t_{u,d} - b_{l,d}}) + b_{l,d}, \tag{21}$$

where $v^t_{l,d}$ and $v^t_{u,d}$ are bound of random walk at d^{th} dimension in the t^{th} iteration. ALO performs local search by narrowing the range of random walk:

$$v^{t+1}_{l,d} = \frac{v^t_{l,d}}{I}, \quad v^{t+1}_{u,d} = \frac{v^t_{u,d}}{I}, \tag{22}$$

where I is the ratio of constriction. ALO is a NIO with a single elite. The new solution is obtained by averaging the current random walk and the random walk around the best solution founded:

$$X^{t+1}_{i,d} = \frac{R^t_{ant,d} + R^t_{best,d}}{2}. \tag{23}$$

If an ant finds a position with better fitness, the ant lion updates its position correspondingly.

ALO is outlined in Fig. 14.

I. GREY WOLF OPTIMIZER

Inspired by the social hierarchy and hunting behaviour of grey wolves, GWO selects three leaders in each iteration. The fittest wolf is X_α. The second and third are X_β and X_γ, respectively. Other wolves are represented by X_ω which follow the leaders during the hunting. Encircling prey is modelled as:

$$\begin{aligned} X_I &= X^t_{I,d} - A \times |2r \times X^t_{I,d}| - X^t_{i,d}, \\ I &= (\alpha, \beta, \gamma), \end{aligned} \tag{24}$$

where X_I represents the impact of the elites on X^t_i. The definitions of A is same as that in WOA. In hunting, GWO updates the positions of wolves toward:

$$X^{t+1}_{i,d} = \frac{X_{\alpha,d} + X_{\beta,d} + X_{\delta,d}}{3}. \tag{25}$$

The parameter A determine if the GWO performs global search or local search. If an individual is near the leaders ($|A| \in [-1, 1]$), then it performs local search. X_α, X_β, and X_δ have even impact on other individuals in the population. Otherwise, it performs global search, as illustrated in Fig. 11.

The leader and the elite are identical in GWO. Fig.15 illustrates the GWO algorithm. GWO is a multi-elite NIO algorithm.

J. DISPERSIVE FLIES OPTIMIZATION

DFO mimics the swarming behaviour of flies over food sources. The flies disperse if they are disturbed. They return and form a swarm again immediately after the disturb is over. DFO is similar to the FPA algorithm. Both of them use a probability parameter to balance the global search and

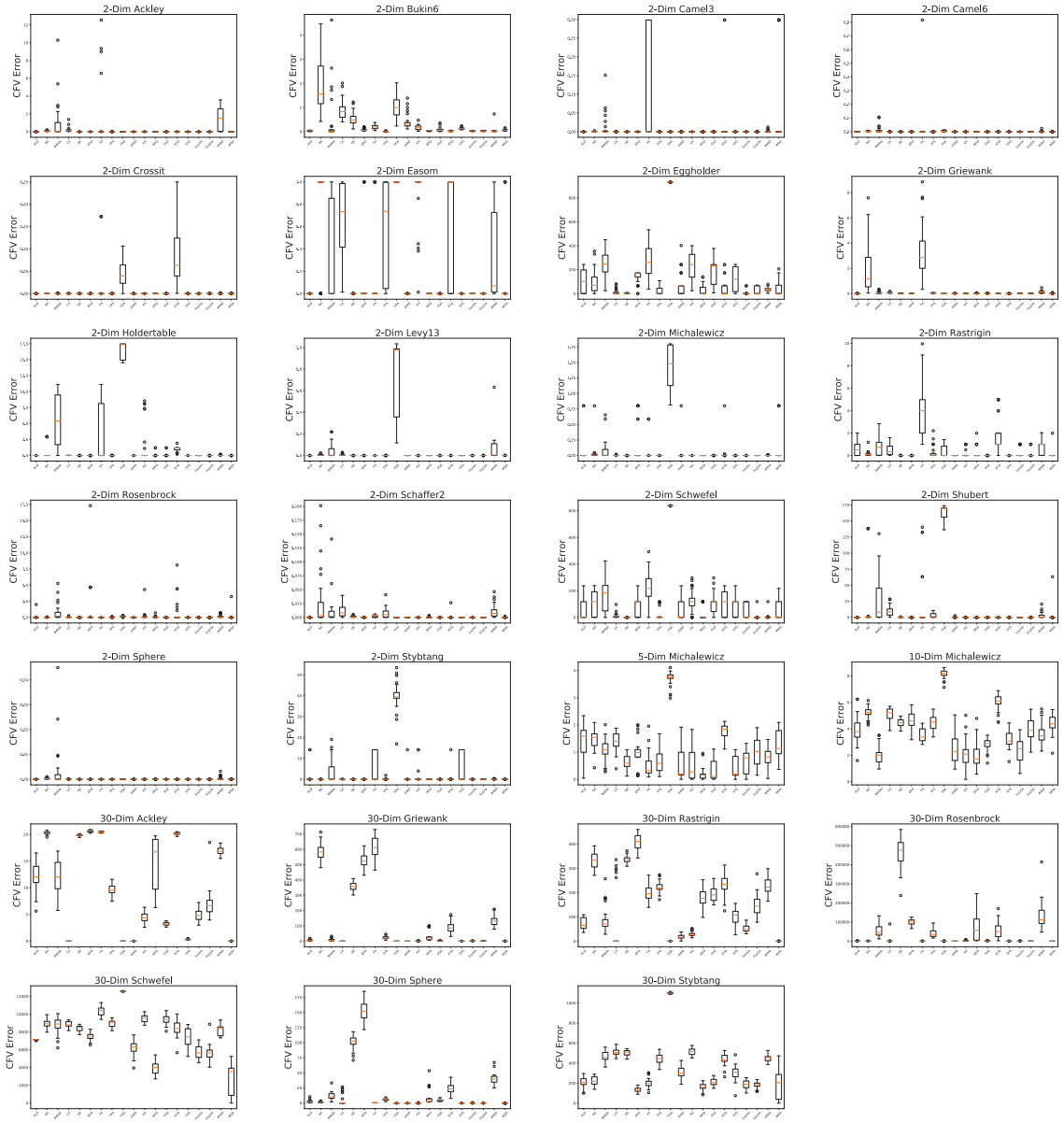


FIGURE 27. Box plots of the CFV errors.

the local search. Specifically, the parameter in DFO is the disturbance parameter d_t.

DFO maintains a ring topology. The local search is performed under the guidance of the left neighbour, the right neighbours, and the best individual:

$$\begin{aligned} X_{i,d}^{t+1} &= X_{nghb\_b,d}^{t} + r \cdot (X_{best,d} - X_{i,d}^{t}), \\ X_{nghb\_b,d} &= \min(f(X_{left,d}), f(X_{right,d})), \\ r &\sim U(0,1). \end{aligned} \tag{26}$$

DFO uses a variable d_t to simulate the disturb. When a random number r is smaller than d_t, DFO relocates the individual as:

$$X_{i,d}^{t+1} = b_{l,d} + r \cdot (b_{u,d} - b_{l,d}), \quad r \sim U(0,1), \tag{27}$$

which is a uniformly random position in the search domain.

DFO is a single-elite NIO algorithm. The flowchart of DFO is shown in Fig. 16.

K. KRILL HERD

KH mimics the krill herd in response to specific biological and environmental processes. Three actions including foraging activity, movement induced by other krill individuals, and random diffusion are considered. The search dynamic is modelled as the following Lagrangian model:

$$dX_i/dt = N_i + F_i + D_i, \tag{28}$$

where N_i is the motion induced by other krill individuals; F_i is the foraging motion, and D_i is the physical diffusion of the i^{th}

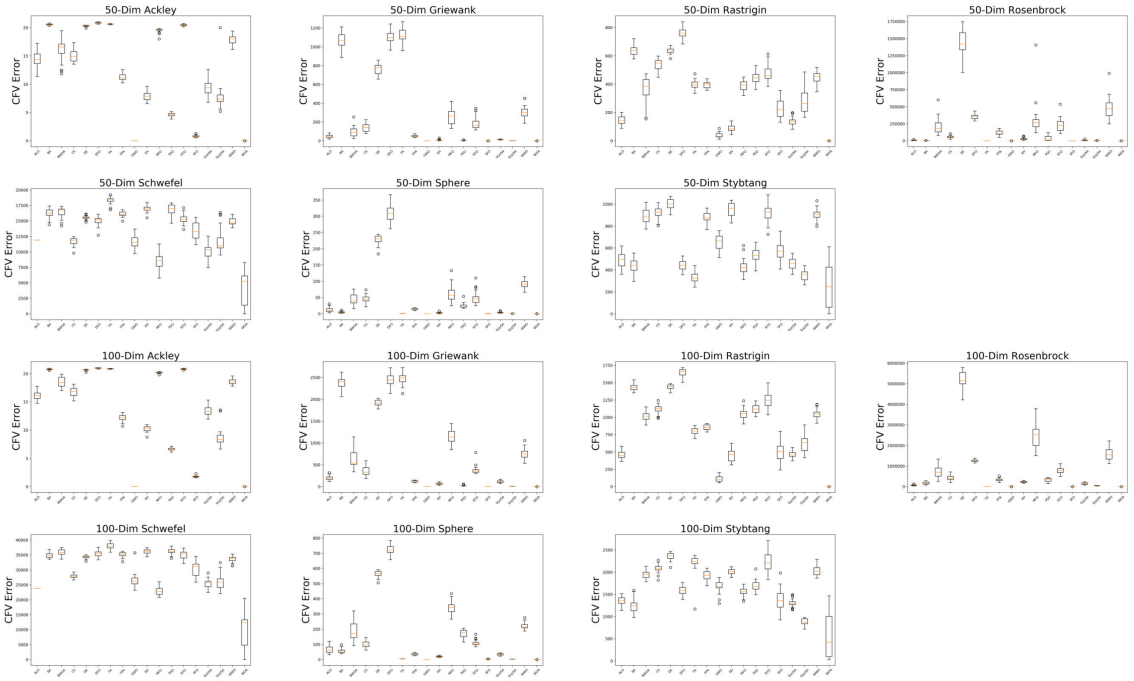


FIGURE 28. Box plots of the 50D and 100D functions.

krill individual. Each krill herd individual update its position by:

$$X_{i,d}^{t+1} = X_{i,d}^{t} + dX_i/dt. \tag{29}$$

The movement guided by other krills is modelled as:

$$\begin{aligned} N_i^{t+1} &= N^{max} \cdot \alpha_i + \omega_n \cdot N_i^t, \\ \alpha_i &= \alpha_i^{local} + \alpha_i^{target}, \end{aligned} \tag{30}$$

where N^{max} is the maximum speed, ω_n is the inertia weight, α_i^{local} is the neighbour effect, α_i^{target} is the target direction effect. The foraging activity is modelled as:

$$F_i^{t+1} = V_f \cdot \beta_i + \omega_f \cdot F_i^t, \quad \beta_i = \beta_i^{food} + \beta_i^{best}, \tag{31}$$

where V_f is the foraging speed, ω_f is the inertia weight, β_i^{food} is the food attractiveness and β_i^{best} is the effect of the best position of i^{th} krill. The neighbourhood is determined using a sensing distance around a krill individual (see Fig. 17).

The physical diffusion of the krill individuals is modelled as:

$$D_i = D_{max} \cdot \delta, \tag{32}$$

where δ is the random directional vector and its arrays are random values in [-1,1]. Additionally, crossover and mutation are incorporated into the KH.

Fig. 18 is the flowchart of KH.

L. FLOWER POLLINATION ALGORITHM

FPA is inspired by the pollination process of flower. About 90% of flowering plants belong to biotic pollination, while 10% of pollination takes abiotic form. Similarly, FPA uses a switch probability to control the local pollination and global pollination. The global search (global pollination) is modelled as:

$$X_{i,d}^{t+1} = X_{i,d}^{t} + l \cdot (X_{i,d}^{t} - X_{best,d}), \quad l \sim L(\lambda), \tag{33}$$

where $L(\lambda)$ represents the Lévy distribution. The local search (local pollination) is modelled as:

$$X_{i,d}^{t+1} = X_{i,d}^{t} + \epsilon \cdot (X_{j,d}^{t} - X_{k,d}^{t}), \quad \epsilon \sim U(0,1), \tag{34}$$

where j and k are two randomly selected individuals.

Fig. 19 is the flowchart of FPA. Since the global search of FPA is guided by the elite, FPA is a single-elite NIO algorithm.

