Boosting Quantum Rotation Gate Embedded Slime Mould Algorithm

Article in Expert Systems with Applications · April 2021
DOI: 10.1016/j.eswa.2021.115082

CITATIONS
2

READS
267

6 authors, including:

Yu Caiyang
Wenzhou University
5 PUBLICATIONS 150 CITATIONS

Lejun Zhang
Yangzhou University
52 PUBLICATIONS 338 CITATIONS

Ali Asghar Heidari
National University of Singapore
117 PUBLICATIONS 5,649 CITATIONS

Huilin Chen
Wenzhou University
224 PUBLICATIONS 9,127 CITATIONS

Some of the authors of this publication are also working on these related projects:

- Machine learning for financial distress prediction View project
- The Colony Predation Algorithm (CPA) View project

All content following this page was uploaded by Ali Asghar Heidari on 09 May 2021.
Boosting quantum rotation gate embedded slime mould algorithm

Caiyang Yu\textsuperscript{a}, Ali Asghar Heidari\textsuperscript{b,c,2}, Xiao Xue\textsuperscript{d,e}, Lejun Zhang\textsuperscript{f,1,a}, Huiling Chen\textsuperscript{a,1,a}, Weibin Chen\textsuperscript{a,1,a}

\textsuperscript{a} Department of Computer Science and Artificial Intelligence, Wenzhou University, Wenzhou 325035, China
\textsuperscript{b} School of Surveying and Geospatial Engineering, College of Engineering, University of Tehran, Tehran, Iran
\textsuperscript{c} Department of Computer Science, School of Computing, National University of Singapore, Singapore
\textsuperscript{d} College of Intelligence and Computing, Tianjin University, Tianjin 300072, China
\textsuperscript{e} College of Computer Science and Technology, Henan Polytechnic University, Jiaosuo 454003, China
\textsuperscript{f} College of Information Engineering, Yangzhou University, Yangzhou 225127, China

\section*{ABSTRACT}

The slime mould algorithm is an interesting swarm-based algorithm proposed in 2020 based on this entity's trajectory finding abilities in nature. It simulates slime mould movement, foraging, and other behaviors to find the problem's optimal solution. Because of the complexity of the slime mould's trajectory, the SMA has strong randomness and makes the generated population diverse. However, in the later iteration of the algorithm, the complexity of the problem to be dealt with increases, it tends to drop into the local best, and the convergence rate slows down. Therefore, in this study, an improved SMA, named WQMSA, is proposed to remedy the above imperfections. Specifically, two strategies of quantum rotation gate and an operation from water cycle are used for the first time to improve the robustness of the original SMA. The purpose of adding both mechanisms is to keep the algorithm in equilibrium among exploration and exploitation inclinations. While expanding the search space of individual population, it also makes a more detailed exploration of the local area. The quantum rotation gate, which rotates by its small angle, can adequately exploit the algorithm and search in the local scope enough. Simultaneously, the water cycle mechanism can help the algorithm search thoroughly in the space to find the optimal solution. The improved algorithm was compared with 14 classical meta-heuristics and 14 advanced algorithms on the test set IEEE CEC 2014, and the results were obtained, with WQMSA ranking first in both comparisons. Also, to further illustrate the role of WQMSA in practical application, three engineering problems are used for verification. Experimental results show that WQMSA also performs well in solving such practical problems. A website at https://aliaagharheidari.com will support this research.

\section*{1. Introduction}

Optimization can be tackled under a wide umbrella of approaches such as fuzzy optimization (Huang et al., 2019a), robust optimization, large scale tasks (Cao, Zhao, Gu, Ling, & Ma, 2020), memetic methods, multi and many objective problems (Cao, Wang, Zhang, Song, & Lv, 2020; Sun et al., 2020). Among the many approaches, the form of a meta-heuristic algorithm to solve the combinatorial optimization problem has received much heat from scholars. The meta-heuristic is to find the best solution at an acceptable cost. Although the resulting solution may not be optimal, there is no specific algorithm to solve most optimization problems, nor is there an algorithm that can solve most optimization problems. Thus, the meta-heuristic algorithm becomes a realistically, desirable, and effective method than the traditional methods (Zhang, 2013, 2015). It has found its application in a variety scenarios including maximum satisfiability problem (Zeng, Lu, & Mao, 2011; Zeng & Li et al., 2012), Sliding-mode control (Qu, Xu, Zhou, & Zhang, 2021), integrated control algorithms (Zhang et al., 2021), intelligent management system (Mi et al., 2020), PID optimization control (Zeng et al., 2015, 2014; Zeng, Xie, Chen, & Weng, 2019), wind speed prediction (Chen, Zeng, Lu, & Weng, 2019), scheduling problem (Pang, Zhou, Tsai, & Chou, 2018; Zhou, Pang, Chen, & Chou, 2018), prediction problems in educational field (Lin et al., 2019; Tu, Lin, Chen, Li, & Li, 2019; Wei et al., 2020, 2017; Zhu et al., 2020), engineering applications (Ba et al., 2020; Chen, Wang, & Zhao, 2020e; Tu et al., 2020; Wang, 2018), tomography analysis (Yang et al., 2020), and parameter extraction of solar cell (Abbasi et al., 2020; Chen, Jiao, et al., 2019; Chen, Jin, Wang, Heidari, & Zhao, 2020; Ridha et al., 2021; Ridha, Heidari, Wang, & Chen, 2020).

