Electronic Research Archive

http://www.aimspress.com/journal/era

ERA, 32(4): 2380–2406.
DOI: 10.3934/era.2024109
Received: 27 December 2023
Revised: 29 February 2024
Accepted: 11 March 2024
Published: 25 March 2024

Research article

A new hybrid Lévy Quantum-behavior Butterfly Optimization Algorithm and its application in NL5 Muskingum model

Hanbin Liu[1], Libin Liu[1], Xiongfa Mai[1,*] and Delong Guo[2]

[1] Center for Applied Mathematics of Guangxi, Nanning Normal University, Nanning 530100, China
[2] School of Mathematics and Statistics, Qiannan Normal University, Duyun 558000, China

* **Correspondence:** Email: maixf@nnnu.edu.cn.

Abstract: This paper presents a novel hybrid algorithm that combines the Butterfly Optimization Algorithm (BOA) and Quantum-behavior Particle Swarm Optimization (QPSO) algorithms, leveraging *gbest* to establish an algorithm communication channel for cooperation. Initially, the population is split into two equal subgroups optimized by BOA and QPSO respectively, with the latter incorporating the Lévy flight for enhanced performance. Subsequently, a hybrid mechanism comprising a weight hybrid mechanism, a elite strategy, and a diversification mechanism is introduced to blend the two algorithms. Experimental evaluation on 12 benchmark test functions and the Muskin model demonstrates that the synergy between BOA and QPSO significantly enhances algorithm performance. The hybrid mechanism further boosts algorithm performance, positioning the new algorithm as a high-performance method. In the Muskingum model experiment, the algorithm proposed in this article can give the best sum of the square of deviation (SSQ) and is superior in the comparison of other indicators. Overall, through benchmark test function experiments and Muskin model evaluations, it is evident that the algorithm proposed in this paper exhibits strong optimization capabilities and is effective in addressing practical problems.

Keywords: Quantum-behavior Particle Swarm Optimization; Butterfly Optimization Algorithm; hybrid algorithm; NL5 Muskingum model

1. Introduction

Intelligent optimization algorithms are random search algorithms based on biological habits and natural phenomena. In recent years, intelligent optimization algorithms have been applied in many fields, such as medicine, engineering, and so on. Metanephritic algorithms can be classified into three main types: biological evolution, natural phenomena, and species' living habits. Biological evolution methods, like Genetic Algorithms (GA) [1] and Differential Evolution (DE) [2], are inspired by biological genetics,

mutation, and evolution strategies. Natural phenomenon algorithms are based on the physical laws of nature, such as Sine Cosine Algorithm (SCA) [3] and Biogeography-Based Optimizer (BBO) [4]. Population life habits algorithms are inspired by the relationship between population individuals, including Particle Swarm Optimization (PSO) [5], Cuckoo Search Algorithm (CSA) [6], and Butterfly Optimization Algorithm (BOA) [7]. Scholars continue to develop new or improved algorithms despite the existence of many algorithms already because the *No Free Lunch Theory* [8] argues logically that there is currently no optimization technique capable of solving all optimization problems. Additionally, when dealing with high-dimensional complex problems, metaheuristic algorithms tend to have slower convergence speeds and lower accuracy, making them susceptible to getting stuck in local optima [9]. Therefore, scientists persistently work on enhancing algorithms with superior performance to address experimental requirements such as modeling simulation [10], path planning [11], and robot frameworks [12].

BOA [7] is an intelligent optimization algorithm that imitates the foraging and courtship behavior of butterflies, and it has gained popularity among scientists and has been successfully applied in various fields. For example, Aygül et al. [13] applied BOA to calculate maximum power point tracking in PV systems under partial shadowing conditions. Arora and Anand [14] also used BOA to create learning automata. Li et al. [15] developed an updated version of BOA based on the cross-entropy technique to balance exploration and exploitation. However, its optimization performance significantly declines when faced with complex problems. This can be attributed to two main factors. First, there needs to be a balanced ratio between algorithmic exploration and search behavior exploitation throughout the search process. Second, the optimization process heavily relies on individual butterfly communication. When the optimal butterfly gets trapped in a local optimum, it attracts other butterflies, leading to premature convergence. In order to overcome these problems, scholars have taken some measures to improve the algorithm. Some scholars have rewritten the fragrance formula in the algorithm [16, 17], which have some extent improved the convergence speed of the algorithm. However, the convergence accuracy and robustness still need to be improved. Assiri [18] attempted to incorporate the logistic mapping into the algorithm, but did not converge to the theoretical optimal value in function testing. Li et al. [19] applied quantum bits, quantum spiral gates, and quantum non-gates to the algorithm in a non-stage manner, improving its performance, but still having certain limitations when dealing high dimension problems. BOA also has been hybridized with other metaheuristics, such as Hybrid Butterfly Flower Pollination Algortihtm (HBFPA) [20] and Butterfly Artificial Bee Colony (BABC) [21].

Besides, the emergence of quantum computation theory has greatly influenced the field of computational science. Research focusing on incorporating quantum behavior into intelligent algorithms, such as the quantum genetic algorithm [22], the quantum PSO [23], and other new metaheuristic algorithms [24], has garnered significant interest. These advancements have successfully enhanced the efficiency and performance of these algorithms and helped scholars solve practical problems [25]. One notable application is the integration of quantum computation theory into optimization methods, with QPSO [26] being a popular example. The algorithm's ability to generate particles in a probabilistic manner throughout the solution space helps it avoid getting stuck in local optima, making researchers favor this, and often using QPSO as a tool for hybrid algorithms. Kumar et al. [27] proposed Cuckoo Search Adaptive Gaussian Quantum-behavior Particle Optimization (CSAGQPSO), which combines CSA and Adaptive Gaussian Quantum-behavior Particle Optimization (AGQPSO), a hybrid of two improved versions of QPSO, and applied it to ordinary differential equations. In CSAGQPSO, CSA and AGQPSO respectively undertake development and exploration tasks, which makes the algorithm simpler while having good

performance. Mai et al. [28] combined QPSO and CSA, designing hybrid mechanisms to create a new hybrid Cuckoo-Quantum-behavior Particle Swarm Optimization (C-QPSO), successfully estimating the parameters of the Muskingum model. C-QPSO performs well and stands out in comparison with several improved algorithms, but the running cost of the algorithm is relatively high because there are many improvements in it. Wang et al. [29] introduced a novel QPSO variant by guiding it with the length of the potential well (LPW) and assisting it with Adam (abbreviated as Adam-LGQPSO), a combination of QPSO and Adam that enhances the algorithm's efficiency. The algorithm introduces the Adam algorithm in the initialization stage to improve the quality of the population and improve the optimization quality, but the effect is not very good in benchmark function experiments.

Despite its effectiveness, there is still room for improvement in terms of convergence accuracy. One tool that has shown promise in enhancing algorithm optimization performance is the concept of Lévy flight. Lévy flight is a random walk process that combines both local and global search strategies. In nature, many animals employ a mix of short and long distance searches to find food in unpredictable environments. Inspired by this, researchers have been attracted to the Lévy flight strategy and incorporated the idea of Lévy flight into swarm intelligence algorithms. Ling et al. [30] proposed an enhancement to the Wolf Optimization Algorithm method by integrating Lévy flight, which improved the global optimization capability of Lévy Wolf Optimization Algorithm. Zhong et al. [31] introduced the opposition-based learning Equilibrium Optimizer algorithm, which combines Lévy flight and evolutionary population dynamics to tackle high-dimensional global optimization problems. They have found that by allowing the algorithms to explore candidate solutions that are far away from the current optimal solution, the search space expands and the global search capability is enhanced, resulting in improved performance.

The hybrid algorithm resulting from the combination is more likely to incorporate the strengths of the two original algorithms while mitigating their weaknesses. Apart from HBFPA [20], BABC [21], CSAGQPSO [27], C-QPSO [28], and Adam-LGQPSO [29] mentioned previously, there are other successful instances as well [32, 33]. By examining these successful cases, it is argued that the design of hybrid algorithms should emphasize the cooperative work between algorithms, with a key focus on inter-algorithm communication. The update mechanism of QPSO and BOA is simple and effective, which has a certain advantage in the design of hybrid algorithms. Second, in the update mechanisms of QPSO and BOA, the individual's position update is influenced by the *gbest* point, promoting the cooperative work between the two algorithms and enabling them to leverage each other's strengths while minimizing their limitations. In addition, to maintain the computational speed of the algorithm, we divide the population into two equally large subgroups during the initialization phase, and design a hybrid mechanism to fuse the population during the update phase to further strengthen communication between algorithms.

Therefore, in this article, we propose a new hybrid algorithm that combines BOA and QPSO and uses Lévy flight, and we apply it to evaluate the parameters of the Muskingum model. BOA is an algorithm with a fast convergence rate, but it is prone to local optima, while QPSO is an algorithm with good global search ability, and the two can form functional complementarity. Specifically, the algorithm divides the population into two equally sized subgroups during the initialization phase, optimizes them through two algorithms, and then mixes the subgroups through a weight mixing mechanism during the optimization phase. In brief, the contributions of this paper are as follows:

- We propose a hybrid algorithm to estimate the parameters of the nonlinear five parameters Musk-

ingum model, by combining both advantages of QPSO and BOA.
- We conduct extensive simulation experiments to evaluate our proposed algorithm, and the results show that our algorithm has better performance than some classic and recent offloading algorithms.
- We formulate the Muskinian model as parameter estimation problems of numerical solution to a differential equation, use new algorithms to evaluate model parameters, and improve the fit between evaluation values and observed values.

It should be pointed out that there has tremendous interest in developing hybrid intelligent algorithms to estimate the parameters of Muskingum models [34]. For example, Ouyang et al. [35] proposed a hybrid approach that combined PSO with the Nelder-Mead simplex (NMS) algorithm to optimize the parameters of the Muskingum model. Unlike traditional methods, the hybrid PSO-NMS approach does not require initial values for each parameter. Okkan and Kirdemir [36] combined the PSO algorithm with the Levenberg-Marquardt (LM) algorithm to estimate the parameters of a nonlinear Muskingum model with three parameters. Akbari and Hessami-Kermani [37] utilized the PSO-GA algorithm to optimize Muskingum parameters. It is important to note that the parameter optimization of the Muskingum model with four parameters (NL4) [28] is still in its early stages, and research on the Muskingum model with five parameters (NL5) [38] is even more limited. Thus, we will apply the proposed L-QBOA to estimate the parameters of the NL5 Muskingum model and hope to provide an effective new method for the study of the Muskingum model.

This article will describe L-QBOA in detail in Section 2, verify the performance of the algorithm through benchmark function testing in Section 3 and apply it to the NL5 Muskingum model, and finally summarize the full text in Section 4.