M. FRUIT-FLY ALGORITHM OPTIMIZATION

FOA is a meta-heuristic based on the foraging behaviour of the fruit fly. The fruit fly has super olfaction. It can even smell food source from 40 km away. In FOA, each individual performs stochastic search. Unlike other NIOs, FOA normalizes individual positions with respect to the optimum. The update rule can be formulated as

$$\begin{aligned} X_i &= X_{axis} + r, \quad Y_i = Y_{axis} + r, \\ X_{axis,d}, Y_{axis,d} &= U(b_{l,d}, b_{u,d}), \\ r &\sim U(-1,1). \end{aligned} \tag{35}$$

For normalization, FOA calculates the distance to the optimum as the reciprocal of concentration value:

$$S_i = \frac{1}{\sqrt{x_i^2 + y_i^2}}. \tag{36}$$

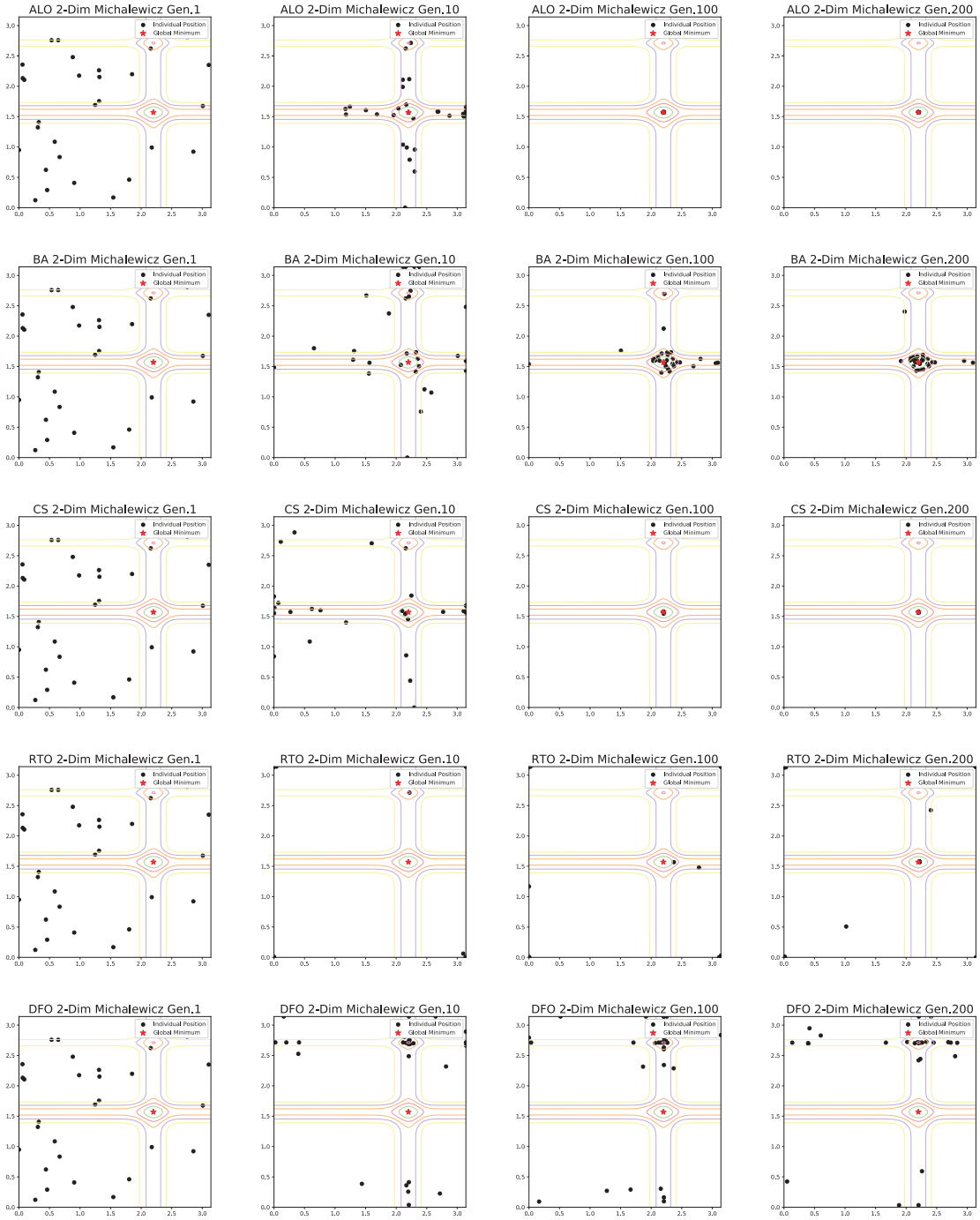


FIGURE 29. The contour plots of ALO, BA, CS, DE, DFO.

FOA is a single-elite algorithm. The elite with best fitness attracts other individuals, which makes X_{axis}, Y_{axis} in Eq. 35 be replaced by the position of the elite. Moreover, FOA is an easily implemented NIO due to its simple computational process.

N. MOTH FLAME OPTIMIZATION

The moth flies towards the flame which is the inspiration of MFO. When the light source is far away, the moth may fly in a straight line over a long distance following the moonlight navigation. When the light source is not far away, the moth will fly and spiral closer to the flames. MFO maintains a swarm of moths and a group of flames. The movement of each moth can be formulated as

$$X_{i,d}^{t+1} = D_{i,j} \cdot e^{br} \cdot \cos(2\pi r) + F_{j,d}, \quad r \sim U(-1, 1), \quad (37)$$

where b is a constant defining the shape of the spiral, $D_{i,j}$ is the distance between the i^{th} moth and the j^{th} flame.

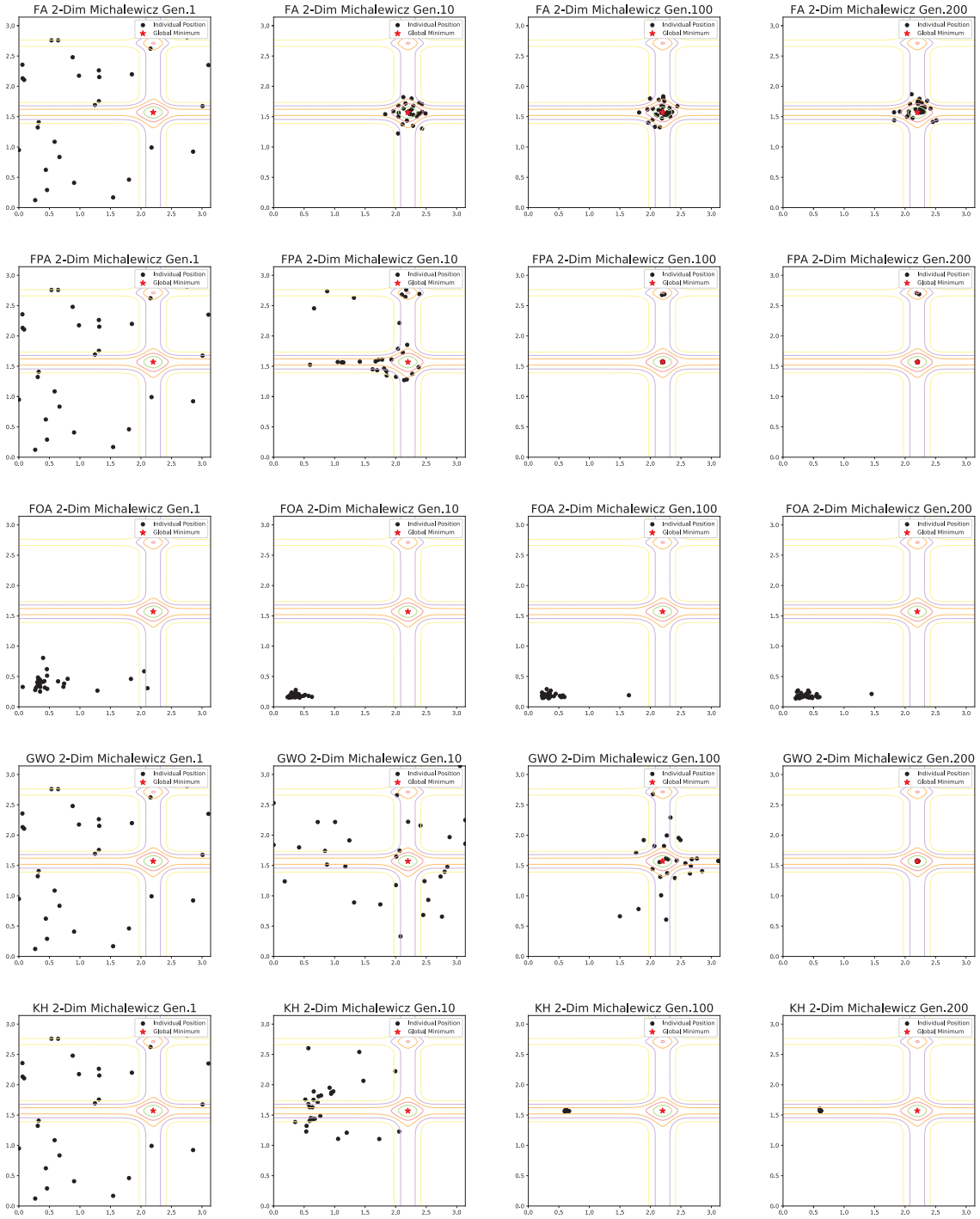


FIGURE 30. The contour plots of FA, FPA, FOA, GWO, KH.

As sub-step 1-2 illustrated, MFO sorts the current moths according to fitness which will become the flames of the next generation.

The order in the sequence of moths keeps unchanged during the course of the search. Each moth flies toward each flame one to one in order, as shown in Fig. 20. If a moth finds a better position from Eq. 37, the moth and the flame are updated. Fig. 21 is the flowchart of MFO. There is no elite which impacts other moths, so MFO falls into the non-elite NIO category.

O. BAT ALGORITHM

Inspired by the echolocation behaviour of bats, BA is a single-elite NIO algorithm. Bats fly with velocity v_i at position x_i with a minimum frequency f_{min}, varying wavelength λ and loudness A_0 to search for prey. They adjust the pulse rates(r_i)

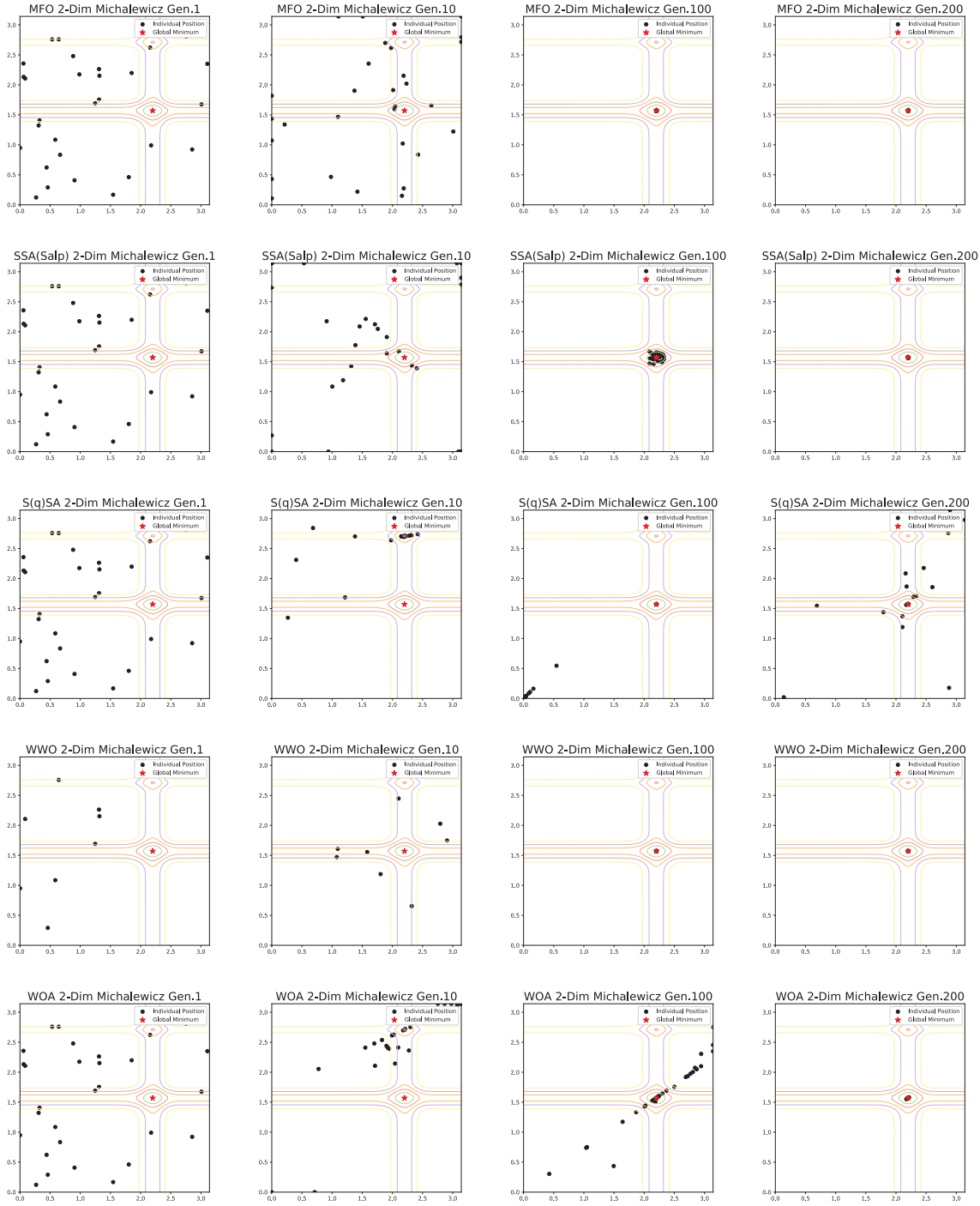


FIGURE 31. The contour plots of MFO, SSA, S(q)SA, WOA, WWO.