Swarm intelligence optimization algorithms are a class of heuristics that have emerged in recent decades and are developed by observing organisms' growth patterns in nature. Such algorithms draw on the
ideas and essence of many different disciplines, including mathematics, computational science, neuroscience, statistics, biology, etc (Jiang et al., 2017; Xue et al., 2021). For example, the classic genetic algorithm (GA) (Holland, 1992) is based on biology combined with computational science (Mogilner & Edelstein-Keshet, 1999). Representative classic swarm intelligence optimization algorithms are particle swarm algorithm (PSO) (Bai, Guo, Zhou, Zhang, & Zhang, 2021; Kennedy & Eberhart, 1995), simulated annealing algorithm (SA) (Kirkpatrick, Gelatt, & Vecchi, 1987), ant colony optimization algorithm (Dorigo & Blum, 2005; Tu et al., 2020; Zhao, Liu, Yu, Asghar Heidari, Wang, Oliva, et al., 2020; Zhao, Liu, Yu, Asghar Heidari, Wang, Liang, et al., 2020), and different evolution (Storn and Price, 1997) (DE), etc. In recent years there have been many special algorithms produced, some of which have been greatly improved in terms of performance, such as bat algorithm (BA) (Yang, 2010), extremal optimization algorithm (EA) (Chen, Zeng, & Liu, 2019; Chen, Zeng, & Lu, 2019; Li, Lu, Zeng, Wu, & Chen, 2016; Zeng et al., 2016), moth search algorithm (MSA) (Wang, 2018), monarch butterfly optimization (MBO) (Feng, Deb, Wang, & Alavi, 2021), moth-flame algorithm (MFO) (S. Mirjalili, 2015), salp swarm algorithm (SSA) (Mirjalili et al., 2017; H. Zhang et al., 2020), fruit fly optimization algorithm (FOA) (Fan et al., 2011; Pan, 2012; X. Wang et al., 2020; H. Yu et al., 2020), gray wolf optimizer (GWO) (Hu et al., 2021; Mirjalili, Mirjalili, & Lewis, 2014), Harris hawks optimization (HHO) (Heidari et al., 2019; S. Song et al., 2020a), hunger games search (HGS) (Yang, Chen, Heidari, & Gandomi, 2021), runge kutta optimization (RUN) (Ahmadianfar, Heidari, Gandomi, Chu, & Chen, 2021), grasshopper optimization algorithm (GOA) (Saremi, Mirjalili, & Lewis, 2017; Yu et al., 2021a), whale optimization algorithm (WOA) (Mirjalili & Lewis, 2016; Tu et al., 2020), etc. As each algorithm is generated, there is a performance gap, more or less, between the other algorithms, so improving the performance of the algorithms has become another major direction for scholars to study, such as Xu, Chen, Luo, et al., (2019), Shan et al., (2020), Zhang, Liu, Wang, Chen, and Li (2020), Xu & Chen (2014), H. Chen et al., (2019), and so on. The above algorithm is widely used in various fields due to its simple structure, convenient operation, and excellent optimization accuracy as, Dawid et al. (Polap, 2020) proposed a new method for adaptive image analysis by combining GA and cascaded of the convolutional classifiers, as well as, a heuristic validation mechanism was utilized for training different types of neural networks designed for image reconstruction (Polap & Srivastava, 2020). Hafeez et al. (Hafeez, Alamgeer, & Khan, 2020) developed a genetic wind-driven optimization algorithm for future electric load forecasting. All these applications strongly demonstrate the effectiveness of the swarm intelligence optimization algorithm in practical applications and further illustrate its research value (Ma et al., 2021).