2. A new hybrid algorithm

The main purpose of this paper is to present a new algorithm called L-QBOA, which is based on QPSO and BOA. The QPSO algorithm was modified and combined with BOA to create L-QBOA. To better understand L-QBOA, we will first provide a detailed description of BOA and QPSO.

2.1. Butterfly Optimization Algorithm

Butterfly Optimization Algorithm (BOA) [7] is an intelligent algorithm that simulates the feeding behavior of butterflies. In this algorithm, there are N butterflies.

$$X = [X_1, X_2, \cdots, X_N], \tag{2.1}$$

where each butterfly X_i represents a solution, $i = 1, \cdots, N$. The fitness of the butterfly is related to the solution vector of the butterfly. Butterflies can emit specific fragrances, communicate, and attract each other through the fragrances scattered in the air. Let c, I, and a be the perception coefficient, the stimulus intensity, and the power exponent, respectively. The general value of the power exponent a is 0.01. The equation of fragrance f is given by

$$f = cI^a, \tag{2.2}$$

and then c is given by

$$c^{t+1} = c^t + \frac{b}{c^t \times MaxIT}, \tag{2.3}$$

where b is a constant with a value of 0.025, and $MaxIT$ is the maximum number of iterations. Each butterfly will fly randomly or move towards the one that emits a stronger fragrance. During the search process, butterflies have two modes of movement.

First, one of the butterflies emits the strongest fragrance and attracts them by being perceived by other butterflies. This is the global search stage for butterflies:

$$X_i^{t+1} = X_i^t + \left(r^2 \times gbest\,(t) - X_i^t\right) \times f_i. \tag{2.4}$$

Second, when the scent in the air cannot be accurately perceived, they will be randomly attracted by a fragrance and move randomly. At this point, butterflies perform a local search:

$$X_i^{t+1} = X_i^t + \left(r^2 \times X_j^t - X_k^t\right) \times f_i. \tag{2.5}$$

Finally, butterflies will choose one of the modes for optimization in a probabilistic manner:

$$\begin{cases} Global \quad search, r \le p \\ Local \quad search, r > p \end{cases}. \tag{2.6}$$

In above equations, r is a random number [0, 1], p is usually set to 0.8, $gbest$ is the optimal solution in the current iteration, f_i is the fragrance of the i^{th} butterfly, and X_j^t and X_k^t respectively represent the solution vector of the j^{th} and k^{th} butterfly in the t iterations.

2.2. Modified Quantum-behavior Particle Swarm Optimization (MQPSO)

To improve the performance of QPSO [26], we add Lévy flight into QPSO by using the idea of reference [39]. Next, we give some details of the modification as follows.

In this algorithm, there are N particles:

$$Y = [Y_1, Y_2, \ldots, Y_N]\,. \tag{2.7}$$

Each particle in the algorithm has corresponding local attractors P_i:

$$P_i = \varphi\,(t) \times pbest_i\,(t) + [1 - \varphi\,(t)] \times gbest\,(t)\,, \tag{2.8}$$

where $pbest_i\,(t)$ is the particle's history optimal position, and $\varphi\,(t)$ is a random number in [0, 1].

Here, Lévy flight, intensive short-range exploration and occasional long-range search, are used to expand the algorithm's search space and enrich the optimization methods of algorithms. Let $m(t)$ represent the mean of all particle's history optimal position and u is a rand number in [0,1]. When $u(t) > 0.5$, particles will update their positions in Lévy flight:

$$Y_i^{t+1} = L(\lambda) \times \left|Y_i^t - P_i\right| \pm \alpha \left|m\,(t) - Y_i^t\right| \times \ln\left(\frac{1}{u}\right), \tag{2.9}$$

where α, called the contraction-expansion coefficient, decreases linearly from 1 to 0.5, and $L(\lambda)$ is Lévy flight:

$$L(\lambda) = 0.01 \times \frac{r_a \times \sigma}{|r_b|^{\frac{1}{\beta}}}, \tag{2.10}$$

where r_a and r_b are two standard normally distributed random numbers, β is a constant equal to 1.5, and σ is given by

$$\sigma = \left(\frac{\Gamma(1+\beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\frac{\beta-1}{2}}} \right)^{\frac{1}{\beta}}. \tag{2.11}$$

$\Gamma(x)$ denotes the Gamma function. When $u(t) \leq 0.5$, the i^{th} particle's position is renewed by

$$Y_i^{t+1} = P_i \pm \alpha \times \left| m(t) - Y_i^t \right| \times \ln\left(\frac{1}{u}\right). \tag{2.12}$$

2.3. Lévy Quantum-behavior Butterfly Optimization Algorithm

The butterfly optimization algorithm exhibits a strong evolutionary mechanism. However, it also shares some limitations with other general metaheuristic algorithms. These include a slow rate of convergence, low precision, and a tendency to get trapped in local optima. To address these shortcomings, many hybrid algorithms have been developed to enhance performance by integrating the strengths of different algorithms. In this study, we propose a new algorithm, the Lévy Quantum-behavior Butterfly Optimization Algorithm (L-QBOA), by combining the above two algorithms and designing hybrid mechanisms. The Lévy quantum-behavior butterfly optimization algorithm will be explained in detail below.

In this method, the populations are updated by using BOA and MQPSO. First, the swarm is randomly divided into two equally sized sub-swarms X and Y during the initialization phase, and will not be re-divided thereafter. Then, swarm X is updated by BOA and swarm Y is updated using MQPSO, and their membership relationships remain unchanged in the subsequent iteration process. Later, L-QBOA performs the following mechanisms (weight hybrid mechanism, elite strategy and diversification mechanism) to continue the optimization work in the optimization phase.

2.3.1. Weight hybrid mechanism

The algorithm divides the population into two subgroups during the initialization phase and optimizes them through two algorithms. To enhance collaboration between algorithms, a combination method is planned to be constructed during the optimization phase. Several combination ideas have been considered: first, sequentially performing different meta-heuristic algorithms overall; second, sequentially performing different meta-heuristic algorithms in each iteration [40]; and third, combining basic ideas of different meta-heuristic algorithms to design a new population updating strategy [41]. The second approach is deemed more suitable for BOA and QPSO, we argue.

Different from literature [40], weights are calculated for the two algorithms before mixing, rather than random mixing. Because the algorithm proposed in this article uses the basic BOA algorithm and QPSO with Lévy flight, there may be some differences in the quality of particles optimized through the two algorithms.

In each iteration, a new particle Z is produced

$$Z_i^t = \eta_Y \times Y_i^t + \eta_X \times X_i^t, \tag{2.13}$$

where

$$\eta_Y = \frac{f_X}{f_X + f_Y}, \eta_X = \frac{f_Y}{f_X + f_Y}, \tag{2.14}$$

or

$$\eta_Y = \frac{f_Y}{f_X + f_Y}, \eta_X = \frac{f_X}{f_X + f_Y}, \tag{2.15}$$

and f_X and f_Y are the fitness of X_i^t and Y_i^t, respectively. The expressions in Eq (2.14) or (2.15) can assign weights to X in BOA and Y in MQPSO to ensure that the new particle has a certain degree of randomness while appearing in a relatively good position. For example, use Eq (2.14) in minimizing optimization problems; if Y_i^t is better, f_Y will be smaller and η_Y will be larger, allowing the information of particles with relatively good positions to be more fully retained in Z. If it is a maximization optimization problem, use Eq (2.15); if Y_i^t is better, f_Y will be larger and η_Y will be larger.

Then, Z is compared with X and Y, keeping the better ones, and *pbest* and *gbest* in the MQPSO are updated. Here, the quality of Z is not necessarily better than X and Y, so in order to ensure the convergence of the algorithm, we can not blindly retain Z, so it is necessary to choose the best of Z, X and Y particles. In addition, the existence of Z maintains the diversity of algorithm population to a certain extent, so this mechanism is multi-functional. This can lead the algorithm to develop in a positive direction.

2.3.2. Elite strategy

Whether it is BOA or MQPSO, *gbest* has always been utilized as a guiding factor in the algorithm's operation, thus the quality of its position greatly affects the algorithm's performance. To enhance the quality of *gbest*, we propose an elite strategy for its optimization. This strategy specifically targets the improvement of this position and generates a new random position near it, following a normal distribution

$$gbest_{new} = gbest + \frac{range}{\xi} \times randn\,(1, D)\,, \tag{2.16}$$

where D represents the dimension of the problem, $range = (randge_1, randge_2, \ldots, randge_D)$, and $randge_d$ ($d \in [1, D]$) are the maximum distance values of the search space for each dimension, and ξ is a larger positive number that can be adjusted according to different problems. Finally, we select the superior particle out of the two.

2.3.3. Diversification mechanism

Population diversity is an important indicator that significantly affects algorithm performance. To maintain diversity, a diversification mechanism that mixes the mechanisms of literature [39] and literature [42] is employed in L-QBOA in the second half of the algorithm optimization work. The mechanism uses NS, a counter for the number of individual stagnation generations, recording the update of the best solution *gbest*. If *gbest* remains unchanged in a given iteration, NS is incremented by one; otherwise, it is reset to zero. When the value of NS exceeds the set threshold NS_{MAX}, the algorithm determines that the population is overcrowded, resulting in a decrease in population diversity and falling into local optima. At this point, it is necessary to re-initialize ($\omega\%$) of the poorer particles to maintain population diversity. What needs to be considered is that re-initializing too many particles will reduce the convergence speed of the algorithm, but re-initializing too few particles has little effect on maintaining the diversity of the algorithm. So, we need to set the two parameters according to the actual situation.

Algorithm 1 L-QBOA

Input: Population size, Dimension, Perception coefficient, Stimulus intensity, Power exponent, Conversion probability

Output: The best fitness

Initialize the population and divided into tow sub-swarms X and Y

while termination criterion satisfied **do**

Update swarm X in two stages: Global search and Local search (Eq (2.6)), by using Eqs (2.4) and (2.5)

Update swarm Y by using Eqs (2.9) and (2.12)

Operate the weight hybrid mechanism, generate Z by using Eq (2.13)

Operate the elite strategy by using Eq (2.16) and retain the optimal

Operate the diversification mechanism

Keep the best solution *gbest*

Iterations + 1

end while

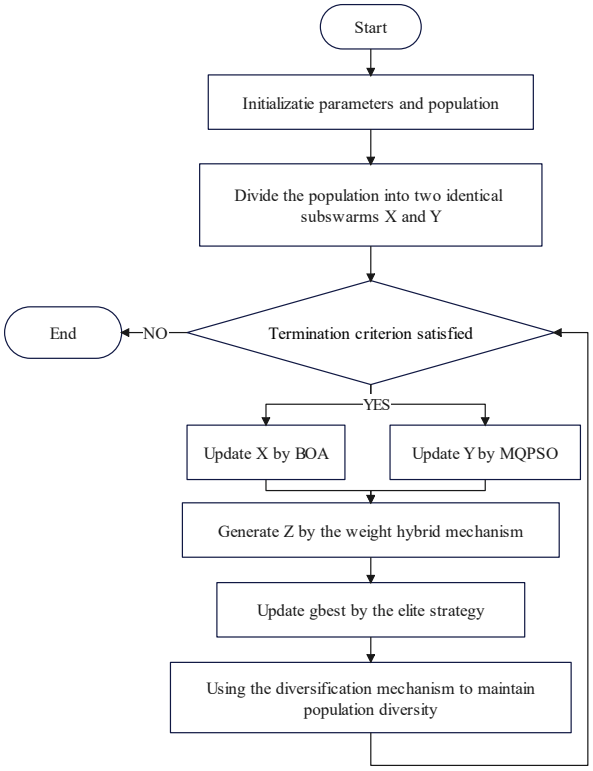


Figure 1. Flow chart of proposed L-QBOA.