according to the degree of proximity to the optimal value individual. At the beginning of each iteration, BA updates the position and speed with the following formula.

$$\begin{aligned} v_i^{t+1} &= v_i^t + (X_i^t - X_{best}^t) \cdot f_i, \\ X_i^{t+1} &= X_i^t + v_i^t, \end{aligned} \tag{38}$$

where $f_i = f_{min} + (f_{max} - f_{min}) \cdot \beta$ ($\beta \sim U(0, 1)$) is used to adjust the velocity change. BA uses pulse rates to switch global search and local search. When random number $r > r_i$, BA performs local search around the elites, which is modelled as:

$$\begin{aligned} X_{i,d}^{t+1} &= X_{i,d}^t + \epsilon \cdot \bar{A}, \\ \bar{A} &= \frac{\sum_1^n A_i}{n}, \\ \epsilon &\sim U(-1, 1). \end{aligned} \tag{39}$$

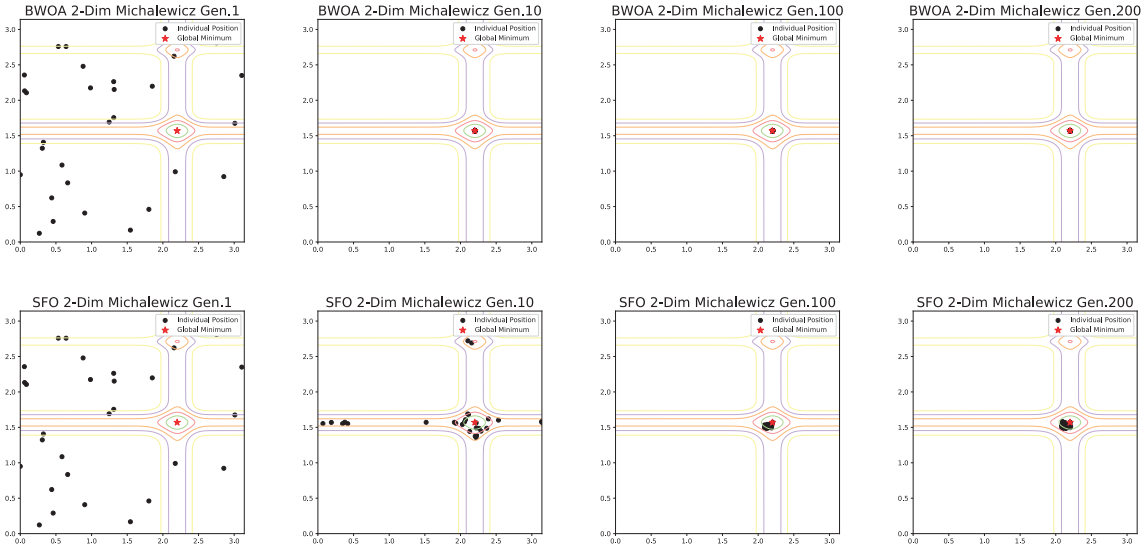


FIGURE 32. The contour plots of SFO, BWOA.

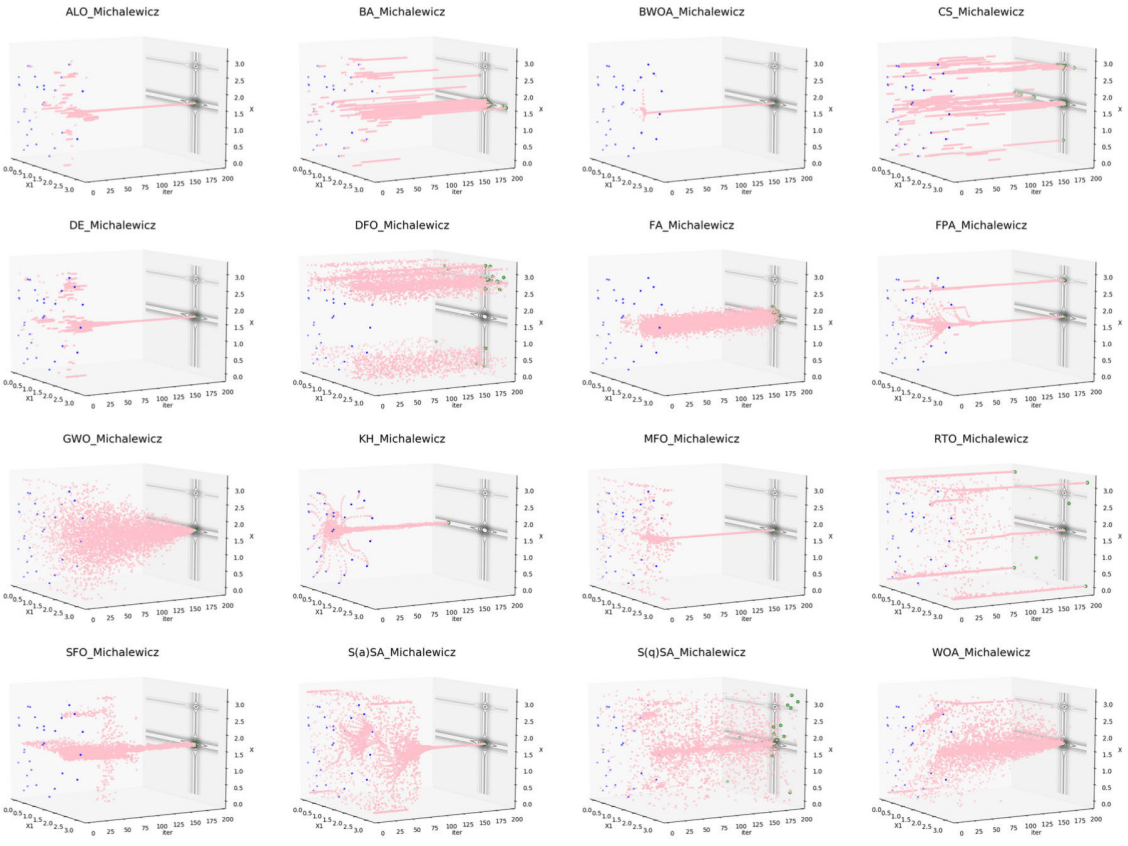


FIGURE 33. The contour progress on Michalewicz-2D.

Then BA performs global search by flying randomly in the search space. BA assumes that when the bat finds the prey, it temporarily stops making sounds. The new solution is accepted when $r < r_i$. At the same time, BA updates the individual's loudness and pulse rate.

$$A_i^{t+1} = \alpha \cdot A_i^t, \quad r_t^{t+1} = r_i^0(1 - e^{-\gamma t}),$$
$$\alpha = \gamma = 0.9. \tag{40}$$

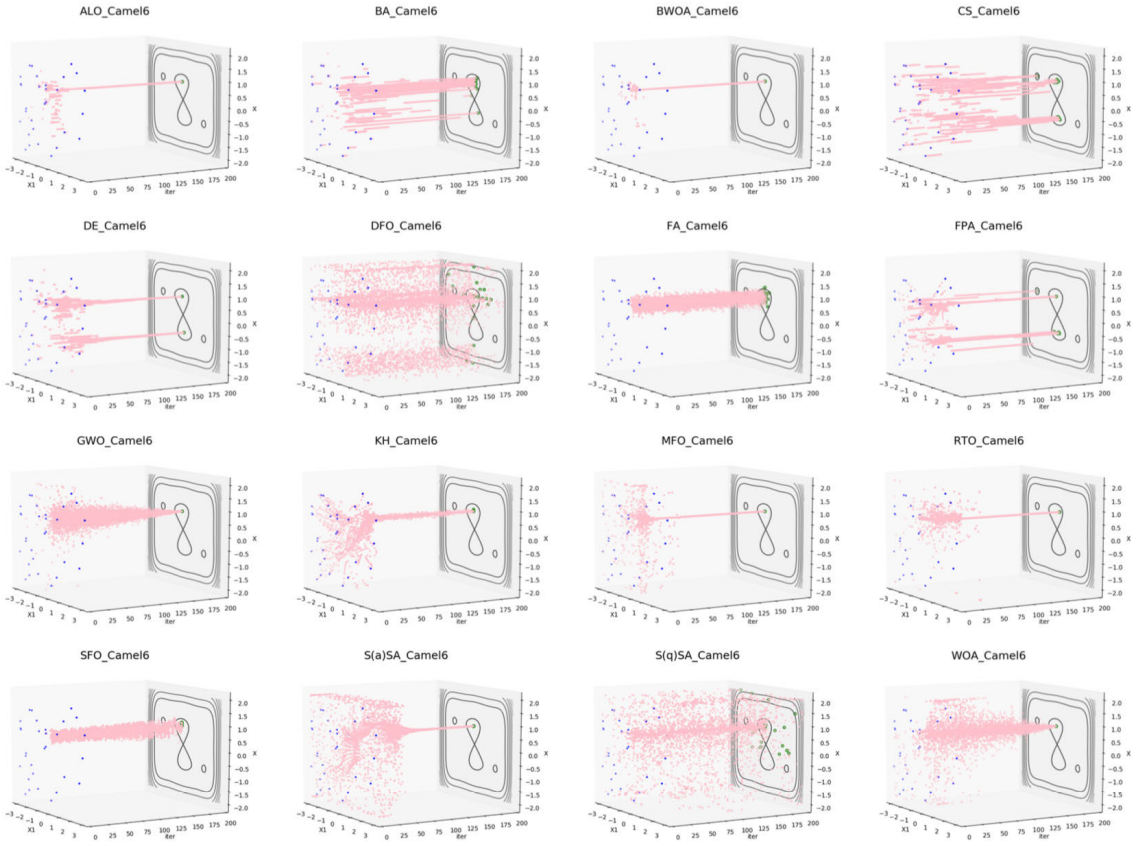


FIGURE 34. The contour progress on Camel6-2D.

P. FIREFLY ALGORITHM

Fireflies produce short and rhythmic flashes to attract mating partners or potential prey. A firefly's attractiveness is proportional to the light intensity, which defines the fitness of the individual in FA. The brighter flash an individual in the population produces, the greater it impacts on the other individuals. Moreover, brightness is related to absorption rate and distance which also determine the attractiveness of an individual to other individuals. The attractiveness is modelled as

$$\beta(r) = \beta_0 e^{-\gamma r^2}, \quad \beta_0 = 1, \tag{41}$$

where β_0 is the attractiveness at $r = 0$.

Each firefly is attracted by a better adjacent firefly:

$$\begin{aligned} X_{i,d}^{t+1} &= X_{i,d}^{t} + \beta_r \cdot e^{-\gamma {r_{i,j}}^2} + \alpha \cdot (r - \frac{1}{2})), \\ \alpha, r &\sim U(0,1), \quad r_{i,j} = |X_i - X_j|. \end{aligned} \tag{42}$$

The attractiveness attenuates with distance r^2 exponentially.

FA is a multi-elite algorithm, however there is a minor difference between FA and other multi-elite algorithms. In FA, each individual is compared with other individuals in each generation. If there exists a better individual from the viewpoint of itself, it will move towards the better individual. The elite for each individual is not fixed as the best individual in the population due to the absorption rate and distance. Therefore, more than one elites may exist in the population, and the total number of elites is not fixed during the process of the optimization, unlike other multi-elite algorithms. The flowchart of FA is given in Fig. 22.

Q. CUCKOO SEARCH

CS is inspired by the aggressive reproduction strategy of cuckoo. Some cuckoos engage the obligate brood parasitism by laying their eggs in the nests of other host birds. Each cuckoo lays one egg at a time, and dumps its egg in randomly chosen nest. In CS, nests, cuckoos and eggs are all considered when generating new candidate solution. CS is a non-elite algorithm, which means that in each updating process each individual is not a reference to other individuals. The updating process mainly rely on the random walk Lévy flight. However, random walk does not always produce better solutions. For those cases where the quality of the solution is not improved, CS uses a process similar to the simulated annealing algorithm to accept a worse solution with a certain probability, which correspond to sub-step 4-2 and 4-3 in the framework.

The random walk is modelled using Lévy flight:

$$X_{i,d}^{t+1} = X_{i,d}^{t} + \alpha \cdot Lévy(\lambda), \quad \lambda = 1.5, \tag{43}$$

TABLE 7. The experimental configuration.

Item	Specification
CPU	6 Intel(R) Xeon(R) CPU E5-2603 v4 @ 1.70GHz
Memory	16G
Storage	128G SSD
Operation System	Ubuntu 18.04
Programming language	Python 3.6.7

where α is the step size related to the scales of the problem. A better nests X with high quality of eggs will carry over to the next generations with a certain probability. Fig. 23 is the flowchart of CS.

IV. PERFORMANCE EVALUATION

In this section, all the collected NIO algorithms will be evaluated on 41 benchmark functions which are frequently used in related literature. Since these NIO algorithms have just been proposed for a short time, we add PSO and DE which are traditional excellent NIO algorithms to make comparisons more informative.