In the last 2–3 years, a number of heuristic solutions for optimization problems have emerged, such as political optimizer (PO) (Askari, Younas, & Saeed, 2020) algorithm has good convergence in solving optimization problems, but its time complexity can be too high for a primal function, so it cannot be well applied to practical problems. For some improved algorithms involving many parameters, the performance may be significantly improved than the original algorithm.

A recent article on the slime mould algorithm (SMIA) (Li et al., 2020a) has generated much buzz among researchers. SMA is the result of a study of the activity and dynamics of slime mould. On two datasets of 23 benchmark functions and IEEE CEC 2014 functions, SMA showed strong performance compared with other algorithms, e.g., WOA, GWO, and MFO etc. Combined with the code, it can be seen that SMA has the advantages of simplicity, flexibility, and more. However, in some instances or functions, SMA also exhibits defects such as failure to converge to global optimum or slow convergence. This paper proposes a variant of SMA, called the water cycle and quantum-behaved SMA (WQSMA), to have a better SMA performance.

In WQSMA, two efficient mechanisms, including quantum rotation gate (QRG) and water cycle (WC), are integrated into the original SMA. Both mechanisms work by altering the position of the slime mould population to obtain a more optimal solution. First of all, the QRG mechanism changes the slime mould population's position through the rotation gate. Each rotation will lead the population in a better direction. The second is the WC mechanism, based on water circulation in nature, incorporated into the SMA to treat water as a slime mold population. The position of the water in nature reflects the position of the population. While both mechanisms can increase population diversity, there are differences between the two. For QRG, we set the rotation angle in the experiment to be very small, so the population position change is small, helping local search ability. Whereas in WC, population position changes are more significant, as they facilitate global search. So the combination of the two mechanisms can make a sufficient improvement to the SMA. To validate the improved WQSMA, the algorithms were tested in the IEEE CEC 2014 function set, and multiple meta-heuristics algorithms and advanced algorithms were selected for comparison. Finally, the algorithm is applied to an engineering problem to test whether the improved algorithm helps solve a real problem. The main work of this study can be divided into the following points.

a) The QRG and WC mechanisms were introduced to address the shortcomings of the SMA.

b) Conduct a series of comparative experiments on WQSMA to analyze its performance.

c) The proposed WQSMA is applied to engineering problems, and good results are obtained

The structure of this paper is presented as follows. Section 2 provides background knowledge of the article. Section 3 describes the WQSMA in detail. Section 4 shows the experimental results, including the application of WQSMA to engineering cases. The research results are summarized in Section 5.

2. Background

2.1. Quantum-like component

2.1.1. Literature review

The concept of “quantum” was proposed a century ago, and it first appeared in physics. The hottest research on quantum computers at the moment requires the full use of quantum properties. Many scholars have introduced quantum theory into different algorithms (Caiyang, Cai, Wang, Zhao, & Li, 2020; Yu, Heidari, & Chen, 2020). Coelho et al. (Coelho, 2010) proposed a novel quantum-behaved particle swarm algorithm (QPSO) by combining the classical particle swarm algorithm with quantum mechanics theory and using Gaussian probability distribution variation operator instead of the original random sequence to prevent premature convergence of the original algorithm into the local optimum solution. Zou et al. (Zou, Wang, Hei, Chen, & Yang, 2014) propose to incorporate dynamic group strategy (DGS) into a teaching–learning-based optimization (TLBO) algorithm, allowing learners in the original algorithm to choose to learn in the mean of the group rather than the mean of the class, reducing the range and improving the accuracy. Simultaneously, quantum behavioral learning strategies were introduced in the group learning process to maintain
population diversity. Based on the excellent performance of the quantum-inspired evolutionary algorithm (QEA) in solving the knapsack problem, Han et al. (Han & Kim, 2004) have further improved the algorithm and proposed research strategies on QEA such as termination criterion, Q-gate, and two-phase scheme. The results also demonstrate that the improved method is significantly better than before in terms of adaptability and robustness. Jiao et al. (Jiao, Li, Gong, & Zhang, 2008) proposed a new quantum-inspired immune clonal algorithm (QICA) to solve the global optimization problem. Two strategies, QRG and dynamic angle adjustment, were used to accelerate convergence and improve efficiency during antibody renewal. Liu et al. (Liu, Jiao, Ma, Ma, & Shang, 2016) proposed and validated a new simplex quantum behavioral particle swarm algorithm to overcome the premature convergence and stagnant search problems in the original method for the optimization problem in gradient hydroelectric plants. Mariani et al. (Mariani, Klassen Duck, Guerra, dos Santos Coelho, & Rao, 2012) combined the Zaslavski chaotic map sequences to propose a new minimization-based shell-and-tube heat exchanger quantum particle swarm optimization (QPSOZ) problem, which shows that QPSOZ has optimal performance compared to traditional PSO, GA, etc.