In the proposed L-QBOA described in this article, two independent populations are optimized by BOA and MQPSO separately. However, both algorithms jointly optimize one *gbest*, which represents the optimal solution from the two populations. This *gbest* serves as a guiding factor for algorithm optimization, allowing the two independent algorithms to converge in the same direction. Real-time

updates to the *gbest* during the optimization process ensures algorithm convergence. Additionally, the Z particles generated in the weight hybrid mechanism consist of X and Y particles. This inclusion helps maintain population diversity to a certain extent while retaining some information from X and Y particles. The algorithm establishes communication channels between the two algorithms using the *gbest* and Z particles, enabling each algorithm to function independently while maintaining their synergy.

The pseudo code of L-QBOA is shown in Algorithm 1. Meanwhile, the flow chart of the proposed L-QBOA algorithm is shown in Figure 1.

2.4. L-QBOA performance analysis

In this part, this paper analyzes the computational complexity and memory efficiency of L-QBOA. These measure denote the time complexity and space complexity of the algorithm, respectively.

2.4.1. Computation complexity

Computational complexity is a crucial metric in computer science that measures the difficulty of algorithms by assessing their efficiency and the computational resources they require. In general, the computational complexity of an algorithm tends to increase as the problem size grows, even if the algorithm remains the same. The complexity of algorithms plays a significant role in selecting appropriate algorithms or problem solutions. Hence, conducting a computational complexity analysis is essential to determine the feasibility and practicality of an algorithm for a given problem.

L-QBOA's computational complexity consists of four main components: population initialization, fitness value calculation, swarm location update, and the weight hybrid mechanism. We denote N as the population size, D as the dimension, and T as the maximum number of iterations. During the swarm initialization phase, a random initialization generates a matrix, which requires a time complexity of $O(ND)$. Then, the fitness value of the population is calculated with a time complexity of $O(ND)$. When starting the iteration, the time complexity is proportional to T. In this phase, both the temporal complexity of fitness value calculation and location update are $O(ND)$. Finally, the weight hybrid mechanism requires a computational complexity of $O(ND)$. So, L-QBOA's computation procedure is described as

$$\begin{aligned} &O(ND) + O(ND) + O(TND) + O(TND) + O(TND) \\ =&O(TND) \ . \end{aligned} \tag{2.17}$$

The computational complexity of BOA and QPSO can be easily calculated as $O(TND)$. So, the design of the new algorithm L-QBOA does not increase computational complexity.

2.4.2. Memory efficiency

Spatial complexity is a metric that measures the memory resources used by algorithms. It quantifies the additional memory space needed during algorithm execution and how it increases with input size. When analyzing spatial complexity, we mainly consider the memory occupied by data structures, variables, pointers, and recursive calls. This analysis is essential in algorithm design and analysis as it helps evaluate resource consumption, choose suitable data structures, and optimize memory usage.

It is worth noting that the space complexity of the parameter setting of the algorithm is constant, that is, it is not affected by the complexity of the problem. The space complexity of L-QBOA is determined to be $O(N)$, where N is the size of the problem. This complexity is mainly caused by the update of the

population during iteration. The time complexity of the population is $O(N)$, so the space complexity of BOA and QPSO is $O(N)$.

In summary, based on the analysis, it can be concluded that L-QBOA and the two basic algorithms (BOA and QPSO) have similar space complexity characteristics, that is, linear space complexity. This means that the amount of additional memory space required by these algorithms grows proportionally to the problem size.

3. Evaluation and experimental results

In this section, two class of numerical experiments are given to demonstrate the performance of our proposed hybrid algorithms. The first is the benchmark function experiment, and the other part is the NL5 Muskingum Model experiment.

3.1. Experimental results of benchmark functions

All these algorithms are implemented by MATLAB R2016a and all runs are performed on a laptop with and an Intel(R) Core(TM) i5-7300HQ CPU @ 2.50 GHz, with 2*8 GB of RAM and with Windows 10 as the operating system. In the following experiments, the population size is set as 100, and the maximum number of iterations is 1000, and each operation is independently run 30 times. The parameter settings of BOA [7], MQPSO, QPSO + BOA, QSSA (2022) [43], HBA (2022) [44] and L-QBOA are shown in Table 1.

Table 1. Parametric settings of algorithms.

Algorithm	Parameters
BOA	$a = 0.01$, $b = 0.025$, $c = 0.01$
MQPSO	α linear decrease from 1 to 0.5
QPSO + BOA	$a = 0.01$, $b = 0.025$, $c = 0.01$, α linear decrease from 1 to 0.5
QSSA	$\omega_1 = 0.6$, $\omega_2 = 0.9$, $c_1 = 0.6$, $c_2 = 0.9$, $P_{dp} = 0.2$, $P = 0.5$, $sf = 32.8$
HBA	$\beta = 6$, $C = 2$
L-QBOA	$a = 0.01$, $b = 0.025$, $c = 0.01$, α linear decrease from 1 to 0.5, $\xi = 1000$, NS_{MAX}= 5 [42], ω= 10 [39]

Before explaining the experiment, let us first explain an algorithm: QPSO + BOA. Compared to L-QBOA, this algorithm does not have the weight hybrid mechanism, elite strategy, and diversification mechanism. In QPSO + BOA, the *gbest* in Eqs (2.8) and (2.4) is optimized by MQPSO and BOA jointly and BOA and MQPSO use *gbest* as the information exchange channel. The purpose of sorting out such an algorithm is to more intuitively reflect the usefulness of our designed hybrid mechanism.

As is shown in Tables 2 and 3, we select 12 benchmark functions as the test set. F1–F6 are unimodal functions, and F7–F12 are multimodal functions. The characteristics of the unimodal functions and the multimodal functions are shown in these two tables, including the function name, expression of functions, dimension, the search area, and the theoretical optimal value. In order to illustrate the advantages of our proposed L-QBOA, we use different algorithms (BOA [7], MQPSO, QPSO + BOA, QSSA (2022) [43], HBA (2022) [44] and L-QBOA) to solve the minimum value of these benchmark functions. Here, in the following numerical experiments, we will analyze the performance of these algorithms from the following five aspects: convergency accuracy comparison, convergency speed

comparison, robustness comparison, high dimension, and average CPU time.

Table 2. The description of classical unimodal functions.

Function	Name	Expression	D	Rang	Optimum
F1	Discus	$F(x) = 10^6 \times x_1^2 + \sum_{i=2}^{D} x_i^2$	30	[-100,100]	0
F2	Sphere	$F(x) = \sum_{i=1}^{D} x_i^2$	30	[-100,100]	0
F3	Sum Squares	$F(x) = \sum_{i=1}^{D} i x_i^2$	30	[-10,10]	0
F4	Bent Cigar	$F(x) = x_1^2 + \sum_{i=2}^{D} 10^6 \times x_i^2$	30	[-1.28,1.28]	0
F5	Schwefel 2.22	$F(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	30	[-10,10]	0
F6	Schwefel 1.2	$F(x) = \sum_{i=1}^{D} \left(\sum_{k=1}^{i} x_k \right)^2$	10	[-10,10]	0

Table 3. The description of classical multimodal functions.

Function	Name	Expression	D	Rang	Optimum
F7	Rastrigin	$F(x) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10)$	30	[-5.12,5.12]	0
F8	Schaffer	$F(x) = 0.5 + \frac{\sin^2(\sqrt{x_1^2+x_2^2}) - 0.5}{(1+0.001(x_1^2+x_2^2))^2}$	2	[-100,100]	0
F9	Boachevsky2	$F(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)\cos(4\pi x_2) + 0.3$	2	[-100,100]	0
F10	Boachevsky3	$F(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1 + 4\pi x_2) + 0.3$	2	[-100,100]	0
F11	Griewank	$F(x) = \frac{1}{4000}(\sum_{i=1}^{D}(x_i - 100)^2) - \prod_{i=1}^{D} \cos(\frac{x_i - 100}{\sqrt{i}}) + 1$	30	[-600,600]	0
F12	Ackley	$F(x) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + e$	30	[-32,32]	0

3.1.1. Convergency accuracy comparison

All statistical results viz, mean values (AVE), standard deviation (STD), maximum (MAX), and minimum (MIN) values of all benchmark functions and average CPU times are shown in Tables 4 and 5. The bold values in these tables represent the best values of the index in the function texts. Let us first examine the experimental data comparison between QPSO + BOA and BOA and MQPSO. The convergence accuracy of QPSO + BOA in F1–F8 is higher than that of BOA and MQPSO, which demonstrates that the combination of BOA and MQPSO can enhance the algorithm's performance. Even though MQPSO itself may be superior to BOA, their combination can further optimize each other's performance. When comparing L-QBOA and QPSO + BOA, it becomes evident that the performance of L-QBOA is significantly better. This proves that the hybrid mechanism can effectively enhance the algorithm's performance by balancing exploration and exploitation. Then we compare the performance of algorithms L-QBOA, QSSA, and HBA. Among the three algorithms, QSSA had the

lowest performance, while HBA emerged as the winner in the F6 experiment. Interestingly, L-QBOA performed the best in all experiments except for F6. In F12, MQPSSO, QPSO + BOA, QSSA, and L-QBOA behaved the same. So, it can be concluded that L-QBOA has high convergence accuracy.