A. EXPERIMENTAL SETUP

We implement all aforementioned NIO algorithms in python language and run them on the same environment. The hardware and software configurations are shown in Table 7.

B. BENCHMARK FUNCTIONS

Twenty benchmark functions ([18], [19]) are listed in Table 9. Fig. 24 is the images of 2-D benchmark functions defined in Table 9. The readers can find 3-D surface plots for most two variable functions in the supplementary material. Some benchmark functions have multi-modal feature, while others are unimodal. Some benchmark functions have three variables or more, such as the Michalewicz function. Therefore, the benchmark functions including various versions of some functions are 41 in total.

C. ALGORITHM IMPLEMENTATION

It is difficult to make a fair comparison among the NIO algorithms with arbitrarily assigned parameters, since parameters do affect the results for most benchmark functions. Choosing the parameters recommended by the original authors is a compromise way. The recommended parameters are acceptable in most cases.

Specifically, the recommended parameters of these NIO algorithms along with PSO and DE, are listed in Table 8. Swarm size is a common parameter for all algorithms. As shown in Table 8, most NIO algorithms recommend 30 as the swarm size. Only WWO recommend a small swarm size which is much less than 30. For some algorithms without recommended swarm size, we assign the value 30 to the swarm size in order to make comparison as fair as possible. Similar treatment is given for the algorithms in which recommended swarm size is not far away from 30 (such as PSO in which the swarm size is 25), or 30 is in the recommended range.

TABLE 8. The parameters of NIO algorithms (SS is the abbreviation of "swarm size").

NIO	SS	Parameters
ALO	30	-
BA	30	$\alpha = 0.9, \gamma = 0.9, f_{min} = 0, f_{max} = 2$
CS	30	$p_a = 0.25, \alpha = 1, \lambda = 1.5$
DE	30	$F = 2, CR = 0.9$
DFO	30	$d_t = 0.001$
FA	30	$\alpha = 0.5, \beta = 1, absorption = 0.001$
FPA	30	$P = 0.8$
FOA	30	-
GWO	30	-
KH	30	$N_{max} = 0.01, V_f = 0.02,$ $D_{max} = 0.002, C_t = 0.93,$ $W_n = 0.42, W_f = 0.38, nn = 5$
MFO	30	-
PSO	30	$c_1 = 2, c_2 = 2, w = 0.7,$ $v_{max} = 4, v_{min} = -4$
S(a)SA	30	-
S(q)SA	30	$G_C = 1.9, P_{dp} = 0.1, rho = 1.204,$ $v = 5.25, s = 154, cd = 0.6,$ $hg = 8, sf = 18, sc = 0.3$
WWO	8	$h_{max} = 6, \lambda = 0.5, \alpha = 1.0026,$ $\beta \in [0.001, 0.25]$
WOA	30	-
BWOA	30	$p_r = 0.6, c_r = 0.44, m_r = 0.4$
SFO	30	$A = 4, e = 0.001, pp = 0.02$

We use recommended value for other behavioural parameters. Some authors did not recommend parameters in their original paper. In that case, we use the parameters in literature they published later or from source codes they provided. If the author only recommends a range for a behavioural parameter, we randomly adopt a value in the range.

KH has four crossover and mutation strategies to address various optimization problems. We just implement one strategy to simplify comparison. Most algorithms generate initialize swarm randomly. We employ the same seed when starting the pseudo random generating processes to generate the same initial swarm for fair comparison.

The bound of 200 generations is the common stopping criterion for most NIO algorithms with swarm size 30. Larger generation bound is given to WWO which has less swarm size.

D. PERFORMANCE TEST

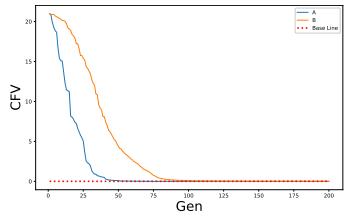
The convergence curves are shown in Fig.25 and Fig.26. Each sub-figure represents a benchmark function. The convergence curves of various NIO algorithms for each benchmark function are plotted together in each sub-Fig.

Most algorithms have good convergence performance. For low dimensional functions, it is common to find the optimum with satisfied accuracy within 100 generations. And

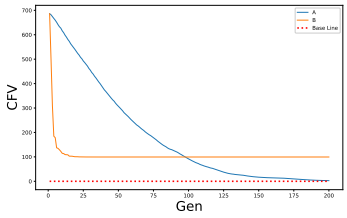
TABLE 9. Benchmark functions.

Name	Definition	Dim	Domain	f_{min}
Ackley	$f_1: -a \cdot e^{(-b\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2})} - e^{(\frac{1}{n}\sum_{i=1}^{n} cos(cx_i))} + a + e^1$	2	$x_i \in [-32.768, 32.768]$	0
Bukin6	$f_2: 100\sqrt{\lvert x_2 - 0.01x_1^2\rvert} + 0.01\lvert x_1 + 10\rvert$	2	$x_1 \in$[-15,5] $x_2 \in$[-3,3]	-10
Camel3	$f_3: 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1x_2 + x_2^2$	2	$x_i \in [-5, 5]$	0
Camel6	$f_4: (4 - 2.1x_1^2 + \frac{x_1^4}{3})x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$	2	$x_1 \in$[-3,3] $x_2 \in$[-2,2]	-1.0316
Cross-in-Tray	$f_5: -0.0001 \cdot (\lvert sin(x_1)sin(x_2) \cdot (e^{(\lvert 100 - \frac{\sqrt{x_1^2+x_2^2}}{\pi}\rvert)}\rvert + 1)^{0.1}$	2	$x_i \in [-10, 10]$	-2.06261218
Easom	$f_6: cos(x_1)cos(x_2)e^{((x_1\pi)^2(x_2\pi)^2)}$	2	$x_i \in [-100, 100]$	-1
Eggholder	$f_7: -(x_2 + 47)sin(\sqrt{\lvert x_2 + \frac{x_1}{2} + 47\rvert}) - x_1 sin(\sqrt{\lvert x_1 - (x_2 + 47)\rvert})$	2	$x_i \in [-512, 512]$	-959.6407
Griewank	$f_8: 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}})$	2	$x_i \in [-600, 600]$	0
Holdertable	$f_9: -\lvert sin(x_1)cos(x_2)e^{(\lvert 1 - \frac{\sqrt{x_1^2+x_2^2}}{\pi}\rvert)}\rvert$	2	$x_i \in [-10, 10]$	-19.2085
Levy13	$f_{10}: sin^2(3\pi x_1) + (x_1 - 1)^2(1 + sin^2(3\pi x_2)) + (x_2 - 1)^2(1 + sin^2(2\pi x_2))$	2	$x_i \in [-10, 10]$	0
Michalewicz	$f_{11}: -\sum_{i=1}^{n} sin(x_i)sin^{2m}(\frac{ix_i^2}{\pi})$	2	$x_i \in [0, \pi]$	-1.8013
Rastrigin	$f_{12}: 10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i))$	2	$x_i \in [-5.12, 5.12]$	0
Rosenbrock	$f_{13}: \sum_{i=1}^{n}[b(x_{i+1} - x_i^2)^2 + (a - x_i)^2]$	2	$x_i \in [-5, 10]$	0
Schaffer2	$f_{14}: 0.5 + \frac{sin^2(x_1^2 - x_2^2) - 0.5}{(1 + 0.001(x_1^2 + x_2^2))^2}$	2	$x_i \in [-100, 100]$	0
Schwefel	$f_{15}: 418.9829d - \sum_{i=1}^{n} x_i sin(\sqrt{\lvert x_i\rvert})$	2	$x_i \in [-500, 500]$	0
Shubert	$f_{16}: \prod_{i=1}^{n}\left(\sum_{j=1}^{5} cos((j+1)x_i + j)\right)$	2	$x_i \in [-10, 10]$	-186.7329
Sphere	$f_{17}: \sum_{i=1}^{n} x_i^2$	2	$x_i \in [-5.12, 5.12]$	0
Stybtang	$f_{18}: \frac{1}{2}\sum_{i=1}^{n}(x_i^4 - 16x_i^2 + 5x_i)$	2	$x_i \in [-5, 5]$	$-39.16599d$
Michalewicz-5D	$f_{19}: -\sum_{i=1}^{n} sin(x_i)sin^{2m}(\frac{ix_i^2}{\pi})$	5	$x_i \in [0, \pi]$	-4.687658
Michalewicz-10D	$f_{20}: -\sum_{i=1}^{n} sin(x_i)sin^{2m}(\frac{ix_i^2}{\pi})$	10	$x_i \in [0, \pi]$	-9.66015
Ackley-30D	$f_{21}: -a \cdot e^{(-b\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2})} - e^{(\frac{1}{n}\sum_{i=1}^{n} cos(cx_i))} + a + e^1$	30	$x_i \in [-32.768, 32.768]$	0
Griewank-30D	$f_{22}: 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}})$	30	$x_i \in [-600, 600]$	0
Rastrigin-30D	$f_{23}: 10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i))$	30	$x_i \in [-5.12, 5.12]$	0
Rosenbrock-30D	$f_{24}: \sum_{i=1}^{n}[b(x_{i+1} - x_i^2)^2 + (a - x_i)^2]$	30	$x_i \in [-5, 10]$	0
Schwefel-30D	$f_{25}: 418.9829d - \sum_{i=1}^{n} x_i sin(\sqrt{\lvert x_i\rvert})$	30	$x_i \in [-500, 500]$	0
Sphere-30D	$f_{26}: \sum_{i=1}^{n} x_i^2$	30	$x_i \in [-5.12, 5.12]$	0
Stybtang-30D	$f_{27}: \frac{1}{2}\sum_{i=1}^{n}(x_i^4 - 16x_i^2 + 5x_i)$	30	$x_i \in [-5, 5]$	$-39.16599d$
Ackley-50D	$f_{28}: -a \cdot e^{(-b\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2})} - e^{(\frac{1}{n}\sum_{i=1}^{n} cos(cx_i))} + a + e^1$	50	$x_i \in [-32.768, 32.768]$	0
Griewank-50D	$f_{29}: 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}})$	50	$x_i \in [-600, 600]$	0
Rastrigin-50D	$f_{30}: 10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i))$	50	$x_i \in [-5.12, 5.12]$	0
Rosenbrock-50D	$f_{31}: \sum_{i=1}^{n}[b(x_{i+1} - x_i^2)^2 + (a - x_i)^2]$	50	$x_i \in [-5, 10]$	0
Schwefel-50D	$f_{32}: 418.9829d - \sum_{i=1}^{n} x_i sin(\sqrt{\lvert x_i\rvert})$	50	$x_i \in [-500, 500]$	0
Sphere-50D	$f_{33}: \sum_{i=1}^{n} x_i^2$	50	$x_i \in [-5.12, 5.12]$	0
Stybtang-50D	$f_{34}: \frac{1}{2}\sum_{i=1}^{n}(x_i^4 - 16x_i^2 + 5x_i)$	50	$x_i \in [-5, 5]$	$-39.16599d$
Ackley-100D	$f_{35}: -a \cdot e^{(-b\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2})} - e^{(\frac{1}{n}\sum_{i=1}^{n} cos(cx_i))} + a + e^1$	100	$x_i \in [-32.768, 32.768]$	0
Griewank-100D	$f_{36}: 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}})$	100	$x_i \in [-600, 600]$	0
Rastrigin-100D	$f_{37}: 10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i))$	100	$x_i \in [-5.12, 5.12]$	0
Rosenbrock-100D	$f_{38}: \sum_{i=1}^{n}[b(x_{i+1} - x_i^2)^2 + (a - x_i)^2]$	100	$x_i \in [-5, 10]$	0
Schwefel-100D	$f_{39}: 418.9829d - \sum_{i=1}^{n} x_i sin(\sqrt{\lvert x_i\rvert})$	100	$x_i \in [-500, 500]$	0
Sphere-100D	$f_{40}: \sum_{i=1}^{n} x_i^2$	100	$x_i \in [-5.12, 5.12]$	0
Stybtang-100D	$f_{41}: \frac{1}{2}\sum_{i=1}^{n}(x_i^4 - 16x_i^2 + 5x_i)$	100	$x_i \in [-5, 5]$	$-39.16599d$

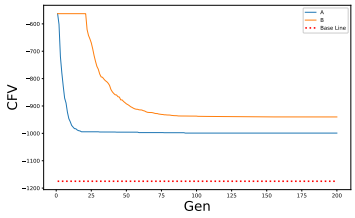
the performance difference among diverse NIO algorithms with recommended parameters is not as great as expected. For higher dimensional functions, NIO algorithms shows significantly different performances.



(a) Champion NIO of type I: A



(b) Champion NIO of type I: A



(c) Champion NIO of type II: A

FIGURE 35. Definition of champion NIO.

TABLE 10. Occurrence of being champion NIOs.