2.1.2. Quantum bit

Quantum bits are the theoretical keystone of a concept, known as quantum computing. In a conventional computer, the information unit can only be used in binary “0” or “1” to represent, in other words, it is either in the “0” state or in the “1” state so that it can show only two states. In a binary quantum computer, the information unit is called a qubit, which in addition to being in a “0” state or a “1” state, can also be in a superposition state.

The superposition state is an arbitrary linear superposition of a state of “0” and a state of “1”, which can be either a state of “0” or a state of “1”, or a state of “0” and a state of “1” that exist together with a certain probability. It is defined as follows:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \left(\begin{array}{c} \alpha \\ \beta \end{array}\right)$$

where $\alpha$ and $\beta$ are two coefficients that mean the probability amplitude of two states. $|\psi\rangle$ is the final quantum state obtained. $|\alpha|^2$ and $|\beta|^2$ are the probability that quantum bits are in states “0” and “1”, and the sum of the two is 1. The function can be shown as:

$$|\alpha|^2 + |\beta|^2 = 1$$

2.1.3. Quantum rotation gate

In WQSM, we introduce the QRG mechanism to increase population diversity and thereby enhance local search capabilities. Qubits are characterized as binary. The population data generated by the swarm intelligence optimization algorithm are floating-point data, so the quantum bit’s discrete data should become the algorithm’s continuous data. Each individual of the population in the algorithm consists of multiple dimensions. The values obtained from each dimension’s initialization are continuous data, so these values are directly put into the quantum rotation gate as quantum bits for operation. The adjusting operation and updating process of the QRG is as follows.

$$U(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

$$\alpha' = U(\theta) \alpha$$

$$\beta' = U(\theta) \beta$$

where $\theta_i$ denotes the $i$-th rotation angle, its size, and direction of rotation are set in advance. Besides, its adjustment method in WQSM is shown in Table 1; $(\alpha, \beta)^T$ and $(\alpha', \beta')^T$ represent the quantum bit states vectors before and after the rotation gate update of the $i$-th quantum bit of a chromosome.

where $f(x_i)$ indicates the fitness value of the $i$-th population, best_fitness indicates the best fitness value to date, $s$ indicates the angle of rotation (the angle is consistent with the original literature), and $f (\alpha, \beta)$ indicates the direction of rotation. $\theta_i$ indicates the individual value for the $j$-th position in the corresponding population, $\bar{\theta}_i$ represents the individual value of the $j$-th position in the most optimal population to date.

The primary role of QRG is to determine the superior population by comparing the fitness values of individuals in two population sets and then to shift the value of the poorer population to the superior one.

Algorithm 1 Pseudocode of QRG

Input parameters: position and fitness values of $X$, position and fitness values of optimal individuals popsize, dim;

Initialize $\alpha$, $\beta$, rotation angle $\Delta$, rotation degree $s$;

While $i < \text{popsize}$

While $j < \text{dim}$

Update $\alpha$, $\beta$, $\Delta$, and $s$;

Compare the fitness(s) and best_fitness;

Update the delta and $s$ according to Table 1;

Perform QRG by Eq. (4) to obtain updated $X$;

$j = j + 1$;

End while

Calculate the fitness value;

$i = i + 1$;

End while

Return fitness value and position of $X$;