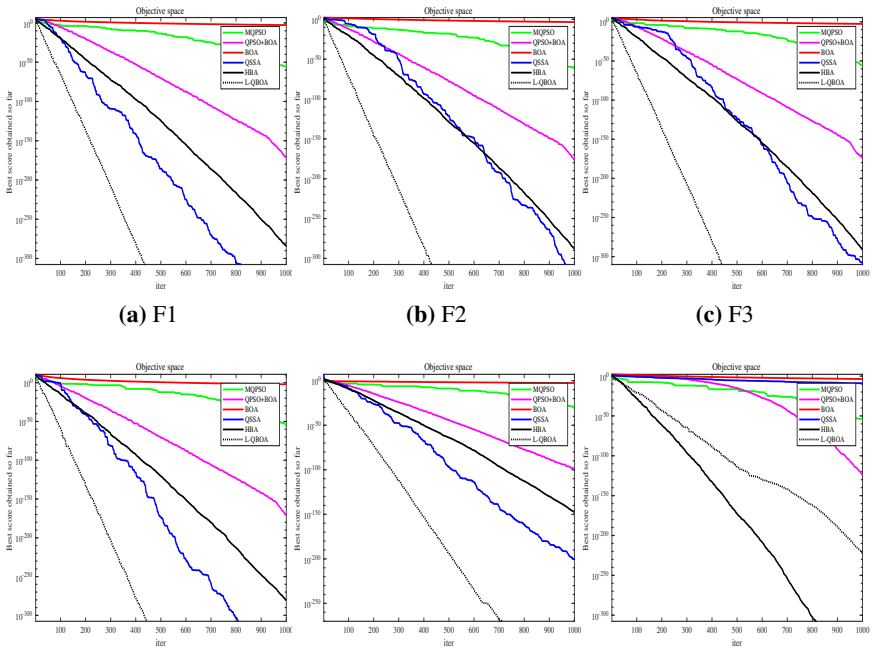
Table 4. Comparison of L-QBOA with different algorithms (F1–F6).

Function		BOA	MQPSO	QPSO + BOA	QSSA	HBA	L-QBOA
F1	AVE	4.8485e-16	5.5338e-122	3.6293e-165	1.1277e-193	1.0977e-269	**0**
	STD	4.1044e-17	2.2906e-121	**0**	**0**	**0**	**0**
	MAX	5.7260e-16	1.2477e-120	6.9485e-164	2.1418e-192	2.5657e-268	**0**
	MIN	3.9228e-16	4.0628e-135	3.9729e-174	2.4595e-267	8.1545e-292	**0**
F2	AVE	4.5843e-16	6.2144e-121	1.0462e-169	1.7462e-202	1.3149e-272	**0**
	STD	3.7739e-17	3.4037e-120	**0**	**0**	**0**	**0**
	MAX	5.3457e-16	1.8643e-119	1.9612e-168	5.2386e-201	2.5720e-271	**0**
	MIN	3.7792e-16	6.2858e-140	3.2071e-179	5.4300e-284	6.2961e-288	**0**
F3	AVE	4.9001e-16	5.4942e-122	8.9987e-167	1.7925e-195	2.6509e-270	**0**
	STD	5.3072e-17	3.0065e-121	**0**	**0**	**0**	**0**
	MAX	6.0399e-16	1.6468e-120	2.4517e-165	5.3774e-194	7.2587e-269	**0**
	MIN	3.9072e-16	3.3441e-139	1.5572e-175	3.3906e-273	1.1139e-291	**0**
F4	AVE	5.1221e-16	5.9203e-116	9.8648e-160	6.1051e-196	9.2267e-267	**0**
	STD	4.2817e-17	3.2425e-115	4.1096e-159	**0**	**0**	**0**
	MAX	6.2509e-16	1.7760e-114	2.1231e-158	1.7774e-194	2.2114e-265	**0**
	MIN	4.3181e-16	6.0132e-129	6.0412e-168	5.5357e-274	4.7684e-288	**0**
F5	AVE	1.2651e-02	3.5567e-25	4.1138e-96	6.3785e-133	1.2325e-140	**0**
	STD	4.7360e-03	1.1399e-24	5.6488e-96	3.1551e-132	6.6602e-140	**0**
	MAX	1.9514e-02	5.0722e-24	1.8074e-95	1.7242e-131	3.6496e-139	**0**
	MIN	2.0397e-03	7.9370e-01	3.9894e-100	4.9501e-194	1.7922e-148	**0**
F6	AVE	2.9927e-04	1.5666e-45	2.5032e-107	1.3487e-06	**0.0001e-309**	1.7925e-197
	STD	5.1149e-05	5.7580e-45	1.3709e-106	2.6657e-06	**0**	**0**
	MAX	5.1316e-04	2.3407e-44	7.5086e-106	1.2390e-05	**6.4229e-323**	4.3437e-196
	MIN	2.2260e-04	7.1556e-31	3.4746e-125	1.0382e-09	**0**	1.0408e-222

3.1.2. Convergence speed comparison

The simulation graphs of algorithm iteration curves for 12 experiments are displayed in Figures 2 and 3. The convergence speeds of QPSO + BOA, MQPSO, and BOA were compared in 12 experiments. QPSO + BOA exhibited the fastest convergence speed, attributed to the cooperative effect of QPSO and BOA. It was observed that L-QBOA outperformed QPSO + BOA due to the enhanced convergence speed facilitated by the hybrid mechanism. QPSO + BOA has a certain degree of performance improvement compared to MQPSO and BOA, but due to the insufficient fine-grained optimization ability of algorithms, the algorithms cannot converge. When facing multimodal functions, BOA is prone to falling into local optima, making it difficult to converge to the optimal solution. In the initial eight experiments, QSSA exhibited slower convergence compared to HBA and L-QBOA. HBA demonstrated the fastest convergence in experiments F6 and F8, whereas L-QBOA outperformed in the remaining six

experiments. The last four experiments showed comparable performance between the two algorithms. L-QBOA converges the fastest in most experiments. Consequently, it can be concluded that L-QBOA possesses a significant advantage in terms of convergence speed. (Notes: The algorithms appears to do not converge in most of the test functions, but in reality, the data output from MATLAB shows convergence (see Tables 4 and 5), which is a normal phenomenon [30, 31, 45, 46].)



(a) F1 **(b)** F2 **(c)** F3

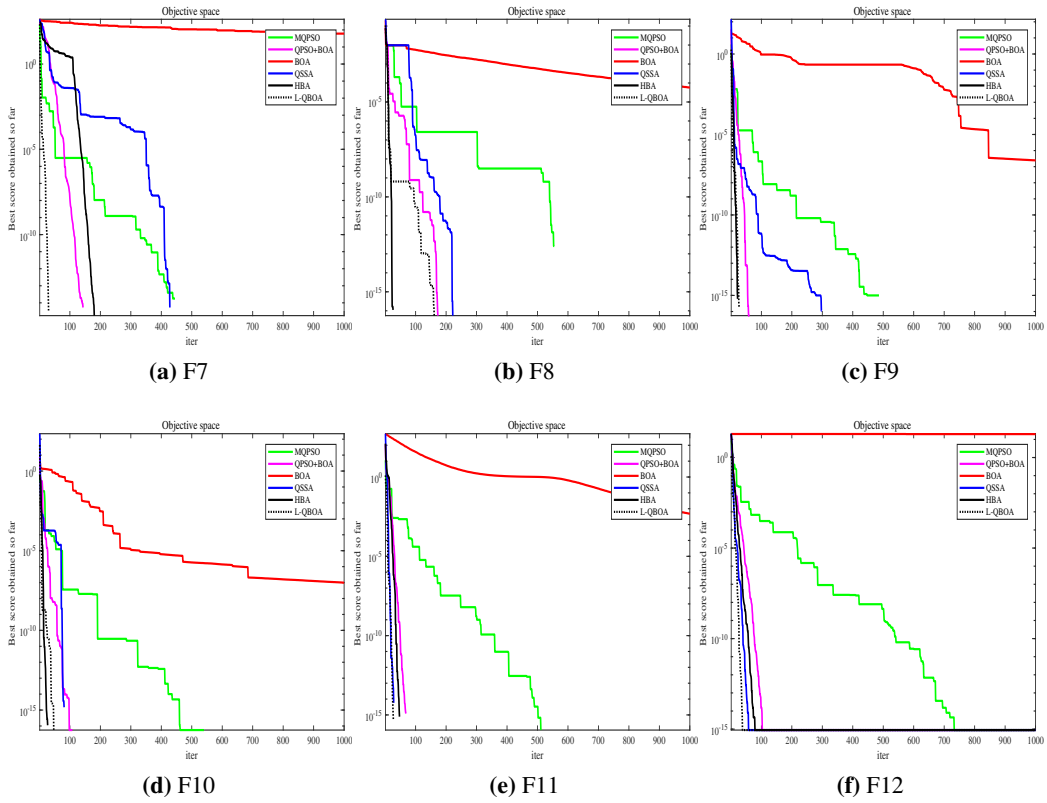
(d) F4 **(e)** F5 **(f)** F6

Figure 2. The convergence curves of all algorithms in unimodal benchmark functions.

3.1.3. Robustness comparison

In order to compare the robustness of L-QBOA with BOA, MQPSO, QPSO + BOA and QSSA, we chose a search space of 30 dimensions with a maximum generation of 1000 for all test functions. The accuracy of all function experiments is 0, except for the Ackley function. Due to the particularity of the Ackley function, we set its corresponding experimental accuracy to 8.8818e-16. Each instance is calculated 30 times, and the relevant success rates and average iteration times of the five algorithms are shown in Table 6. The optimal values for each experiment are bold in this table. From the table, it can be seen that for unimodal function testing, in F1–F5, only L-QBOA can converge to the optimum with a success rate of 100%, while other algorithms have a success rate of 0, and only HBA can converge to the best in F6, but with a success rate of only 33.33%. In the case of multi-modal functions, QPSO + BOA has a higher success rate compared to BOA and MQPSO. Even when the success rate is the same, QPSO + BOA has a smaller average number of iterations compared to MQPSO and BOA. This is attributed to the synergistic effect of the two algorithms. The success rate of L-QBOA is the same as QPSO + BOA, but L-BOA has fewer iterations. It seems that the hybrid mechanism has played a role. Furthermore, for the multimodal functions, L-QBOA outperforms QSSA in terms of success rate and average number of iterations. For the success rate, L-QBOA has a 100% success rate of six, while

HBA has only five. Although HBA requires an average of 1 less number of times in F9 experiments than L-QBOA, L-QBOA outperforms HBA in the remaining 5 experiments. Therefore, in terms of robustness, L-QBOA is superior.



(a) F7 **(b)** F8 **(c)** F9

(d) F10 **(e)** F11 **(f)** F12

Figure 3. The convergence curves of all algorithms in multimodal benchmark functions.