Champion NIO	Occurrence
WOA	14
SFO	7
GWO	5
FA	4
RTO	2
BWOA	2
DFO	2
KH	1
MFO	1
BA	1
S(q)SA	1
S(a)SA	1

It should be noted that employing the recommended parameters is still unfair from the view of best performance. In most cases, an algorithm with the recommended parameters may generate good results for some benchmark functions, but may give poor performance for other benchmark functions.

We run all algorithms with random initial swarm for 30 times independently. Comparison of the CFV error is shown in Fig.27 and Fig.28. Since the results of the CFV error show tremendous difference in scale, it is hard to observe the slight difference in the same scale. Therefore, we also provide the logarithm version of box plots to make the difference more observable, which can be found in the supplementary material. It is interesting that no NIO algorithm shows advantage over other algorithms for all benchmark functions.

The contour plots regarding how diverse NIO algorithms with the same initial swarm find the optimum are shown in Fig.29, Fig.30, Fig.31, and Fig.32. The snapshots of Gen 1, 10, 100, and 200 are exhibited in different columns. We only present contour plots of five NIO algorithms to save place. For the same reason, only the contours of function f_3 is displayed here. Vast other contour plots can be found in the supplementary material. We also provide the 3D contour progress plots for two 2D benchmark functions as shown in Fig.33 and Fig.34. From the 3D contour progress plots, one can observe more detailedly how diverse NIOs converge during the course of search.

E. PERFORMANCE ANALYSIS

From the convergence test and repeated test for the NIO algorithms, a basic fact is that most NIO algorithms can reach or gradually approach the optimum of the benchmark function (2D, 5D, 10D) with the default parameters in Table 8. The convergence curves (see Fig. 25 and Fig. 26) show apparent diversities for 30D or higher benchmark functions. Before determining which algorithm is more competitive than others for a certain benchmark function, we define the ''champion NIO'' for a benchmark function as follows:

Given a certain accuracy, the algorithm who first hits the optimum of the benchmark function with the pre-assigned accuracy is defined as the champion NIO of type I (see Fig. 35a and 35b). If all the algorithms do not hit the optimum with the pre-assigned accuracy, then the algorithm with the smallest error within the certain generation is defined as the champion NIO of type II (see Fig. 35c). Specifically, in this test, the generation bound is set to 200.

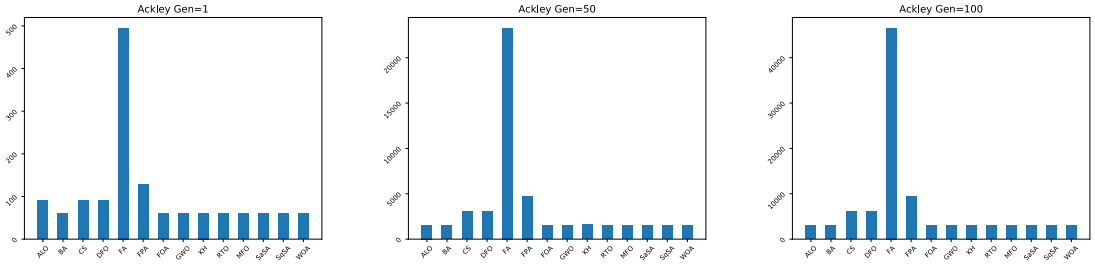
With the definition of champion NIO in advance, we can summarize the previous results in Table 10 from the viewpoint of ''champion NIO''. As shown in Table 10, it is a comparatively frequent occurrence for WOA, SFO, GWO, and FA to be a champion NIO. As a matter of fact, for many benchmark functions, the performance difference among some champion/quasi-champion NIO algorithms is very small. Therefore, some NIO algorithm that do not appear in the table may be an excellent but unlucky algorithm.

Compared with Table 10, Table 11 presents more details on which algorithm the champion NIO is for each function, and in which Type. All the champion NIO in the 2-D benchmark is Type I. It means that there is at least one algorithm that converges to the optimum with a certain accuracy within 200 generations on all 2-D benchmarks.

The outperformance results of benchmarks in 5D or higher dimensions are shown in Table 11. It can be seen from the table that WOA and GWO perform very well on high-dimensional benchmark functions. WOA becomes champion for fourteen times, while GWO becomes champion for five times, though these two algorithms do not perform well in low-dimensional benchmark functions. It is worth

TABLE 11. Champion NIOs for each benchmark function.

Benchmark	Type	Champion NIO	Benchmark	Type	Champion NIO	Benchmark	Type	Champion NIO
f_1	I	SFO	f_{15}	I	RTO	f_{29}	I	WOA
f_2	I	RTO	f_{16}	I	S(q)SA	f_{30}	I	WOA
f_3	I	SFO	f_{17}	I	FA	f_{31}	II	GWO
f_4	I	FA	f_{18}	I	DFO	f_{32}	II	WOA
f_5	I	BA	f_{19}	I	KH	f_{33}	I	WOA
f_6	I	SFO	f_{20}	II	S(a)SA	f_{34}	II	FA
f_7	I	MFO	f_{21}	I	GWO	f_{35}	I	WOA
f_8	I	SFO	f_{22}	I	WOA	f_{36}	I	WOA
f_9	I	FA	f_{23}	I	WOA	f_{37}	I	WOA
f_{10}	I	BWOA	f_{24}	II	WOA	f_{38}	II	GWO
f_{11}	I	BWOA	f_{25}	II	WOA	f_{39}	II	WOA
f_{12}	I	SFO	f_{26}	I	GWO	f_{40}	I	WOA
f_{13}	I	SFO	f_{27}	II	DFO	f_{41}	II	WOA
f_{14}	I	SFO	f_{28}	I	GWO			



(a) 1 generation (b) 50 generations (c) 100 generations

FIGURE 36. Cost function evaluation counts for different generations.

TABLE 12. Applications of the NIO algorithms I.

Fields		S(q)SA	S(a)SA	WOA	ALO	MFO	GWO
Energy	Power System	[84]	[85], [86]	-	[87], [88]	[89], [90]	[91]
	Battery	-	-	-	[92]	-	-
Enviroment	Air quality	[93]	-	-	-	-	-
	Protection	-	-	-	-	-	[94], [95]
Communication	5G	-	-	[96]	-	-	[97]
	Electronics	[98]	[99]	-	-	-	[100]
Computer & Network	Vedio / Image	-	[101]	-	[102]	-	-
	Robots	-	-	[103]	-	-	[104]
	Social Network	-	-	[105]	-	-	[106]
	Neural Network	-	-	-	[87]	-	[43], [107]
Others		[108]	-	[109]	[110]	-	-

mentioning that 3/5 champion NIO types of GWO are of type I, while 9/14 champion NIO types of WOA are of type I.

F. COST FUNCTION EVALUATION COUNTS

We take measurements of the cost function evaluation counts extensively instead of the computational complexity estimation for the NIOs. For each NIO algorithm, a counting agent was inserted into the python codes to collect and sum up the calls of cost function evaluation during the optimization process.

If we specify a target cost function value, some NIOs may fail for some benchmark functions. So we fix the generation bound and swarm size to make fair measurements. Fig.36 exhibits the cost function evaluation counts for the Ackley function in the 1st, 50th and 100th generation. All NIOs have 30 individuals as described in Table 8. Since WWO has only 8 individuals, its cost function evaluation count is not exhibited in Fig.36. The original FA needs more cost function evaluations because each firefly wants to determine which one is the best adjacent firefly, thus calculating and

TABLE 13. Applications of the NIO algorithms II.

Fields		RTO	KH	FOA	FPA	BA	FA	CS
Energy	Power System	[111], [112]	-	-	-	-	[113]	-
	Battery	-	-	-	-	-	-	-
Enviroment	Air quality	-	-	-	-	-	[114]	[115]
	Protection	-	[116]	-	-	-	-	-
Communication	5G	-	-	-	-	[117]	-	-
	Electronics	-	[118]	-	-	-	-	[119]
Computer & Network	Vedio / Image	-	-	[120]	[121], [122]	-	-	-
	Robots	-	-	-	-	[123]	[124]	[125]
	Social Network	-	-	-	-	-	-	-
	Neural Network	-	[126]	-	-	[127]	-	[89]
Others		[34], [128]	-	[129]	[130]	-	[131]	[132]

comparing several times in each generation. In fact, the cost function evaluation counts of the original FA can be reduced by calculating cost function value once in advance and comparing many times later.

V. APPLICATIONS OF THE NIO ALGORITHMS

The past couples of years have seen vast successful applications in many and various fields such as energy, environment, and communication.

It is impossible to compare and recommend professional algorithms for all these applications, therefore, we briefly summarize some successful applications that use NIO algorithms in literature instead. The scope is still within the SCIE collection of WOS.

Our survey includes some traditional applications and some new applications published recently, as shown in Table 12 and Table 13. Since some NIO algorithms such as SFO and BWOA were just published, time is needed for finding various supporting applications.

VI. BEHAVIOURAL PARAMETER OPTIMIZATION

Manually optimizing the behavioural parameter is not an efficient way. Employing another overlaid optimizer to perform the behavioural parameter optimization is a wise choice. The overlaid optimizer is referred to as meta-optimizer [133]. The NIO optimizer itself can be employed as a meta-optimizer. In that case, it is interesting that both the meta-optimizer and the optimized optimizer are NIO algorithms.

Fig.37 depicts the flowchart of a typical meta-optimization system. As shown in Fig.37, the input of the optimized NIO is the parameters generated by meta-optimizer. The optimized NIO as a whole "black box" is the optimization objective of the meta-optimizer. The meta-optimizer outputs the optimized behavioural parameters finally.

When the parameters are optimized, the calculation increases exponentially. To avoid combinatorial explosion, we only select CS, DE, KH, and S(q)SA as optimized NIOs, and select CS, GWO, KH, and WOA as the meta-optimizers. The selected meta-optimizers are comparatively efficient NIO algorithms. Table 14 lists the numbers of the behavioural

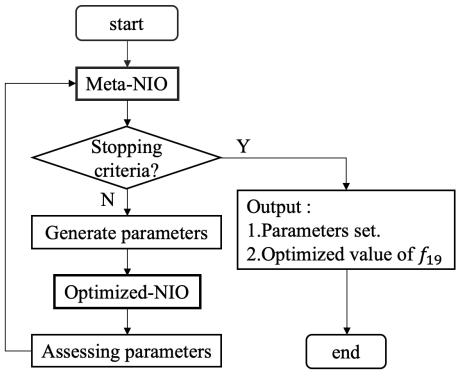


FIGURE 37. The flowchart of meta-optimization.

TABLE 14. Parameters and dimensions of the Optimized NIOs.

NIO	Dim	Parameter range
CS	3	$p_a \in [0, 0.5], \alpha \in [0, 2], \lambda \in [1, 2]$
DE	2	$F \in [0, 2], CR \in [0, 1]$
KH	2	$N_{max} \in [0, 0.1], V_f \in [0, 0.1]$
S(q)SA	3	$G_c \in [1, 10], P_{dp} \in [0.1, 0.9], sf \in [16, 37]$

parameters and the pre-assigned parameter ranges of the optimized NIOs. The number of behavioural parameters of the optimized NIO is the dimension of the meta-optimization problem.

Each optimized NIO runs 200 generations with 30 individuals to find the optimum of the function f_{19}. Each meta-optimizer runs 100 generations with 30 individuals to find optimal behavioural parameters of the optimized NIO. Table 15 gives the optimized behavioural parameters of CS, GWO, KH, and S(q)SA.

Despite the fact that diverse meta-optimizers generate diverse optimized parameters, these parameters are all excellent in finding optimum. The behavioural parameters of CS, DE, KH, and S(q)SA obtained by using GWO meta-optimizer gain distinct advantage in optimizing the benchmark function f_{19} (Michalewicz function with $D = 5$), as shown in Fig.38. Generally, after parameter optimization, the optimized-NIO

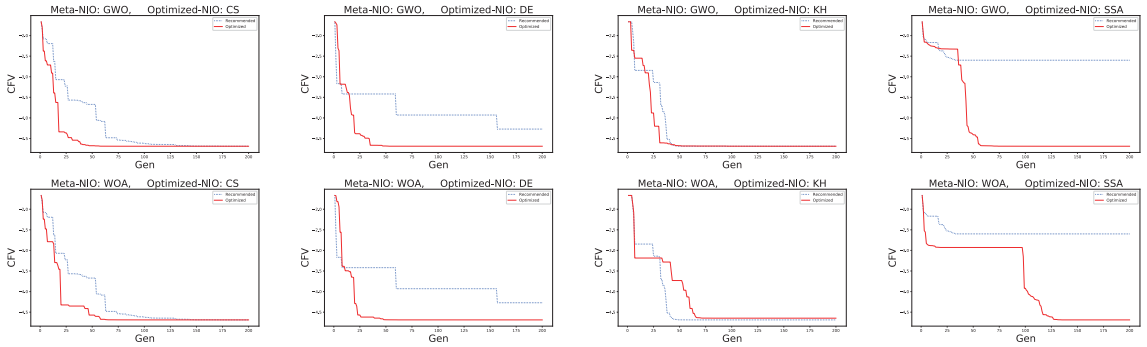


FIGURE 38. Convergence curves (recommended parameters vs optimized parameters).