End

2.2. Water cycle

2.2.1. Literature review

The concept of the “water cycle” is derived from the water cycle algorithm (WCA). WCA process is simple, intuitive, easy to understand, and has strong search capabilities. As a result, several scholars have investigated it to improve the algorithm itself and apply it within other fields. Chen et al. (Chen, Wang, Dong, & Wang, 2020) introduced the concept of hierarchical learning to improve the global search capability of the WCA and proposed the WCA for hierarchical learning (HLWCA), which was tested on the IEEE CEC 2017 benchmark set to demonstrate the effectiveness of the newly proposed algorithm. Korašy et al. (Korashy, Kamel, Youssef, & Jurado, 2019) put forward a modified version of the water circulation algorithm, namely MSMA, which can effectively solve the optimal coordination problem of the coordination optimization model of the directional current relay. Compared with the original WCA method, the improved method is more effective and superior. Osaba et al. (Osaba, Del Ser, Sadollah, Bilbao, & Camacho, 2018) proposed an enhanced discrete WCA (DWCA) to solve symmetric and asymmetric travel agents’ problems. To verify this
method’s effectiveness, DWCA was tested on 33 functions and compared with the simulated annealing algorithm, bat algorithm, and other six different algorithms. The results showed that this algorithm was due to any other algorithms. Pahehkeloei et al. (Pahehkeloei, Afifi, Sadollah, & Kim, 2017) improved the original WCA’s performance by introducing a local optimization mechanism called gradient-based. Twelve famous benchmark functions were used to test the algorithm, and the Friedman test verified the algorithm’s excellent performance. El-fargany et al. (El-Fargany & Hasanian, 2019) used WCA to solve the coordination problem of overcurrent relay and widely verified the advantages of WCA based on methodology through simulation and comprehensive comparison. Elhameed et al. (Elhameed & El-Fargany, 2017) used the efficient WCA method to solve the single-target and multi-target of economic load dispatching, aiming to produce the best active power generation value for each unit. Qiao et al. (Qiao, Zhou, Zhou, & Wang, 2019) used WCA and percolation operators to replace the original K-means algorithm for clustering operation. The algorithm abandoned the original precipitation process in WCA and only used the flow process and soakaway operator to combine. The proposed algorithm performs well in the final solution’s quality and speed on 10 test data sets.

2.2.2. Water cycle algorithm

WC mechanism is summarized from the WCA (Eskandar, Sadollah, Bahreininejad, & Hamdi, 2012) in the optimization algorithm. It may not be a very well-established method, but it has shown some applicability, and the operators are usable. The input populations’ fitness values are classified into three levels: superior to inferior, sea, river, and stream. First, the streams flow into the river, then the river into the sea, finally, the water’s evaporation and the rainfall. All of the above steps are accompanied by population renewal and reclassification by calculating their fitness values. So the cycle leads to optimal sea populations.

Since the WC is introduced here as an optimization mechanism, concepts such as initialization in the original WCA do not need to be implemented, only the process of updating the implemented algorithm. The update process includes mainly confluence update and evaporation, and precipitation.

a) Confluence update

Confluence processes are essential in algorithms and are iterative processes for exploring optimal solutions. After precipitation forms a stream, it divides the stream into two parts, one of which flows into the river, while the best streams flow directly into the sea. During the stream’s flow process, new streams will be formed due to the change of location. If the stream’s fitness value after the renewal is better than the river’s fitness value, the location will be exchanged. The same is true of rivers flowing into the ocean. The location update formula is as follows:

\[ X’_{\text{Stream}} = X_{\text{Stream}} + \text{rand} \times C \times (X’_{\text{Sea}} - X’_{\text{Stream}}) \]  
(5)

\[ X’_{\text{Stream}} = X_{\text{Stream}} + \text{rand} \times C \times (X_{\text{River}} - X’_{\text{Stream}}) \]  
(6)

\[ X’_{\text{River}} = X_{\text{River}} + \text{rand} \times C \times (X’_{\text{Sea}} - X’_{\text{River}}) \]  
(7)

rand appearing in the equation above are random numbers between 0 and 1 in the D-dimension, and C is the value of the interval (1,2), usually taking the empirical value 2.

b) Evaporation and precipitation

In the WCA, there is a process of evaporation of water, but the evaporation conditions need to be met, and the evaporation conditions are to judge whether the distance between the stream and the sea or the river and the sea is less than a very small value \( d_{\text{max}} \). The value of \( d_{\text{max}} \) determines the balance between the flow process and the rainfall process. Its formula is as follows:

\[ d^{(i+1)}_{\text{max}} = d^{(i)}_{\text{max}} - \frac{d^{(i)}_{\text{max}}}{\text{max} \_it} \]  
(8)

where max_it is the maximum iterations, and \( d_{\text{max}} \) decreases as the number of iterations increases. If evaporation conditions are met, all streams form a rainfall process on a global scale, i.e., updating stream locations on a global scale, with the following updating formula:

\[ X’_{\text{Stream}} = LB + (UR - LB) \times \text{rand} \]  
(9)

where UB and LB denote upper and lower limits, rand is random numbers between 0 and 1.