3.1.4. High dimensional testing

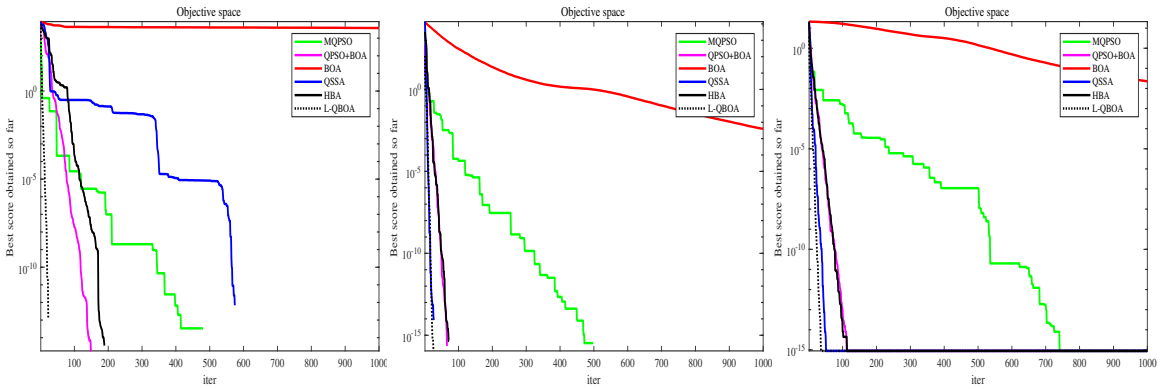
In order to test the performance of L-QBOA in solving high-dimensional optimization problems, the dimensions of the F1, F2, F6, F7, F11, and F12 were set to 100, 500, and 1000, respectively for experiments, while other parameters settings remain unchanged, listed in Tables 7–9.

Table 4 illustrates that the performance of the F6 using the L-QBOA algorithm is inferior to that of HBA. However, when the dimension of function is increased to 100, 500, and 1000, L-QBOA outperforms HBA. In experiments involving the F1 and F2, only the L-QBOA consistently computes the theoretical optimal value. As the dimension increases from 100 to 1000, all algorithms show a certain level of performance decline, except for L-QBOA which exhibits minimal decline.

When analyzing the data of the F7, F11, and F12 functions, combining Figures 4 and 5 can facilitate the comparison of algorithms. Figures 4 and 5 display the simulation convergence curves of three functions with dimensions of 500 and 1000, repectively. It is evident that L-QBOA algorithm demonstrates a clear advantage in computing F7. For F11, the performance of L-QBOA is on par with QSSA. However, in the case of F12, L-QBOA still has a slight advantage. Overall, L-QBOA has demonstrated good performance in computing high-dimensional optimization problems.

Table 5. Comparison of L-QBOA with different algorithms (F7–F12).

Function		BOA	MQPSO	QPSO + BOA	QSSA	HBA	L-QBOA
F7	AVE	4.6304e-14	1.1842e-16	**0**	**0**	**0**	**0**
	STD	5.9728e-15	6.4863e-16	**0**	**0**	**0**	**0**
	MAX	6.2172e-14	3.5527e-15	**0**	**0**	**0**	**0**
	MIN	3.5527e-14	**0**	**0**	**0**	**0**	**0**
F8	AVE	1.6193e-03	4.4409e-17	**0**	1.2955e-03	**0**	**0**
	STD	3.6828e-03	2.3297e-16	**0**	3.3592e-03	**0**	**0**
	MAX	9.7160e-03	1.2768e-15	**0**	9.7159e-03	**0**	**0**
	MIN	1.9984e-15	**0**	**0**	**0**	**0**	**0**
F9	AVE	6.2172e-16	**0**	**0**	6.5494e-02	**0**	**0**
	STD	3.4294e-16	**0**	**0**	1.0175e-01	**0**	**0**
	MAX	9.9920e-16	**0**	**0**	2.1831e-01	**0**	**0**
	MIN	**0**	**0**	**0**	**0**	**0**	**0**
F10	AVE	7.7716e-17	**0**	**0**	2.0539e-16	**0**	**0**
	STD	1.9428e-16	**0**	**0**	4.5756e-16	**0**	**0**
	MAX	8.3267e-16	**0**	**0**	2.4980e-15	**0**	**0**
	MIN	**0**	**0**	**0**	**0**	**0**	**0**
F11	AVE	2.5498e-15	**0**	**0**	**0**	**0**	**0**
	STD	2.5163e-16	**0**	**0**	**0**	**0**	**0**
	MAX	2.9976e-15	**0**	**0**	**0**	**0**	**0**
	MIN	2.1094e-15	**0**	**0**	**0**	**0**	**0**
F12	AVE	1.8551e-13	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**
	STD	7.1957e-15	**0**	**0**	**0**	**0**	**0**
	MAX	2.0339e-13	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**
	MIN	1.7497e-13	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**



(a) F7 (D = 500) **(b)** F11 (D = 500) **(c)** F12 (D = 500)

Figure 4. The convergence curves of algorithms in F7, F11, and F12 with 500 dimensions.

Table 6. Success ratio and average iterations comparison.

Function	Criteria	BOA	MQPSO	QPSO + BOA	QSSA	HBA	L-QBOA
F1	Success ratio	0	0	0	0	0	**100%**
Dsicus	Average iterations	-	-	-	-	-	**461.5333**
F2	Success ratio	0	0	0	0	0	**100%**
Sphere	Average iterations	-	-	-	-	-	**459.2000**
F3	Success ratio	0	0	0	0	0	**100%**
Sum Square	Average iterations	-	-	-	-	-	**456.2333**
F4	Success ratio	0	0	0	0	0	**100%**
Bent Cigar	Average iterations	-	-	-	-	-	**462.2333**
F5	Success ratio	0	0	0	0	0	**100.00%**
Schwefel 1.2	Average iterations	-	-	-	-	-	**764.1333**
F6	Success ratio	0	0	0	0	**33.33%**	0
Schwefel 2.22	Average iterations	-	-	-	-	**974.100**	-
F7	Success ratio	0	96.67%	**100%**	**100%**	83.33%	**100%**
Rastrigin	Average iterations	-	183.4000	169.1379	116.0667	516.300	**33.0690**
F8	Success ratio	0	93.33%	**100%**	86.67%	**100%**	**100%**
Schaffer	Average iterations	-	480.6000	359.8333	561.6667	176.1667	**65.4000**
F9	Success ratio	10%	**100%**	**100%**	70%	**100%**	**100%**
Boachevsky2	Average iterations	980.6000	166.4667	66.6	368.2414	**23.7333**	24.7667
F10	Success ratio	70%	**100%**	**100%**	23.33%	**100%**	**100%**
Boachevsky3	Average iterations	469.8000	512.7000	177.4667	929.6667	42.3667	**24.5333**
F11	Success ratio	0	**100%**	**100%**	**100%**	**100%**	**100%**
Griewank	Average iterations	-	181.7667	67.3000	364.5667	53.4000	**26.4000**
F12	Success ratio	0	**100%**	**100%**	**100%**	**100%**	**100%**
Ackley	Average iterations	-	764.5000	102.5667	53.0667	85.6667	**35.5333**

3.1.5. Average CPU time comparison

For an intelligent algorithm, its intelligence is partly reflected in high optimization performance and fast running time. Part of our work is checking the running time of each algorithm. The average CPU time of all methods is shown in Table 10. It could be seen that QPSO + BOA runs in less time than MQPSO, indicating that dividing the population into two subgroups during the initialization phase can reduce algorithm running costs. While the hybrid mechanism may slightly increase runtime, in most experiments, the algorithm's execution time is still lower than that of the QSSA algorithm. Therefore, the design of L-QBOA focuses on enhancing algorithm performance while keeping operating costs manageable.

3.2. Experimental results of the NL5 Muskingum model

To further verify the effectiveness and advantages of our presented algorithm, L-QBOA is applied to the parameter estimation problem of the NL5 Muskingum model

$$\frac{dS(t)}{dt} = I(t) - O(t), \tag{3.1}$$

$$S(t) = K[xI(t)^{\alpha_1} + (1-x)O(t)^{\alpha_2}]^m, \tag{3.2}$$

where $S(t)$, $I(t)$, and $O(t)$ represent channel storage, inflow, and outflow at time t, respectively; K is the storage-time constant; x is a weight factor; α_1, α_2 are flow parameters; and m is an exponent parameter.

Table 7. Results obtained by L-QBOA and compared algorithms with 100 dimensions.

Function		BOA	MQPSO	QPSO + BOA	QSSA	HBA	L-QBOA
F1	AVE	3.2611e-03	4.6528e-40	1.4488e-161	1.0948e-65	4.2255e-139	**0**
	STD	2.6664e-04	2.5476e-39	7.5410e-161	5.9967e-65	2.2118e-138	**0**
	MAX	3.7336e-03	1.3954e-38	4.1355e-160	3.2845e-64	1.2130e-137	**0**
	MIN	2.7145e-03	9.6926e-60	7.9964e-170	**0**	4.8298e-155	**0**
F2	AVE	3.7530e-04	1.0997e-51	2.1714e-170	2.8116e-12	3.0805e-141	**0**
	STD	3.5151e-05	5.0870e-51	**0**	1.5063e-11	1.6872e-140	**0**
	MAX	4.5002e-04	2.7590e-50	3.5079e-169	8.2556e-11	9.2411e-140	**0**
	MIN	2.9933e-04	1.4322e-62	2.2291e-177	1.7506e-291	3.5827e-159	**0**
F6	AVE	1.4651e-03	2.0973e-23	7.5115e-105	1.5832e+02	5.6416e-91	**1.4673e-117**
	STD	1.1824e-04	7.7875e-23	3.9592e-104	3.1411e+01	3.0788e-90	**7.7904e-117**
	MAX	1.6786e-03	3.2263e-22	2.1701e-103	2.3700e+02	1.6865e-89	**4.2703e-116**
	MIN	1.2205e-03	3.1265e-36	3.7964e-119	1.1277e+02	5.8680e-115	**6.5306e-133**
F7	AVE	1.0447e+02	**0**	**0**	6.7733e+00	2.7303e-01	**0**
	STD	1.6418e+01	**0**	**0**	2.2465e+01	1.0492e+00	**0**
	MAX	1.3480e+02	**0**	**0**	9.9712e+01	5.0197e+00	**0**
	MIN	6.9403e+01	**0**	**0**	**0**	**0**	**0**
F11	AVE	5.0613e-03	**0**	**0**	**0**	**0**	**0**
	STD	2.8156e-04	**0**	**0**	**0**	**0**	**0**
	MAX	5.4629e-03	**0**	**0**	**0**	**0**	**0**
	MIN	4.2910e-03	**0**	**0**	**0**	**0**	**0**
F12	AVE	2.0349e+01	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**
	STD	1.9773e-01	**0**	**0**	**0**	**0**	**0**
	MAX	2.0591e+01	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**
	MIN	1.9811e+01	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**

In order to estimate the parameters K, x, α_1, α_2, and m in Eq (3.2), we use the following steps to construct an optimization problem:

Step 1: Assume $I(t_i)$ and $O(t_i)$ respectively represent the observed values of inflow and outflow, where $t_i = i/NT$ is the time nodal in a given time interval $[0, T]$ for $i = 0, 1, \ldots, N$. Otherwise, let $\hat{O}(t_i)$ be the calculation values of outflow, and the initial value is considered to be the same as the initial inflow: $\hat{O}(t_0) = O(t_0)$.

Step 2: For the time point t_{i-1}, we calculate $S(t)$ by using Eq (3.2).