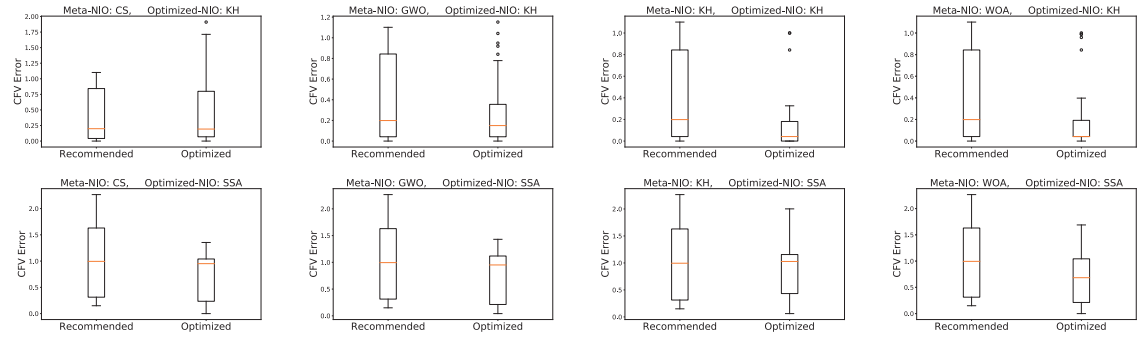


FIGURE 39. Multiple tests (recommended parameters vs optimized parameters).

TABLE 15. Optimized parameters.

Meta-	Optimized-	Optimized parameters
CS	CS	$p_a = 0.499, \alpha = 1.967, \lambda = 1.748$
	DE	$F = 0.083, CR = 0.077$
	KH	$N_{max} = 0.027, V_f = 0$
	S(q)SA	$G_c = 1.495$, $P_{dp} = 0.230$, $sf = 27.5735$
GWO	CS	$p_a = 0.405, \alpha = 1.990, \lambda = 1.744$
	DE	$F = 0.190, CR = 0.012$
	KH	$N_{max} = 0.034, V_f = 0.099$
	S(q)SA	$G_c = 1.006$, $P_{dp} = 0.051$, $sf = 16$
KH	CS	$p_a = 0.154, \alpha = 1.736, \lambda = 1.659$
	DE	$F = 0.304, CR = 0.421$
	KH	$N_{max} = 0.012, V_f = 0.052$
	S(q)SA	$G_c = 2.777$, $P_{dp} = 0.585$, $sf = 25.856$
WOA	CS	$p_a = 0.358, \alpha = 1.923, \lambda = 1.713$
	DE	$F = 0.185, CR = 0.058$
	KH	$N_{max} = 0.008, V_f = 0.036$
	S(q)SA	$G_c = 2.334$, $P_{dp} = 0.227$, $sf = 36.997$

can have substantially better performance on the same test function f_{19}. An exception is that the optimized KH does not converge faster after parameter optimization. The reason may be that the KH with the recommended parameters already have very good performance.

The behavioural parameters of KH obtained by KH, WOA, and GWO meta-optimizers have greater advantage than that by CS, see Fig.39. More results regarding NIO meta-optimizations can be found in the supplementary material.

VII. CONCLUSIONS

This paper presents a review of over a dozen NIO algorithms proposed after 2008. The NIO algorithms are selected from highly impact journal or conference that are also widely cited. We review the taxonomy of NIOs and present a new perspective concerning how many elites the NIO algorithm has. We introduce various NIO algorithms and their inspirations from this perspective. We test all NIO algorithms on 41 widely used benchmark functions. We compared these NIO algorithms in terms of contour plots, 3D plots, and convergence curves. All NIO algorithms have outstanding search ability for easier problems. Most NIO algorithms show superb performance in some hard problems but are not well-behaved for other hard problems. It supports the Free-lunch to some extent.

We optimize the behavioural parameters of various NIO algorithms using the NIO algorithms. Most optimized parameters are consistent with the suggested parameters by authors, while some optimized parameters different to the suggested parameters can provide better performance than the recommended parameters.

It should be noted that there are so many limitations this review cannot transcend. This paper does not cover discrete metaheuristics which may have more applications than the continuous algorithms. It does not cover the various

optimization problems such as multi-objective, multi-modal, dynamic problems, etc.. It does not provide any applications that is an important topic for NIOs indeed. Moreover, the NIOs included in this paper are far less than all excellent NIOs that has been proposed so far.

CONFLICTS OF INTEREST

The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

REFERENCES

[1] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0965997813001853

[2] H. Li and T. Li, "Bark beetle larval dynamics carved in the egg gallery: A study of mathematically reconstructing bark beetle tunnel maps," *Adv. Difference Equ.*, vol. 2019, no. 1, p. 513, Dec. 2019, doi: 10.1186/s13662-019-2452-2.

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

[4] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Comput. Surveys*, vol. 35, no. 3, pp. 268–308, Sep. 2003, doi: 10.1145/937503.937505.

[5] H. Li, P. Zou, Z. Huang, C. Zeng, and X. Liu, "Multimodal optimization using whale optimization algorithm enhanced with local search and niching technique," *Math. Biosci. Eng.*, vol. 17, no. 1, pp. 1–27, 2020, doi: 10.3934/mbe.2020001.

[6] B. Kitchenham and P. Brereton, "A systematic review of systematic review process research in software engineering," *Inf. Softw. Technol.*, vol. 55, no. 12, pp. 2049–2075, Dec. 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584913001560

[7] V. Hayyolalam and A. A. P. Kazem, "Black widow optimization algorithm: A novel meta-heuristic approach for solving engineering optimization problems," *Eng. Appl. Artif. Intell.*, vol. 87, Jan. 2020, Art. no. 103249. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0952197619302283

[8] H. Li and J. Zhang, "Fast source term estimation using the PGA-NM hybrid method," *Eng. Appl. Artif. Intell.*, vol. 62, pp. 68–79, Jun. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0952197617300647

[9] H. Li, J. Zhang, and J. Yi, "Computational source term estimation of the Gaussian puff dispersion," *Soft Comput.*, vol. 23, no. 1, pp. 59–75, Jan. 2019, doi: 10.1007/s00500-018-3440-2.

[10] H. Yi, Q. Duan, and T. W. Liao, "Three improved hybrid metaheuristic algorithms for engineering design optimization," *Appl. Soft Comput.*, vol. 13, no. 5, pp. 2433–2444, May 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494612005352

[11] I. Fister Jr., X.-S. Yang, I. Fister, J. Brest, and D. Fister, "A brief review of nature-inspired algorithms for optimization," 2013, *arXiv:1307.4186*. [Online]. Available: http://arxiv.org/abs/1307.4186

[12] N. Siddique and H. Adeli, "Nature inspired computing: An overview and some future directions," *Cognit. Comput.*, vol. 7, no. 6, pp. 706–714, Dec. 2015, doi: 10.1007/s12559-015-9370-8.

[13] A. K. Kar, "Bio inspired computing—A review of algorithms and scope of applications," *Expert Syst. Appl.*, vol. 59, pp. 20–32, Oct. 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S095741741630183X

[14] F. Valdez, P. Melin, and O. Castillo, "A survey on nature-inspired optimization algorithms with fuzzy logic for dynamic parameter adaptation," *Expert Syst. Appl.*, vol. 41, no. 14, pp. 6459–6466, Oct. 2014. [Online]. Available: http://www.sciencedirect.com/science/articsle/pii/S0957417414002127

[15] P. Agarwal and S. Mehta, "Empirical analysis of five nature-inspired algorithms on real parameter optimization problems," *Artif. Intell. Rev.*, vol. 50, no. 3, pp. 383–439, Oct. 2018, doi: 10.1007/s10462-017-9547-5.

[16] S. J. Nanda and G. Panda, "A survey on nature inspired metaheuristic algorithms for partitional clustering," *Swarm Evol. Comput.*, vol. 16, pp. 1–18, Jun. 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S221065021300076X

[17] D. Molina, J. Poyatos, J. D. Ser, S. García, A. Hussain, and F. Herrera, "Comprehensive taxonomies of Nature- and bio-inspired optimization: Inspiration versus algorithmic behavior, critical analysis and recommendations," 2020, *arXiv:2002.08136*. [Online]. Available: http://arxiv.org/abs/2002.08136

[18] M. Jamil and X. S. Yang, "A literature survey of benchmark functions for global optimisation problems," *Int. J. Math. Model. Numer. Optim.*, vol. 4, no. 2, pp. 150–194, 2013.

[19] J. J. Liang, P. N. Suganthan, and K. Deb, "Novel composition test functions for numerical global optimization," in *Proc. IEEE Swarm Intell. Symp. (SIS)*, Jun. 2005, pp. 68–75.

[20] X.-S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," 2010, *arXiv:1003.1409*. [Online]. Available: http://arxiv.org/abs/1003.1409

[21] J. G. Digalakis and K. G. Margaritis, "An experimental study of benchmarking functions for genetic algorithms," *Int. J. Comput. Math.*, vol. 79, no. 4, pp. 403–416, Jan. 2002, doi: 10.1080/00207160210939.

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

[23] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Cambridge, MA, USA: MIT Press, 1992.

[24] D. B. Fogel and A. Ghozeil, "Schema processing under proportional selection in the presence of random effects," *IEEE Trans. Evol. Comput.*, vol. 1, no. 4, pp. 290–293, Nov. 1997.

[25] P. Merz, "Advanced fitness landscape analysis and the performance of memetic algorithms," *Evol. Comput.*, vol. 12, no. 3, pp. 303–325, Sep. 2004.

[26] S. Shadravan, H. R. Naji, and V. K. Bardsiri, "The sailfish optimizer: A novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems," *Eng. Appl. Artif. Intell.*, vol. 80, pp. 20–34, Apr. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0952197619300016

[27] M. Jain, V. Singh, and A. Rani, "A novel nature-inspired algorithm for optimization: Squirrel search algorithm," *Swarm Evol. Comput.*, vol. 44, pp. 148–175, Feb. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2210650217305229

[28] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp swarm algorithm: A bio-inspired optimizer for engineering design problems," *Adv. Eng. Softw.*, vol. 114, pp. 163–191, Dec. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0965997816307736

[29] H. Faris, M. M. Mafarja, A. A. Heidari, I. Aljarah, A. M. Al-Zoubi, S. Mirjalili, and H. Fujita, "An efficient binary salp swarm algorithm with crossover scheme for feature selection problems," *Knowl.-Based Syst.*, vol. 154, pp. 43–67, Aug. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950705118302132

[30] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Adv. Eng. Softw.*, vol. 95, pp. 51–67, May 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0965997816300163

[31] M. A. E. Aziz, A. A. Ewees, and A. E. Hassanien, "Whale optimization algorithm and moth-flame optimization for multilevel thresholding image segmentation," *Expert Syst. Appl.*, vol. 83, pp. 242–256, Oct. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417417302671

[32] M. M. Mafarja and S. Mirjalili, "Hybrid whale optimization algorithm with simulated annealing for feature selection," *Neurocomputing*, vol. 260, pp. 302–312, Oct. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S092523121730807X

[33] I. Aljarah, H. Faris, and S. Mirjalili, "Optimizing connection weights in neural networks using the whale optimization algorithm," *Soft Comput.*, vol. 22, no. 1, pp. 1–15, Jan. 2018, doi: 10.1007/s00500-016-2442-1.

[34] Y. Labbi, D. B. Attous, H. A. Gabbar, B. Mahdad, and A. Zidan, "A new rooted tree optimization algorithm for economic dispatch with valve-point effect," *Int. J. Electr. Power Energy Syst.*, vol. 79, pp. 298–311, Jul. 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0142061516000405

[35] A. Wadood, S. G. Farkoush, T. Khurshaid, C.-H. Kim, J. Yu, Z. Geem, and S.-B. Rhee, "An optimized protection coordination scheme for the optimal coordination of overcurrent relays using a nature-inspired root tree algorithm," *Appl. Sci.*, vol. 8, no. 9, p. 1664, 2018. [Online]. Available: https://www.mdpi.com/2076-3417/8/9/1664

[36] S. Mirjalili, "The ant lion optimizer," *Adv. Eng. Softw.*, vol. 83, pp. 80–98, May 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0965997815000113

[37] S. Mirjalili, P. Jangir, and S. Saremi, "Multi-objective ant lion optimizer: A multi-objective optimization algorithm for solving engineering problems," *Int. J. Speech Technol.*, vol. 46, no. 1, pp. 79–95, Jan. 2017, doi: 10.1007/s10489-016-0825-8.

[38] M. Raju, L. C. Saikia, and N. Sinha, "Automatic generation control of a multi-area system using ant lion optimizer algorithm based PID plus second order derivative controller," *Int. J. Electr. Power Energy Syst.*, vol. 80, pp. 52–63, Sep. 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0142061516000491

[39] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," *Knowl.-Based Syst.*, vol. 89, pp. 228–249, Nov. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950705115002580

[40] W. Yamany, M. Fawzy, A. Tharwat, and A. E. Hassanien, "Moth-flame optimization for training multi-layer perceptrons," in *Proc. 11th Int. Comput. Eng. Conf. (ICENCO)*, Dec. 2015, pp. 267–272.