Algorithm 2 Pseudocode of WCA

```
Input the position and fitness value of X;
Sort fitness values in reverse order;
According to the fitness value, X is divided into streams, rivers, and sea;
Moving stream to sea;
Update the stream by Eq. (5);
Calculate the fitness value of stream and compare it with the fitness value of sea;
Update the position and fitness value of stream and compare it with the fitness value of sea;
Moving river to the sea;
Update the stream by Eq. (7);
Calculate the fitness value of the river and compare it with the fitness value of the sea;
Update the position and fitness value of sea if there is a better one;
Moving stream to the river;
Update the stream by Eq. (6);
Calculate the fitness value of the stream and compare it with the fitness value of the river;
Update the position and fitness value of the river if there is a better one;
Update the position of all streams based on evaporation conditions and raining process;
Return fitness value and position of X;
```

2.3. Slime mould algorithm

SMA is a meta-heuristic algorithm proposed based on the foraging slime mold behavior. The slime mould relies primarily on propagating waves generated by biological oscillators to alter the cytoplasmic flow in the vein to approach higher food concentrations, then surround the food and secrete enzymes to digest it. The following mathematical model is used to represent the behavior of slime bacteria to obtain food. Also, the general logic of the SMA is expressed in Fig. 1.

\[ X(t + 1) = \begin{cases} X(t) + W \cdot (\bar{W} \cdot X_{\text{A}}(t) - X(t)) & r < p \\ X(t) & r \geq p \end{cases} \]  
(10)

The above formula expresses a model of the constant contraction of slime mould populations as they approach food. Where both \( W \) and \( \bar{W} \) are parameters, \( \bar{W} \) has a range of \([-a, a]\), and \( \bar{W} \) decreases from one to zero with the number of iterations, \( X_{\text{A}} \) indicates the location of the food with the highest odour concentration of all foods found to date, \( \bar{X} \) indicates the location of the current slime mould population, \( X_{\text{A}} \) and \( X_{\text{A}} \) are the locations of randomly selected populations of slime mould, \( W \) is the corresponding weight, \( r \) denotes the random value in the interval of [0, 1].

The value of \( p \) is calculated by the following formula:

\[ p = \tanh |S(t) - DF| \]  
(11)
where \( i \in 1, 2, 3 \ldots n \), \( S(i) \) signifies the fitness of \( X_i \), \( DF \) denotes the best fitness gotten in all iterations.

The value of \( a \) in the range of \( \frac{1}{2} \) is calculated as follows:

\[
a = \arctan\left(-\left(\frac{t}{\max_t}\right) + 1\right) \quad (12)
\]

where \( t \) indicates the number of current iterations and \( \max_t \) indicates the maximum number of iterations.

The weight \( \tilde{w} \) is calculated as follows:

\[
\tilde{W}(\text{SmellIndex}(i)) = \begin{cases} 
1 + r \times \log \left(\frac{W-S(i)}{W-S(\max)}+1\right) & \text{condition} \\
1 - r \times \log \left(\frac{W-S(i)}{W-S(\max)}+1\right) & \text{other}
\end{cases} \quad (13)
\]

\[
\text{SmellIndex} = \text{sort}(S) \quad (14)
\]

where condition specifies that \( S(i) \) is in the first half of the swarm, \( r \) means the random value in the interval of \([0, 1]\), \( W \) means worst fitness value found in the present iterative process, \( W \) means the best fitness gotten in the present iterative process, \( S(i) \) means the fitness value of all populations at a given iteration, \( \text{SmellIndex} \) is the result of an ascending classification of \( S \) (the minimum value problem). Eq. (13) graphically simulates the positive and negative feedback between the vein width of the slime mould and the concentration of the food being probed. The original authors of the paper, after a large number of experiments, concluded that when the random value is less than 0.03, there is another way to obtain the minimum value, or to allow the slime mould to obtain a high concentration of food, which is calculated as follows:

\[
X^i = \begin{cases} 
\text{rand} \times (UB - LB) + LB & \text{rand} \leq z \\
\frac{X_{\min}(t) + \text{rand} \times (W - X_{\min}(t))}{W - X(t)} & r < p \\
\frac{X_{\max}(t) + \text{rand} \times (W - X_{\max}(t))}{W - X(t)} & r \geq p
\end{cases} \quad (15)
\]

where \( z \) is 0.0.3, \( UB \) and \( LB \) denote the lower and upper limitations of the search range, \( \text{rand} \) indicates that a random number between \([0,1]\) is generated at random.