Step 3: Use the backward difference formula to approximate the first-order derivative of Eq (3.1), and combining with Eq (3.2), we obtain

$$\frac{\Delta S(t_i)}{\Delta t_i} \approx \frac{dS(t_i)}{dt_i} = I(t_i) - \left[\frac{1}{1-x}\left(\frac{S(t_i)}{K}\right)^{\frac{1}{m}} - \frac{x}{1-x} I(t_i)^{\alpha_1}\right]^{\frac{1}{\alpha_2}}, \tag{3.3}$$

where $\Delta S(t_i) = S(t_i) - S(t_{i-1})$ and $\Delta t_i = t_i - t_{i-1}$.

Table 8. Results obtained by L-QBOA and compared algorithms with 500 dimensions.

Function		BOA	MQPSO	QPSO + BOA	QSSA	HBA	L-QBOA
F1	AVE	4.2505e-03	1.0727e-40	5.7595e-163	1.8572e-63	4.9601e-95	**0**
	STD	2.9673e-04	5.8151e-40	3.1435e-162	1.0172e-62	1.9811e-94	**0**
	MAX	5.1401e-03	3.1860e-39	1.7035e-161	5.5715e-62	1.0309e-93	**0**
	MIN	3.8314e-03	1.1430e-54	3.2085e-172	**0**	1.2684e-113	**0**
F2	AVE	1.0526e-03	4.3029e-49	2.3975e-165	4.5642e-13	3.0319e-98	**0**
	STD	4.9680e-05	1.8842e-48	**0**	2.3529e-12	1.3122e-97	**0**
	MAX	1.1767e-03	1.0235e-47	6.8515e-164	1.2890e-11	7.1607e-97	**0**
	MIN	9.6180e-04	3.6124e-61	7.5190e-177	**0**	2.5930e-114	**0**
F6	AVE	2.8131e-03	1.7207e-08	2.7251e-103	4.8304e+03	7.7202e-62	**3.2805e-106**
	STD	2.5592e-04	9.1005e-08	1.3355e-102	1.2195e+03	3.9047e-61	**1.0114e-105**
	MAX	3.3493e-03	4.9878e-07	7.3078e-102	7.6269e+03	2.1376e-60	**3.9039e-105**
	MIN	2.3951e-03	1.7383e-21	4.3358e-118	2.0455e+03	2.6941e-83	**5.1107e-122**
F7	AVE	3.9227e+03	**0**	5.9212e-17	2.8718e-12	7.2691e-01	**0**
	STD	7.6776e+01	**0**	3.2432e-16	1.5160e-11	3.0849e+00	**0**
	MAX	4.0821e+03	**0**	1.7764e-15	8.3094e-11	1.6103e+01	**0**
	MIN	3.7804e+03	**0**	**0**	**0**	**0**	**0**
F11	AVE	4.1975e-03	**0**	**0**	**0**	**0**	**0**
	STD	1.1361e-04	**0**	**0**	**0**	**0**	**0**
	MAX	4.4227e-03	**0**	**0**	**0**	**0**	**0**
	MIN	3.9695e-03	**0**	**0**	**0**	**0**	**0**
F12	AVE	2.4617e-02	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**
	STD	1.0965e-03	**0**	**0**	**0**	**0**	**0**
	MAX	2.6820e-02	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**
	MIN	2.2735e-02	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**

Step 4: Calculate the outflow in the next time

$$\hat{O}(t_{i+1}) = \left[\frac{1}{1-x} \left(\frac{S(t_{i+1})}{K} \right)^{\frac{1}{m}} - \frac{x}{1-x} I(t_{i+1})^{\alpha_1} \right]^{\frac{1}{\alpha_2}}. \tag{3.4}$$

Step 5: Use the presented L-QBOA algorithm to solve the above optimization problem with SSQ as the objective function. The expression of SSQ will be given below (see Eq (3.5)).

In this paper, the performance of L-QBOA and the other parameter estimation procedures are evaluated by using the following measures:

1) SSQ: the sum of the square of the deviation.

$$SSQ = \sum_{i=1}^{N} \left[O(t_i) - \hat{O}(t_i) \right]^2. \tag{3.5}$$

Table 9. Results obtained by L-QBOA and compared algorithms with 1000 dimensions.

Function		BOA	MQPSO	QPSO + BOA	QSSA	HBA	L-QBOA
F1	AVE	4.7153e-03	1.9763e-41	8.5286e-163	2.8094e-26	4.6996e-84	**0**
	STD	4.3260e-04	1.0810e-40	3.1435e-162	1.5388e-25	1.4884e-83	**0**
	MAX	5.8403e-03	5.9211e-40	1.5560e-161	8.4282e-25	5.5976e-83	**0**
	MIN	3.8830e-03	3.1507e-58	1.6047e-170	**0**	5.5469e-107	**0**
F2	AVE	1.2587e-03	4.3029e-49	2.3975e-165	4.5642e-13	3.0319e-98	**0**
	STD	4.9680e-05	1.8842e-48	**0**	2.3529e-12	1.3122e-97	**0**
	MAX	1.1767e-03	1.0235e-47	6.8515e-164	1.2890e-11	7.1607e-97	**0**
	MIN	9.6180e-04	3.6124e-61	7.5190e-177	**0**	2.5930e-114	**0**
F6	AVE	3.5643e-03	1.5957e-06	7.5607e-104	1.8856e+04	3.5371e-58	**6.8570e-106**
	STD	3.1514e-04	6.4568e-06	3.2559e-103	3.9362e+03	1.9349e-57	**3.7386e-105**
	MAX	4.2011e-03	3.5250e-05	1.7485e-102	3.0045e+04	1.0598e-56	**2.0480e-104**
	MIN	2.7165e-03	6.6767e-15	4.6963e-119	1.0988e+04	1.3191e-78	**8.0186e-121**
F7	AVE	8.5651e+03	**0**	1.1842e-16	7.4429e-14	1.3041e-01	**0**
	STD	1.1275e+02	**0**	6.4863e-16	4.0599e-13	7.1426e-01	**0**
	MAX	8.7820e+03	**0**	3.5527e-15	2.2240e-12	3.9122e+00	**0**
	MIN	8.3482e+03	**0**	**0**	**0**	**0**	**0**
F11	AVE	3.5822e-03	**0**	**0**	**0**	**0**	**0**
	STD	1.1716e-04	**0**	**0**	**0**	**0**	**0**
	MAX	3.8137e-03	**0**	**0**	**0**	**0**	**0**
	MIN	3.3334e-03	**0**	**0**	**0**	**0**	**0**
F12	AVE	2.0576e-02	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**
	STD	4.8651e-04	**0**	**0**	**0**	**0**	**0**
	MAX	2.1675e-02	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**
	MIN	1.9342e-02	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**	**8.8818e-16**

2) SAD: sum of absolute values.

$$SAD = \sum_{i=1}^{N} \left|\hat{O}(t_i) - O(t_i)\right|. \tag{3.6}$$

3) MARE: the mean absolute relative error.

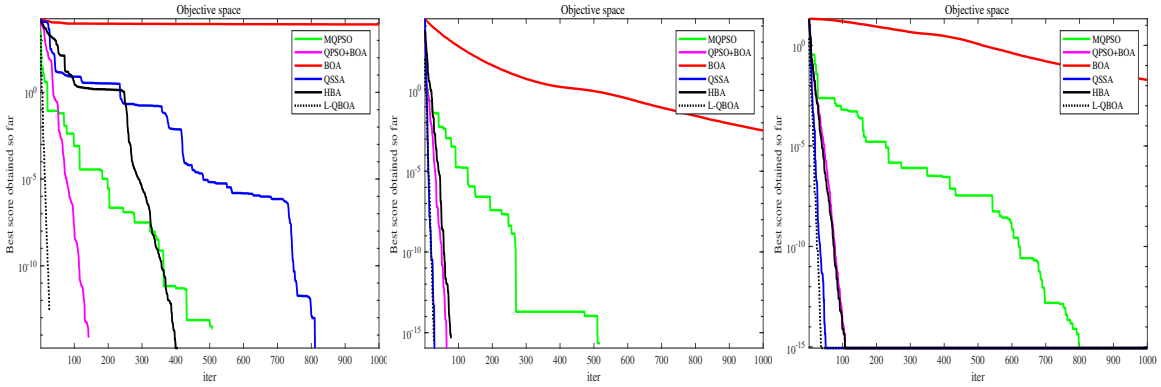
$$MARE = \frac{1}{N} \sum_{i=1}^{N} \frac{\left|O(t_i) - \hat{O}(t_i)\right|}{O(t_i)}. \tag{3.7}$$

4) PBIAS: the percent bias.

$$PBIAS = \frac{\sum\limits_{i=1}^{N} \left(\hat{O}(t_i) - O(t_i)\right)}{\sum\limits_{i=1}^{N} \hat{O}(t_i)} \times 100. \tag{3.8}$$

Table 10. Average CPU time comparison of all methods.

Function	BOA	MQPSO	QPSO + BOA	QSSA	HBA	L-QBOA
F1	0.7792	9.5839	2.1831	14.9995	0.1608	6.2647
F2	0.9597	10.4209	2.5662	12.7189	0.2075	6.7518
F3	1.0975	10.3747	2.5274	13.7242	0.2632	6.9182
F4	0.7962	9.6228	2.1986	13.2340	0.2017	6.2599
F5	3.0168	4.2704	1.1875	7.9094	0.2696	3.0168
F6	0.3857	4.3345	1.2164	1.2681	0.2943	4.3345
F7	0.9498	10.0267	2.4020	10.5691	0.1642	6.6174
F8	0.8917	10.2886	2.0004	14.7565	0.1813	6.6610
F9	0.8987	10.0042	2.0174	15.5245	0.1190	6.5867
F10	0.7321	9.7914	1.7439	12.5426	0.1189	6.2293
F11	1.5599	11.1749	3.2801	15.8137	0.3214	8.1274
F12	1.0994	10.0511	2.7588	18.2873	0.2555	6.9165



(a) F7 (D = 1000) **(b)** F11 (D = 1000) **(c)** F12 (D = 1000)

Figure 5. The convergence curves of algorithms in F7, F11, and F12 with 1000 dimensions.

These four indicators can all be used to represent the fit degree of calculated data and observed data, and SSQ is generally used as the main reference indicator, which is also used as the objective function in this paper. The smaller the value, the higher the degree of fit between the evaluation value and the observed value.

This paper applied L-QBOA to three cases to verify the performance of our proposed method. The data calculated by SFLA-NMS and GA-GRG comes from reference [37]. L-QBOA also will be compared with C-QPSO, a new method once be applied to Muskingum model by Mai et al. [28] in 2023.