[41] Y.-J. Zheng, "Water wave optimization: A new nature-inspired metaheuristic," *Comput. Oper. Res.*, vol. 55, pp. 1–11, Mar. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0305054814002652

[42] S. Mirjalili, S. Saremi, S. M. Mirjalili, and L. D. S. Coelho, "Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization," *Expert Syst. Appl.*, vol. 47, pp. 106–119, Apr. 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417415007435

[43] S. Mirjalili, "How effective is the grey wolf optimizer in training multi-layer perceptrons," *Int. J. Speech Technol.*, vol. 43, no. 1, pp. 150–161, Jul. 2015, doi: 10.1007/s10489-014-0645-7.

[44] S. Saremi, S. Z. Mirjalili, and S. M. Mirjalili, "Evolutionary population dynamics and grey wolf optimizer," *Neural Comput. Appl.*, vol. 26, no. 5, pp. 1257–1263, Jul. 2015, doi: 10.1007/s00521-014-1806-7.

[45] S. Mirjalili, S. M. Mirjalili, and A. Hatamlou, "Multi-verse optimizer: A nature-inspired algorithm for global optimization," *Neural Comput. Appl.*, vol. 27, no. 2, pp. 495–513, Feb. 2016, doi: 10.1007/s00521-015-1870-7.

[46] M. M. Al-Rifaie, "Dispersive flies optimisation," in *Proc. Federated Conf. Comput. Sci. Inf. Syst.*, Sep. 2014, pp. 529–538.

[47] S. Fidanova, Ed., *Dispersive Flies Optimisation and Medical Imaging*. Cham, Switzerland: Springer, 2016.

[48] A. H. Gandomi and A. H. Alavi, "Krill herd: A new bio-inspired optimization algorithm," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 17, no. 12, pp. 4831–4845, Dec. 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1007570412002171

[49] G.-G. Wang, A. H. Gandomi, and A. H. Alavi, "Stud krill herd algorithm," *Neurocomputing*, vol. 128, pp. 363–370, Mar. 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231213009193

[50] G.-G. Wang, L. Guo, A. H. Gandomi, G.-S. Hao, and H. Wang, "Chaotic krill herd algorithm," *Inf. Sci.*, vol. 274, pp. 17–34, Aug. 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025514002291

[51] G.-G. Wang, A. H. Gandomi, A. H. Alavi, and G.-S. Hao, "Hybrid krill herd algorithm with differential evolution for global numerical optimization," *Neural Comput. Appl.*, vol. 25, no. 2, pp. 297–308, Aug. 2014, doi: 10.1007/s00521-013-1485-9.

[52] G.-G. Wang, A. H. Gandomi, and A. H. Alavi, "An effective krill herd algorithm with migration operator in biogeography-based optimization," *Appl. Math. Model.*, vol. 38, nos. 9–10, pp. 2454–2462, May 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0307904X13006756

[53] X.-S. Yang, "Flower pollination algorithm for global optimization," in *Unconventional Computation and Natural Computation*, J. Durand-Lose and N. Jonoska, Eds. Berlin, Germany: Springer, 2012, pp. 240–249.

[54] X.-S. Yang, M. Karamanoglu, and X. He, "Multi-objective flower algorithm for optimization," *Procedia Comput. Sci.*, vol. 18, pp. 861–868, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050913003943

[55] X.-S. Yang, M. Karamanoglu, and X. He, "Flower pollination algorithm: A novel approach for multiobjective optimization," *Eng. Optim.*, vol. 46, no. 9, pp. 1222–1237, Sep. 2014, doi: 10.1080/0305215X.2013.832237.

[56] W.-T. Pan, "A new fruit fly optimization algorithm: Taking the financial distress model as an example," *Knowl.-Based Syst.*, vol. 26, pp. 69–74, Feb. 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950705111001365

[57] L. Wang, X.-L. Zheng, and S.-Y. Wang, "A novel binary fruit fly optimization algorithm for solving the multidimensional knapsack problem," *Knowl.-Based Syst.*, vol. 48, pp. 17–23, Aug. 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950705113001159

[58] J. R. González, D. A. Pelta, C. Cruz, G. Terrazas, and N. Krasnogor, Eds., *A New Metaheuristic Bat-Inspired Algorithm*. Berlin, Germany: Springer, 2010.

[59] X. Yang and A. H. Gandomi, "Bat algorithm: A novel approach for global engineering optimization," *Eng. Computations*, vol. 29, no. 5, pp. 464–483, Jul. 2012, doi: 10.1108/02644401211235834.

[60] A. H. Gandomi and X.-S. Yang, "Chaotic bat algorithm," *J. Comput. Sci.*, vol. 5, no. 2, pp. 224–232, Mar. 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877750313001099

[61] S. Mirjalili, S. M. Mirjalili, and X.-S. Yang, "Binary bat algorithm," *Neural Comput. Appl.*, vol. 25, nos. 3–4, pp. 663–681, Sep. 2014, doi: 10.1007/s00521-013-1525-5.

[62] X.-S. Yang, "Bat algorithm: Literature review and applications," 2013, *arXiv:1308.3900*. [Online]. Available: http://arxiv.org/abs/1308.3900

[63] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *Stochastic Algorithms: Foundations and Applications*, O. Watanabe and T. Zeugmann, Eds. Berlin, Germany: Springer, 2009, pp. 169–178.

[64] X. Yang, "Firefly algorithm, Lévy flights and global optimization," in *Research and Development in Intelligent Systems XXVI*, M. Bramer, R. Ellis, and M. Petridis, Eds. London, U.K.: Springer, 2010, pp. 209–218.

[65] A. H. Gandomi, X.-S. Yang, S. Talatahari, and A. H. Alavi, "Firefly algorithm with chaos," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 18, no. 1, pp. 89–98, Jan. 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1007570412002717

[66] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proc. World Congr. Nature Biologically Inspired Comput. (NaBIC)*, Dec. 2009, pp. 210–214.

[67] P. Civicioglu and E. Besdok, "A conceptual comparison of the cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms," *Artif. Intell. Rev.*, vol. 39, no. 4, pp. 315–346, Apr. 2013, doi: 10.1007/s10462-011-9276-0.

[68] X.-S. Yang and S. Deb, "Multiobjective cuckoo search for design optimization," *Comput. Oper. Res.*, vol. 40, no. 6, pp. 1616–1624, Jun. 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0305054811002905

[69] A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems," *Eng. with Comput.*, vol. 29, no. 1, pp. 17–35, Jan. 2013, doi: 10.1007/s00366-011-0241-y.

[70] X.-S. Yang and S. Deb, "Cuckoo search: Recent advances and applications," *Neural Comput. Appl.*, vol. 24, no. 1, pp. 169–174, Jan. 2014, doi: 10.1007/s00521-013-1367-1.

[71] M. H. Sulaiman, Z. Mustaffa, M. M. Saari, and H. Daniyal, "Barnacles mating optimizer: A new bio-inspired algorithm for solving engineering optimization problems," *Eng. Appl. Artif. Intell.*, vol. 87, Jan. 2020, Art. no. 103330. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0952197619302775

[72] S. H. S. Moosavi and V. K. Bardsiri, "Poor and rich optimization algorithm: A new human-based and multi populations algorithm," *Eng. Appl. Artif. Intell.*, vol. 86, pp. 165–181, Nov. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0952197619302167

[73] H. Yapici and N. Cetinkaya, "A new meta-heuristic optimizer: Pathfinder algorithm," *Appl. Soft Comput.*, vol. 78, pp. 545–568, May 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494619301309

[74] E. H. D. V. Segundo, V. C. Mariani, and L. D. S. Coelho, "Design of heat exchangers using falcon optimization algorithm," *Appl. Thermal Eng.*, vol. 156, pp. 119–144, Jun. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1359431119301644

[75] C. E. Klein and L. dos Santos Coelho, "Meerkats-inspired algorithm for global optimization problems," in *Proc. ESANN*, 2018, pp. 679–684.

[76] H. Shayanfar and F. S. Gharehchopogh, "Farmland fertility: A new metaheuristic algorithm for solving continuous optimization problems," *Appl. Soft Comput.*, vol. 71, pp. 728–746, Oct. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494618304216

[77] J. Pierezan and L. D. S. Coelho, "Coyote optimization algorithm: A new metaheuristic for global optimization problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2018, pp. 1–8.

[78] E. H. de Vasconcelos Segundo, V. C. Mariani, and L. D. S. Coelho, "Metaheuristic inspired on owls behavior applied to heat exchangers design," *Thermal Sci. Eng. Prog.*, vol. 14, Dec. 2019, Art. no. 100431. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2451904919300162

[79] A. Mortazavi, V. Toğan, and A. Nuhoğlu, "Interactive search algorithm: A new hybrid metaheuristic optimization algorithm," *Eng. Appl. Artif. Intell.*, vol. 71, pp. 275–292, May 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0952197618300514

[80] C. E. Klein, V. C. Mariani, and L. dos Santos Coelho, "Cheetah based optimization algorithm: A novel swarm intelligence paradigm," in *Proc. ESANN*, 2018, pp. 25–27.

[81] V. Muthiah-Nakarajan and M. M. Noel, "Galactic swarm optimization: A new global optimization metaheuristic inspired by galactic motion," *Appl. Soft Comput.*, vol. 38, pp. 771–787, Jan. 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494615006742

[82] X.-S. Yang, S. Deb, S. Fong, X. He, and Y.-X. Zhao, "From swarm intelligence to metaheuristics: Nature-inspired optimization algorithms," *Computer*, vol. 49, no. 9, pp. 52–59, Sep. 2016.

[83] W. B. Powell, "A unified framework for stochastic optimization," *Eur. J. Oper. Res.*, vol. 275, no. 3, pp. 795–821, Jun. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221718306192

[84] M. Basu, "Squirrel search algorithm for multi-region combined heat and power economic dispatch incorporating renewable energy sources," *Energy*, vol. 182, pp. 296–305, Sep. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360544219312198

[85] A. Singh and V. Sharma, "Salp swarm algorithm-based model predictive controller for frequency regulation of solar integrated power system," *Neural Comput. Appl.*, vol. 31, no. 12, pp. 8859–8870, Dec. 2019, doi: 10.1007/s00521-019-04422-3.

[86] H. M. Hasanien and A. A. El-Fergany, "Salp swarm algorithm-based optimal load frequency control of hybrid renewable power systems with communication delay and excitation cross-coupling effect," *Electr. Power Syst. Res.*, vol. 176, Nov. 2019, Art. no. 105938. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378779619302573

[87] R. Manuel and G. Emayavaramban, "PALONN: Parallel ant lion optimizer and artificial neural network for power flow control of the micro grid-connected system," *IETE J. Res.*, 2019, doi: 10.1080/03772063.2019.1644208.

[88] F. Z. Alazemi and A. Y. Hatata, "Ant lion optimizer for optimum economic dispatch considering demand response as a visual power plant," *Electr. Power Compon. Syst.*, vol. 47, nos. 6–7, pp. 629–643, 2019. [Online]. Available: https://www.tandfonline.com/toc/uemp20/47/6-7?nav=tocList

[89] H. Buch and I. N. Trivedi, "An efficient adaptive moth flame optimization algorithm for solving large-scale optimal power flow problem with POZ, multifuel and valve-point loading effect," *Iranian J. Sci. Technol., Trans. Electr. Eng.*, vol. 43, no. 4, pp. 1031–1051, Dec. 2019, doi: 10.1007/s40998-019-00211-9.

[90] S. Mohseni, A. C. Brent, and D. Burmester, "A demand response-centred approach to the long-term equipment capacity planning of grid-independent micro-grids optimized by the moth-flame optimization algorithm," *Energy Convers. Manage.*, vol. 200, Nov. 2019, Art. no. 112105. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0196890419311112

[91] M. K. Sattar, A. Ahmad, S. Fayyaz, S. S. Ul Haq, and M. S. Saddique, "Ramp rate handling strategies in dynamic economic load dispatch (DELD) problem using grey wolf optimizer (GWO)," *J. Chin. Inst. Eng.*, vol. 43, no. 2, pp. 200–213, Feb. 2020.

[92] K. Zhang, J. Ma, X. Zhao, X. Liu, and Y. Zhang, "Parameter identification and state of charge estimation of NMC cells based on improved ant lion optimizer," *Math. Problems Eng.*, vol. 2019, pp. 1–18, Jul. 2019.

[93] H. Hu, L. Zhang, Y. Bai, P. Wang, and X. Tan, "A hybrid algorithm based on squirrel search algorithm and invasive weed optimization for optimization," *IEEE Access*, vol. 7, pp. 105652–105668, 2019.

[94] C. Lu, L. Gao, Q. Pan, X. Li, and J. Zheng, "A multi-objective cellular grey wolf optimizer for hybrid flowshop scheduling problem considering noise pollution," *Appl. Soft Comput.*, vol. 75, pp. 728–749, Feb. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494618306756

[95] P. Yao, H. Wang, and H. Ji, "Multi-UAVs tracking target in urban environment by model predictive control and improved grey wolf optimizer," *Aerosp. Sci. Technol.*, vol. 55, pp. 131–143, Aug. 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1270963816301869

[96] Y. Zhang, Z. Deng, and A. Hu, "Study on the user density identification via improved whale optimization algorithm in Device-to-Device communication," *Complexity*, vol. 2019, pp. 1–9, Nov. 2019.