The population’s position can be continually changed by Eq. (15). The fitness value is calculated and then compared with the best value found to date, and if a better value exists, it is replaced. Algorithm 1 presents the pseudo of the SMA.

$$
\text{Algorithm 3 Pseudocode of SMA}
$$

1. Initialize the parameters \( \text{PopSize}, \text{Max\_iteration} \);
2. Initialize the position of slime mould \( X(i) = 1, 2, 3 \ldots n \);
3. Initialize some of the remaining variable parameters;
4. \( \text{While} (t < \text{Max\_iteration}) \)
5. Calculate fitness of all slime mould populations and sort them in ascending order;
6. Update bestFitness, \( X_t \);
7. Calculate the \( \tilde{w} \) of each slime mould by Eq. (13);
8. \( \text{For} \) each search agents
9. \( \text{update} \) \( \mu, v, \omega \);
10. \( \text{update} \) position by Eq. (15);
11. \( \text{End For} \)
12. \( t = t + 1 \);
13. \( \text{End While} \)
14. \( \text{Return} \) bestFitness, \( X_t \);

3. Proposed WQSMA

In this study, QRG and WC mechanisms are added to the original SMA, making the original algorithm more balanced in terms of exploration and exploitation. In the case of QRG, the aim is to determine if there is a more optimal solution between the hitherto optimal solution and the current solution. The rotation is not too strong, and the search can be performed locally, satisfying the concept of exploration. As far as the WC mechanism is concerned, it can increase the search scope, expand the solving space, and fulfill the exploitation concept. Combining these two mechanisms, we derive the flowchart of WQSMA, as shown in Fig. 2.
4. Experimental results and discussions

In this section, we conduct a series of comparative experiments on the proposed WQSMA based on fair comparisons (Chen, Zheng, Li, & Huang, 2021; Shi, Lu, & Zhang, 2019; Sun, Li, & Deng, 2021; Zhang et al., 2021). The first step is a study comparing the effects of individual mechanisms on SMA. The second is to perform a scalability test to verify the algorithm’s performance changes under different dimensional conditions. The third part compares WQSMA with several original and advanced algorithms. The fourth part is to balance and diversity analysis of different mechanisms. Finally, we apply the WQSMA to engineering problems.

The average result (Avg.) and standard deviation (Std.) were utilized to assess the performance of WQSMA, where the optimal outcome for each case will be bolded. Among them, "+/−/−" respectively indicates whether this paper's algorithm is superior to, equal to, or inferior to other algorithms. The Wilcoxon signed-rank test (Garcia, Molina, Lozano, & Herrera, 2009) is used to evaluate the algorithm's average performance in the statistical sense. In this study, it is used to test whether there is a difference between the effect of WQSMA and the other algorithms after pairwise comparison. When the p-value is less than 0.05, the result is significantly different from other methods. If not, the improvement is not statistically significant. Besides, the average ranking value (ARV) was utilized to signify the results. To level the playing field, we set the same public parameters for testing, with 1000 iterations, 30 populations, and 30 dimensions (except for the different dimension settings in Sec. 4.3).

We used Windows Server 2012 R2 OS and a set of codes in MATLAB R2014a software, and the hardware details are as follows: Intel (R) Xeon (R) Sliver 4110 CPU (2.10 GHz) and 16 GB RAM.
4.1. Benchmark function validation

In this study, the test set for the comparative experiment on WQSMA is CEC 2014, so in this subchapter, we describe in detail the function sets that appear in the CEC 2014 test set. The relevant details are shown in Table 2. The CEC 2014 test set divides functions into four categories: unimodal functions ($f_1 - f_5$), simple multimodal functions ($f_6 - f_{10}$), hybrid functions ($f_{17} - f_{20}$), composition functions ($f_{21} - f_{30}$).