3.2.1. Case 1

This is an example of parameter estimation for a nonlinear Muskingum model that Wilson [47] first considered. Table 11 shows the comparison of the SSQ, SAD, MARE and PBIAS for Wilson's data and optimal parameters obtained from L-QBOA and other methods (SFLA-NMS, GA-GRG and C-QPSO). The bold values in the table represent the optimal values of a metric calculated by algorithms. As demonstrated in Table 11, the SSQ = 5.442, SAD = 6.847, MARE = 0.253, and PBIAS = 0.29%

calculated by L-QBOA in this case are all bolded in the table, which is the best value. In this case, L-QBOA estimates the parameters as $K = 0.837$, $x = 0.076$, $\alpha_1 = 0.695$, $\alpha_2 = 0.425$, and $m = 3.820$. As is noted, the computed optimal outflows by L-QBOA are better than those calculated by the other algorithms. The comparison between calculated values from L-QBOA and observed values is shown in Figure 6. It can be seen that L-QBOA performs significantly better than the other three algorithms.

Table 11. Comparison of different models in Case 1.

Algorithm	Parameters value					Statistic values			
	K	x	α_1	α_2	m	SSQ	SAD	MARE	PBIAS
SFLA-NMS [37]	0.809	0.078	0.716	0.449	3.642	5.773	7.897	0.280	0.30%
GA-GRG [48]	0.837	0.076	0.695	0.425	3.817	5.543	7.132	0.262	0.39%
C-QPSO [28]	0.816	0.087	0.647	0.406	4.014	5.553	6.927	0.260	0.30%
L-QBOA	0.837	0.076	0.695	0.425	3.820	**5.442**	**6.847**	**0.253**	**0.29%**

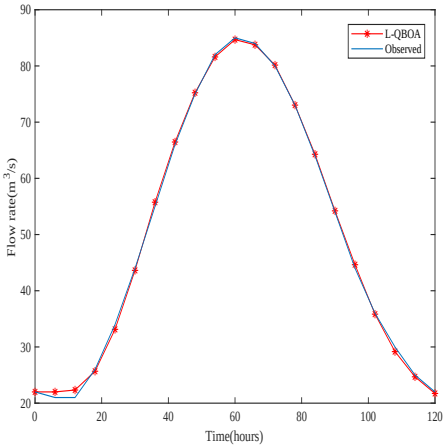


Figure 6. Case 1.

3.2.2. Case 2

The second example is a real example. This case study is a flood event that occurred in the Wye River, England. This flood event is a good example to validate the different processes of parameter estimation in the calibration step of the nonlinear Muskingum model. Table 12 displays the comparison of the best statistic values and corresponding parameters obtained from various algorithms. In this case, the PBIAS = 4.80% calculated by L-QBOA is the same as that calculated by C-QPSO, and the optimal value of PBIAS is 3.15% as calculated by FLA-NMS. However, the optimal values for SSQ, SAD and MARE are still provided by L-QBOA with 27804.91, 602.723, and 2.584, respectively. The corresponding parameters are $K = 0.466$, $x = 0.687$, $\alpha_1 = 1.342$, $\alpha_2 = 1.049$, and $m = 1.493$. The comparison between the outflow curve area and the observed values is shown in Figure 7.

Table 12. Comparison of different models in Case 2.

Algorithm	Parameters value					Statistic values			
	K	x	α_1	α_2	m	SSQ	SAD	MARE	PBIAS
GA-GRG [48]	0.424	0.210	1.272	1.138	1.354	29479.16	684.830	2.943	3.58%
SFLA-NMS [37]	0.600	0.609	1.056	1.163	1.398	28207.55	679.327	3.017	**3.15%**
C-QPSO [28]	0.979	0.704	0.992	1.152	1.395	28054.42	602.724	2.589	4.80%
L-QBOA	0.466	0.687	1.342	1.049	1.493	**27804.91**	**602.723**	**2.584**	4.80%

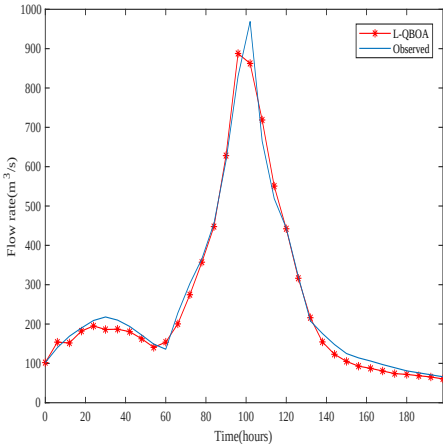


Figure 7. Case 2.

3.2.3. Case 3

The last example is the case of a flood with a multi-peak discharge hydrograph introduced by Viessman and Lewis in 2003 [49]. This particular case has a time interval Δ of 12 hours. The outflow hydrography exhibits two peaks occurring on the 10^{th} and 17^{th} days. The inflow ranges from 143 *cms* to 1775.5 *cms*. The minimum and maximum outflow values are 118.4 and 1509.3 *cms*, respectively. Table 13 presents the SSD, SAD, MARE, and PBIAS values for the NL5 Muskingum model by using L-QBOA and other algorithms, respectively. The values in bold are the optimal values of the metric. In this case, C-QPSO provides an optimal value of PBIAS of 0.07%, and the calculated SSQ and SAD are better than that of SSFLA-NMS and GA-GRG. However, the optimal values for SSQ, SAD, and MARE are again calculated by L-QBOA as 65699.17, 926.20, and 1.423, respectively. The corresponding coefficients are $K = 0.7954$, $x = 8.21 \times 10^{-8}$, $\alpha_1 = 3.364$, $\alpha_2 = 1.412$, and $m = 1.021$. The graph of outflow with the observed values is shown in Figure 8.

Table 13. Comparison of different models in Case 3.

Algorithm	Parameters value					Statistic values			
	K	x	α_1	α_2	m	SSQ	SAD	MARE	PBIAS
GA-GRG [48]	0.089	3×10^{-4}	1.846	1.002	1.393	70738	1018	1.627	0.23%
SFLA-NMS [37]	0.078	5×10^{-7}	3.121	1.420	1.000	69860	993	1.494	0.21%
C-QPSO [28]	0.752	5.57×10^{-4}	1.896	1.131	1.278	66815.72	945.77	1.576	**0.07%**
L-QBOA	0.796	8.21×10^{-8}	3.364	1.412	1.021	**65699.17**	**926.20**	**1.423**	0.22%

Tables 11–13 use SSQ as the main indicator, which is an objective function. In fact, other indicators derived from data may behave differently. When considering the impact of all four measures simultaneously, SFLA-NMS and C-QPSO face challenges. SFLA-NMS achieves a smaller PBIAS in Case 2, while C-QPSO achieves a smaller PBIAS in Case 3. However, both methods show a significant difference in SSQ from the optimal solution compared to L-QBOA. In conclusion, the parameter estimation performance of L-QBOA, as proposed in this paper for the NL5 Muskingum model, is as good or better than other methods.

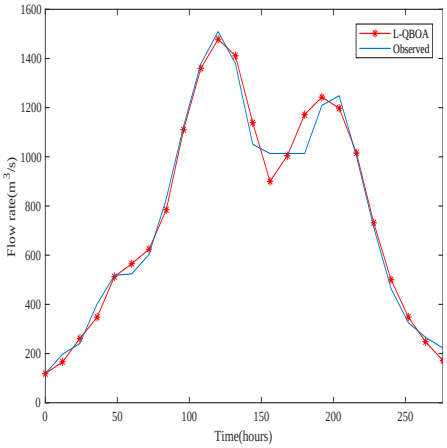


Figure 8. Case 3.

4. Conclusions

This article proposes L-QBOA, a hybrid algorithm that combines QPSO and BOA while introducing the Lévy flight mechanism. The algorithm divides the population into two subgroups during initialization and employs a hybrid mechanism during optimization to enhance collaboration. Experimental results on benchmark functions demonstrate that L-QBOA exhibits strong global optimization capabilities and performs well in high-dimensional problems. At the same time, it has been proven that using *gbest* points to establish algorithm cooperative effects and a hybrid mechanism can effectively improve algorithm performance, which can serve as a reference for designing hybrid algorithms. To validate its practical applicability, L-QBOA was tested on the NL5 Muskingum model, yielding the best SSQ in all three cases. In conclusion, the proposed L-QBOA method shows advantages in the NL5 Muskingum model. In the future, L-QBOA can be considered for application to updated Muskingum models or other parameter estimation problems, numerical solutions of differential equations, and their inverse problems, path planning, or similar issues.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (12361087) and the Natural Science Foundation of Guangxi Province (2022GXNSFAA035618).

Conflict of interest

The authors declare there is no conflict of interest.

References

1. A. Ouyang, Y. Lu, Y. Liu, M. Wu, X. Peng, An improved adaptive genetic algorithm based on DV-Hop for locating nodes in wireless sensor networks, *Neurocomputing*, **458** (2021), 500–510. https://doi.org/10.1016/j.neucom.2020.04.156
2. J. Lin, S. X. Zhang, S. Y. Zheng, Y. M. Pan, Differential evolution with fusion of local and global search strategies, *J. Comput. Sci.*, **63** (2022), 101746. https://doi.org/10.1016/j.jocs.2022.101746
3. W. He, B. Wang, N. Li, X. Gao, W. Li, Q. Jiang, An improved Sine-Cosine algorithm with dynamic selection pressure, *J. Comput. Sci.*, **55** (2021), 101477. https://doi.org/10.1016/j.jocs.2021.101477
4. D. Simon, Biogeography-based optimization, *IEEE Trans. Evol. Comput.*, **12** (2008), 702–713. https://doi.org/10.1109/TEVC.2008.919004
5. S. Khan, M. Kamran, O. U. Rehman, L. Liu, S. Yang, A modified PSO algorithm with dynamic parameters for solving complex engineering design problem, *Int. J. Comput. Math.*, **95** (2018), 2308–2329. https://doi.org/10.1080/00207160.2017.1387252
6. S. Mitra, S. Acharyya, Perturbation and repository based diversified cuckoo search in reconstruction of gene regulatory network: A new cuckoo search approach, *J. Comput. Sci.*, **60** (2022), 101600. https://doi.org/10.1016/j.jocs.2022.101600
7. S. Arora, S. Singh, Butterfly optimization algorithm: a novel approach for global optimization, *Soft Comput.*, **23** (2019), 715–734. https://doi.org/10.1007/s00500-018-3102-4
8. D. H. Wolpert, W. G. Macready, No free lunch theorems for optimization, *IEEE Trans. Evol. Comput.*, **1** (1997), 67–82. https://doi.org/10.1109/4235.585893
9. C. Zhong, G. Li, Z. Meng, W. He, Opposition-based learning equilibrium optimizer with Levy flight and evolutionary population dynamics for high-dimensional global optimization problems, *Expert Syst. Appl.*, **215** (2023), 119303. https://doi.org/10.1016/j.eswa.2022.119303
10. G. Tian, Y. Ren, Y. Feng, M. Zhou, H. Zhang, J. Tan, Modeling and planning for dual-objective selective disassembly using AND/OR graph and discrete artificial bee colony, *IEEE Trans. Ind. Inf.*, **15** (2018), 2456–2468. https://doi.org/10.1109/TII.2018.2884845
11. S. Lou, Y. Zhang, R. Tan, C. Lv, A smooth path planning method for mobile robot using a BES-incorporated modified QPSO algorithm, *Expert Syst. Appl.*, **208** (2022), 118256. https://doi.org/10.1016/j.eswa.2022.118256