[97] A. Farshin and S. Sharifian, "A chaotic grey wolf controller allocator for software defined mobile network (SDMN) for 5th generation of cloud-based cellular systems (5G)," *Comput. Commun.*, vol. 108, pp. 94–109, Aug. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366416306454

[98] P. Wang, Y. Kong, X. He, M. Zhang, and X. Tan, "An improved squirrel search algorithm for maximum likelihood DOA estimation and application for MEMS vector hydrophone array," *IEEE Access*, vol. 7, pp. 118343–118358, 2019.

[99] T. A. A. Ali, Z. Xiao, J. Sun, S. Mirjalili, V. Havyarimana, and H. Jiang, "Optimal design of IIR wideband digital differentiators and integrators using salp swarm algorithm," *Knowl.-Based Syst.*, vol. 182, Oct. 2019, Art. no. 104834. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950705119303090

[100] A. Lipare, D. R. Edla, and V. Kuppili, "Energy efficient load balancing approach for avoiding energy hole problem in WSN using grey wolf optimizer with novel fitness function," *Appl. Soft Comput.*, vol. 84, Nov. 2019, Art. no. 105706. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494619304879

[101] S. Wang, H. Jia, and X. Peng, "Modified salp swarm algorithm based multilevel thresholding for color image segmentation," *Math. Biosci. Eng.*, vol. 17, no. 1, pp. 700–724, 2020.

[102] M. Wang, L. Gao, X. Huang, Y. Jiang, and X. Gao, "A texture classification approach based on the integrated optimization for parameters and features of Gabor filter via hybrid ant lion optimizer," *Appl. Sci.*, vol. 9, no. 11, p. 2173, 2019.

[103] Y. Yang, B. Zhang, Q. Feng, H. Cai, M. Jiang, K. Zhou, F. Li, S. Liu, and X. Li, "Towards locating time-varying indoor particle sources: Development of two multi-robot olfaction methods based on whale optimization algorithm," *Building Environ.*, vol. 166, Dec. 2019, Art. no. 106413. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360132319306237

[104] M. Mitić, N. Vuković, M. Petrović, and Z. Miljković, "Chaotic metaheuristic algorithms for learning and reproduction of robot motion trajectories," *Neural Comput. Appl.*, vol. 30, no. 4, pp. 1065–1083, Aug. 2018, doi: 10.1007/s00521-016-2717-6.

[105] A. M. Al-Zoubi, H. Faris, J. Alqatawna, and M. A. Hassonah, "Evolving support vector machines using whale optimization algorithm for spam profiles detection on online social networks in different lingual contexts," *Knowl.-Based Syst.*, vol. 153, pp. 91–104, Aug. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950705118301928

[106] A. Zareie, A. Sheikhahmadi, and M. Jalili, "Identification of influential users in social network using gray wolf optimization algorithm," *Expert Syst. Appl.*, vol. 142, Mar. 2020, Art. no. 112971. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S095741741930689X

[107] S. Amirsadri, S. J. Mousavirad, and H. Ebrahimpour-Komleh, "A levy flight-based grey wolf optimizer combined with back-propagation algorithm for neural network training," *Neural Comput. Appl.*, vol. 30, no. 12, pp. 3707–3720, Dec. 2018, doi: 10.1007/s00521-017-2952-5.

[108] Y. Wang, D. Shang, and X. Yuan, "A correction method for the proportion of key components in basic hysys library based on an improved squirrel search algorithm," in *Proc. 12th Asian Control Conf. (ASCC)*, Jun. 2019, pp. 236–241.

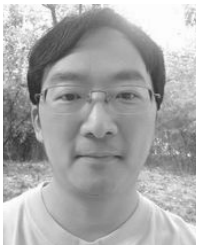
[109] S. Gong, W. Gao, and F. Abza, "Brain tumor diagnosis based on artificial neural network and a chaos whale optimization algorithm," *Comput. Intell.*, vol. 36, no. 1, pp. 259–275, Feb. 2020, [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/coin.12259, doi: 10.1111/coin.12259.

[110] C. Chen and L. Yu, "A hybrid ant lion optimizer with improved Nelder–Mead algorithm for structural damage detection by improving weighted trace lasso regularization," *Adv. Struct. Eng.*, vol. 23, no. 3, pp. 468–484, Feb. 2020, doi: 10.1177/1369433219872434.

[111] A. Benamor, M. T. Benchouia, K. Srairi, and M. E. H. Benbouzid, "A new rooted tree optimization algorithm for indirect power control of wind turbine based on a doubly-fed induction generator," *ISA Trans.*, vol. 88, pp. 296–306, May 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0019057818304580

[112] A. Wadood, T. Khurshaid, S. G. Farkoush, C.-H. Kim, and S.-B. Rhee, "A bio-inspired rooted tree algorithm for optimal coordination of overcurrent relays," in *Intelligent Technologies and Applications*, I. S. Bajwa, F. Kamareddine, and A. Costa, Eds. Singapore: Springer, 2019, pp. 188–201.

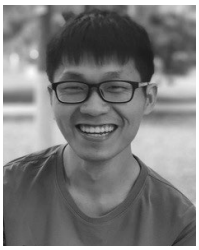
[113] J. Dash, B. Dam, and R. Swain, "Improved firefly algorithm based optimal design of special signal blocking IIR filters," *Measurement*, vol. 149, Jan. 2020, Art. no. 106986. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0263224119308528
[114] D. Ma, J. Gao, Z. Zhang, and Q. Wang, "An improved firefly algorithm for gas emission source parameter estimation in atmosphere," *IEEE Access*, vol. 7, pp. 111923–111930, 2019.
[115] W. Sun and J. Sun, "Daily PM 2.5 concentration prediction based on principal component analysis and LSSVM optimized by cuckoo search algorithm," *J. Environ. Manage.*, vol. 188, pp. 144–152, Mar. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0301479716309835
[116] A. Mojtahedi, H. Hokmabady, S. S. Z. Abyaneh, and H. Nassiraei, "Establishment of a hybrid Fuzzy–Krill herd approach for novelty detection applied to damage classification of offshore jacket-type structures," *J. Mar. Sci. Technol.*, vol. 24, no. 3, pp. 812–829, Sep. 2019, doi: 10.1007/s00773-018-0589-4.
[117] N. Kaur, S. Sharma, and J. Kaur, "Performance comparison of evolutionary algorithms in the design of a hand-pump shape microstrip antenna for 5G applications," *Elektronika ir Elektrotechnika*, vol. 25, no. 5, pp. 31–36, 2019.
[118] D. Saravanan, S. Janakiraman, K. Chandraprabha, T. Kalaipriyan, R. S. Raghav, and S. Venkatesan, "Augmented powell-based krill herd optimization for roadside unit deployment in vehicular ad hoc networks," *J. Test. Eval.*, vol. 47, no. 6, Nov. 2019, Art. no. 20180494.
[119] A. Ghosh and N. Chakraborty, "Cascaded cuckoo search optimization of router placement in signal attenuation minimization for a wireless sensor network in an indoor environment," *Eng. Optim.*, vol. 51, no. 12, pp. 2127–2146, Dec. 2019.
[120] M. Veni and T. Meyyappan, "Digital image watermark embedding and extraction using oppositional fruit fly algorithm," *Multimedia Tools and Applications*, vol. 78, no. 19, pp. 27491–27510, Oct. 2019, doi: 10.1007/s11042-019-7650-0.
[121] K. G. Dhal, J. Gálvez, and S. Das, "Toward the modification of flower pollination algorithm in clustering-based image segmentation," *Neural Comput. Appl.*, vol. 32, no. 8, pp. 3059–3077, Apr. 2020, doi: 10.1007/s00521-019-04585-z.
[122] S. Ouadfel and A. Taleb-Ahmed, "Social spiders optimization and flower pollination algorithm for multilevel image thresholding: A performance study," *Expert Syst. Appl.*, vol. 55, pp. 566–584, Aug. 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417416300550
[123] H. Tang, W. Sun, H. Yu, A. Lin, and M. Xue, "A multirobot target searching method based on bat algorithm in unknown environments," *Expert Syst. Appl.*, vol. 141, Mar. 2020, Art. no. 112945. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417419306633
[124] N. Nedic, V. Stojanovic, and V. Djordjevic, "Optimal control of hydraulically driven parallel robot platform based on firefly algorithm," *Nonlinear Dyn.*, vol. 82, no. 3, pp. 1457–1473, Nov. 2015, doi: 10.1007/s11071-015-2252-5.
[125] P. K. Mohanty and D. R. Parhi, "Optimal path planning for a mobile robot using cuckoo search algorithm," *J. Experim. Theor. Artif. Intell.*, vol. 28, nos. 1–2, pp. 35–52, Mar. 2016.
[126] P. G. Asteris, S. Nozhati, M. Nikoo, L. Cavaleri, and M. Nikoo, "Krill herd algorithm-based neural network in structural seismic reliability evaluation," *Mech. Adv. Mater. Struct.*, vol. 26, no. 13, pp. 1146–1153, Jul. 2019.
[127] N. S. Jaddi, S. Abdullah, and A. R. Hamdan, "Optimization of neural network model using modified bat-inspired algorithm," *Appl. Soft Comput.*, vol. 37, pp. 71–86, Dec. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494615004950
[128] C.-H. Yang, Y.-S. Lin, L.-Y. Chuang, and H.-W. Chang, "Rooted tree optimization algorithm for protein folding prediction," *J. Life Sci. Technol.*, vol. 4, no. 2, pp. 89–93, Jan. 2016.
[129] F. Guo, L. Ren, Y. Jin, and Y. Ding, "A dynamic SVR–ARMA model with improved fruit fly algorithm for the nonlinear fiber stretching process," *Natural Comput.*, vol. 18, no. 4, pp. 747–756, Dec. 2019, doi: 10.1007/s11047-016-9601-2.
[130] J. P. Ram and N. Rajasekar, "A new global maximum power point tracking technique for solar photovoltaic (PV) system under partial shading conditions (PSC)," *Energy*, vol. 118, pp. 512–525, Jan. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360544216315183
[131] L. Dekhici, R. Redjem, K. Belkadi, and A. El Mhamedi, "Discretization of the firefly algorithm for home care," *Can. J. Electr. Comput. Eng.*, vol. 42, no. 1, pp. 20–26, 2019.
[132] G.-G. Wang, A. H. Gandomi, X. Zhao, and H. C. E. Chu, "Hybridizing harmony search algorithm with cuckoo search for global numerical optimization," *Soft Comput.*, vol. 20, no. 1, pp. 273–285, Jan. 2016, doi: 10.1007/s00500-014-1502-7.
[133] M. Stephenson, S. Amarasinghe, M. Martin, and U.-M. O'Reilly, "Meta optimization: Improving compiler heuristics with machine learning," *ACM SIGPLAN Notices*, vol. 38, no. 5, p. 77, May 2003, doi: 10.1145/780822.781141.



HUI LI (Member, IEEE) received the B.S. degree from Xi'an Jiaotong University, in 1994, and the M.S. and Ph.D. degrees from Shanghai Jiaotong University, in 1997 and 1999, respectively.

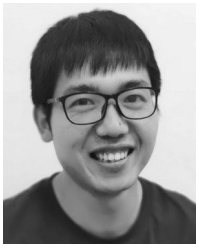
He has been a Visiting Scholar with the University of California, Davis, Davis, CA, USA, in 2013. He is currently an Associate Professor with the Department of Computer Science and Technology, Beijing University of Chemical Technology. His research interests include number theory, algebra, complexity, and artificial intelligence.

XIAO LIU received the B.S. degree from the Beijing University of Chemical Technology, Beijing, China, in 2017. She is currently a Graduate Student with the Beijing University of Chemical Technology.

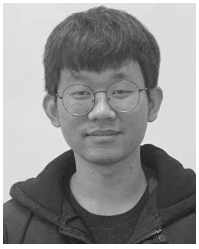


ZHIGUO HUANG received the B.S. degree from the Beijing University of Chemical Technology, Beijing, China, in 2017. He is currently a Graduate Student with the Beijing University of Chemical Technology.

CHENBO ZENG received the B.S. degree from the Beijing University of Chemical Technology, Beijing, China, in 2017. She is currently a Graduate Student with the Beijing University of Chemical Technology.



PENG ZOU received the B.S. degree from the Beijing University of Chemical Technology, Beijing, China, in 2017. He is currently a Graduate Student with the Beijing University of Chemical Technology.



ZHAOYI CHU received the B.S. degree from the Beijing University of Chemical Technology, Beijing, China, in 2018. He is currently a Graduate Student with the Beijing University of Chemical Technology.

JUNKAI YI received the B.S. and Ph.D. degrees from the Beijing Institute of Technology, in 1993 and 1998, respectively. He is currently a Professor with the Beijing Information Science and Technology University.