12. S. Dian, J. Zhong, B. Guo, J. Liu, R. Guo, A human-cyber-physical system enabled sequential disassembly planning approach for a human-robot collaboration cell in Industry 5.0, *Rob. Comput. Integr. Manuf.*, **208** (2022), 118256. https://doi.org/10.1016/j.rcim.2023.102706

13. K. Aygül, M. Cikan, T. Demirdelen, M. Tumay, Butterfly optimization algorithm based maximum power point tracking of photovoltaic systems under partial shading condition, *Energy Sources Part A*, **45** (2023), 8337–8355. https://doi.org/10.1080/15567036.2019.1677818

14. S. Arora, P. Anand, Learning automata-based butterfly optimization algorithm for engineering design problems, *Int. J. Comput. Mater. Sci. Eng.*, **7** (2018), 1850021. https://doi.org/10.1142/S2047684118500215

15. G. Li, F. Shuang, P. Zhao, C. Le, An improved butterfly optimization algorithm for engineering design problems using the cross-entropy method, *Symmetry*, **11** (2019), 1049. https://doi.org/10.3390/sym11081049

16. Y. Fan, J. Shao, G. Sun, X. Shao, A self-adaption butterfly optimization algorithm for numerical optimization problems, *IEEE Access*, **2021** (2021), 88026–88041. https://doi.org/10.1109/ACCESS.2020.2993148

17. Y. Li, X. Yu, J. Liu, Enhanced butterfly optimization algorithm for large-scale optimization problems, *J. Bionic. Eng.*, **19** (2022), 554–570. https://doi.org/10.1007/s42235-021-00143-3

18. A. Assiri, On the performance improvement of butterfly optimization approaches for global optimization and feature selection, *PLoS One*, **16** (2021), 0242612. https://doi.org/10.1371/journal.pone.0242612

19. M. W. Li, D. Y. Xu, J. Geng, W. C. Hong, A ship motion forecasting approach based on empirical mode decomposition method hybrid deep learning network and quantum butterfly optimization algorithm, *Nonlinear Dyn.*, **107** (2022), 2447–2467. https://doi.org/10.1007/s11071-021-07139-y

20. Z. Wang, Q. Luo, Y. Zhou, Hybrid metaheuristic algorithm using butterfly and flower pollination base on mutualism mechanism for global optimization problems, *Eng. Comput.*, **37** (2020), 3665–3698. https://doi.org/10.1007/S00366-020-01025-8

21. A. Toktas, D. Ustun, A triple-objective optimization scheme using butterfly integrated ABC algorithm for design of multilayer RAM, *IEEE Trans. Antennas Propag.*, **68** (2020), 5602–5612. https://doi.org/10.1109/TAP.2020.2981728

22. Z. A. Dahi, C. Mezioud, A. Draa, A quantum-inspired genetic algorithm for solving the antenna positioning problem, *Swarm Evol. Comput.*, **31** (2016), 24–36. https://doi.org/10.1016/j.swevo.2016.06.003

23. S. Dey, S. Bhattacharyya, U. Maulik, Quantum inspired genetic algorithm and particle swarm optimization using chaotic map model based interference for gray level image thresholding, *Swarm Evol. Comput.*, **15** (2014), 38–57. https://doi.org/10.1016/j.swevo.2013.11.002

24. B. Liu, Y. Zhou, Q. Luo, H. Huang, Quantum-inspired African vultures optimization algorithm with elite mutation strategy for production scheduling problems, *J. Comput. Des. Eng.*, **10** (2023), 1767–1789. https://doi.org/10.1093/jcde/qwad078

25. R. N. D. Costa-Filho, Comparative study of three quantum-inspired optimization algorithms for robust tuning of power system stabilizers, *Neural Comput. Appl.*, **35** (2023), 12905–12914. https://doi.org/10.1007/s00521-023-08429-9

26. J. Sun, B. Feng, W. Xu, Particle swarm optimization with particles having quantum behavior, in *Proceedings of the 2004 Congress on Evolutionary Computation*, Portland, OR, USA, (2004), 111–116.

27. N. Kumar, A. A. Shaikh, S. K. Mahato, A. K. Bhunia, Applications of new hybrid algorithm based on advanced cuckoo search and adaptive Gaussian quantum behaved particle swarm optimization in solving ordinary differential equations, *Expert Syst. Appl.*, **172** (2021), 114646. https://doi.org/10.1016/j.eswa.2021.114646

28. X. Mai, H. B. Liu, L. B. Liu, A new hybrid cuckoo quantum-behavior particle swarm optimization algorithm and its application in Muskingum model, *Neural Process. Lett.*, **55** (2023), 8309–8337. https://doi.org/10.1007/s11063-023-11313-1

29. C. Wang, Z. Wang, S. Zhang, J. Tan, Adam-assisted quantum particle swarm optimization guided by length of potential well for numerical function optimization, *Swarm Evol. Comput.*, **79** (2023), 101309. https://doi.org/10.1016/j.swevo.2023.101309

30. Y. Ling, Y. Zhou, Q. Luo, Lévy flight trajectory-based whale optimization algorithm for global optimization, *IEEE Access*, **5** (2017), 6168–6186. https://doi.org/10.1109/ACCESS.2017.2695498

31. C. Zhong, G. Li, Z. Meng, W. He, Opposition-based learning equilibrium optimizer with Levy flight and evolutionary population dynamics for high-dimensional global optimization problems, *Expert Syst. Appl.*, **15** (2023), 119303. https://doi.org/10.1016/j.eswa.2022.1193032

32. R. Sihwail, K. Omar, K. A. Z. Ariffin, M. Tubishat, Improved Harris Hawks optimization using elite opposition-based learning and novel search mechanism for feature selection, *IEEE Access*, **8** (2020), 121127–121145. https://doi.org/10.1109/ACCESS.2020.3006473

33. M. A. Elaziz, D. Yousri, S. Mirjalili, A hybrid Harris hawks-moth-flame optimization algorithm including fractional-order chaos maps and evolutionary population dynamics, *Adv. Eng. Software*, **154** (2021), 102973. https://doi.org/10.1016/j.advengsoft.2021.102973

34. W. C. Wang, W. C. Tian, D. M. Xu, K. W. Chau, Q. Ma, C. J. Liu, Muskingum models' development and their parameter estimation: A state-of-the-art review, *Water Resour. Manage.*, **37** (2023), 3129–3150. https://doi.org/10.1007/s11269-023-03493-1

35. A. Ouyang, L. B. Liu, Z. Sheng, F. Wu, A class of parameter estimation methods for nonlinear Muskingum model using hybrid invasive weed optimization algorithm, *Math. Probl. Eng.*, **2015** (2015), 573894. https://doi.org/10.1155/2015/573894

36. U. Okkan, U. Kirdemir, Locally tuned hybridized particle swarm optimization for the calibration of the nonlinear Muskingum flood routing model, *J. Water Clim. Change*, **11** (2020), 343–358. https://doi.org/10.2166/wcc.2020.015

37. R. Akbari, M. R. Hessami-Kermani, A new method for dividing food period in the variable-parameter Muskingum models, *Hydrol. Res.*, **53** (2015), 241–257. https://doi.org/10.2166/nh.2021.192

38. O. B. Haddad, F. Hamedi, M. Orouji, M. Pazoko, H. A. Loáiciga, A re-parameterized and improved nonlinear Muskingum model for flood routing. *Water Resour. Manage*, **29** (2015), 3419–3440. https://doi.org/10.1007/s11269-015-1008-9
39. X. Lu, G. He, QPSO algorithm based on Levy flight and its application in fuzzy portfolio, *Appl. Soft Comput.*, **99** (2020), 106894. https://doi.org/10.1016/j.asoc.2020.106894
40. J. P. Jacob, K. Pradeep, A multi-objective optimal task scheduling in cloud environment using cuckoo particle swarm optimization, *Wireless Pers. Commun.*, **109** (2019), 315–331. https://doi.org/10.1007/s11277-019-06566-w
41. B. Wang, J. Wei, Particle swarm optimization with genetic evolution for task offloading in device-edge-cloud collaborative computing, in *Advanced Intelligent Computing Technology and Applications. ICIC 2023* (eds. D. S. Huang, P. Premaratne, B. Jin, B. Qu, K. H. Jo, A. Hussain), Springer Nature Singapore, (2023), 340–350. https://doi.org/10.1007/978-981-99-4761-4_29
42. S. Wang, Y. Li, H. Yang, Self-adaptive mutation differential evolution algorithm based on particle swarm optimization, *Appl. Soft Comput.*, **81** (2019), 105496. https://doi.org/10.1016/j.asoc.2019.105496
43. Z. Yu, J. Du, Constrained fault-tolerant thrust allocation of ship DP system based on a novel quantum-behaved squirrel search algorithm, *Ocean Eng.*, **266** (2022), 112994. https://doi.org/10.1016/j.oceaneng.2022.112994
44. F. A. Hashim, E. H. Houssein, K. Hussain, M. S. Mabrouk, W. Al-Atabany, Honey badger algorithm: New metaheuristic algorithm for solving optimization problems, *Math. Comput. Simul.*, **192** (2022), 84–110. https://doi.org/10.1016/j.matcom.2021.08.013
45. A. Faramarzi, M. Heidarinejad, S. Mirjalili, A. H. Gandomi, Marine predators algorithm: A nature-inspired metaheuristic, *Expert Syst. Appl.*, **152** (2020), 113377. https://doi.org/10.1016/j.eswa.2020.113377
46. Y. Xia, Z. Feng, W. Niu, H. Qin, Z. Jiang, J. Zhou, Simplex quantum-behaved particle swarm optimization algorithm with application to ecological operation of cascade hydropower reservoirs, *Appl. Soft Comput.*, **84** (2019), 105715. https://doi.org/10.1016/j.asoc.2019.105715
47. E. M. Wilson, *Engineering hydrology*, MacMillan Education, London, 1974.
48. S. M. Easa, Improved nonlinear Muskingum model with variable exponent parameter, *J. Hydrol. Eng.*, **18** (2013), 1790–1794. https://doi.org/10.1061/(ASCE)HE.1943-5584.0000702
49. W. Viessman, G. Lewis, *Introduction to Hydrology*, Archaea, the United State, 2003.

AIMS Press