4.2. Impacts of components

The WQSMA proposed in this paper incorporates two different mechanisms based on the original SMA. To verify the effect of each of the two mechanisms, they are compared separately in this section. These versions are:

- SMA based on QRG and WC (WQSMA)
- SMA based on WC (WSMA)
- SMA based on QRG (QSMA)
- Original SMA

The results of the judgment are exposed in Appendix 1. Appendix 1 gives the results of the comparison between the original SMA and the improved algorithm after adding the mechanism. Improved algorithms include those with a single mechanism (WSMA and QSMA) and those with two mechanisms (WQSMA). At the end of the table, the ranking of the four algorithms is given, and it can be seen that in the first place is WQSMA. This ranking is based on the ARV indicator, which is averaged by obtaining the algorithm’s ranking on each function. Through the ARV value, it can be concluded that the advantage of WQSMA is quite apparent. In those cases, the rank from the best to worst is as follows approximately: WQSMA > QSMA > WSMA > SMA. With the addition of dual mechanisms, the SMA has much more reliable performance and a good improvement in search capabilities. It is also shown that QRG and WC play a significant role in exploration and exploitation. Comparing QSMA with WSMA, we can find that QSMA is much stronger than WSMA, reflecting that QRG contributes more to the performance improvement of SMA than WC does to the performance improvement of SMA. For the original SMA, the ARV value is the same as WSMA, but in comparison with WQSMA, WSMA is not as effective as WQSMA in 17 functions, while SMA reaches 21. Consequently, in summary, the SMAs have all been improved by adding the mechanism.

The Appendix 2 shows the Wilcoxon’s signed-rank test results of WQSMA compared with SMA after adding different mechanisms. Null values appearing in the table all indicate p-values greater than 0.05 set in advance, which means that these comparisons’ results are not statistically significant. Twelve of the results of the comparison of WQSMA with WSMA and QSMA are not statistically significant. In contrast, five of the results of the comparison with SMA are not statistically significant. However, after removing these non-statistically significant results, WQSMA still achieves better results than the algorithm, so it can be exposed that the competence of SMA has been dramatically improved with the addition of both mechanisms.

4.3. Scalability test

In this section, to better test the performance of WQSMA when compared with other algorithms, scalability tests are performed. Scalability tests can help us better understand how an algorithm performs in different dimensions and can reflect whether the algorithm will experience performance fluctuations at a higher dimension. The test set cho-

<table>
<thead>
<tr>
<th>No.</th>
<th>Function</th>
<th>$F_n^* = F_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Rotated High Conditioned</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>Elliptic Function</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Rotated Cigar Function</td>
<td>200</td>
</tr>
<tr>
<td>4</td>
<td>Rotated Discus Function</td>
<td>300</td>
</tr>
<tr>
<td>5</td>
<td>Shifted and Rotated</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Rosenbrock's Function</td>
<td>400</td>
</tr>
<tr>
<td>7</td>
<td>Shifted and Rotated</td>
<td>500</td>
</tr>
<tr>
<td>8</td>
<td>Ackley's Function</td>
<td>600</td>
</tr>
<tr>
<td>9</td>
<td>Weierstrass Function</td>
<td>700</td>
</tr>
<tr>
<td>10</td>
<td>Griewank's Function</td>
<td>800</td>
</tr>
<tr>
<td>11</td>
<td>Shifted and Rotated</td>
<td>900</td>
</tr>
<tr>
<td>12</td>
<td>Schewefel's Function</td>
<td>1000</td>
</tr>
<tr>
<td>13</td>
<td>Rastrigin Function</td>
<td>1100</td>
</tr>
<tr>
<td>14</td>
<td>HappyCat Function</td>
<td>1200</td>
</tr>
<tr>
<td>15</td>
<td>Katsuura Function</td>
<td>1300</td>
</tr>
<tr>
<td>16</td>
<td>Function</td>
<td>1400</td>
</tr>
<tr>
<td>17</td>
<td>Expanded Griewank's Function</td>
<td>1500</td>
</tr>
<tr>
<td>18</td>
<td>Expanded Griewank's Function</td>
<td>1600</td>
</tr>
<tr>
<td>19</td>
<td>Expanded Griewank's Function</td>
<td>1700</td>
</tr>
<tr>
<td>20</td>
<td>Expanded Griewank's Function</td>
<td>1800</td>
</tr>
<tr>
<td>21</td>
<td>Expanded Griewank's Function</td>
<td>1900</td>
</tr>
<tr>
<td>22</td>
<td>Expanded Griewank's Function</td>
<td>2000</td>
</tr>
<tr>
<td>23</td>
<td>Expanded Griewank's Function</td>
<td>2100</td>
</tr>
<tr>
<td>24</td>
<td>Expanded Griewank's Function</td>
<td>2200</td>
</tr>
<tr>
<td>25</td>
<td>Expanded Griewank's Function</td>
<td>2300</td>
</tr>
<tr>
<td>26</td>
<td>Expanded Griewank's Function</td>
<td>2400</td>
</tr>
<tr>
<td>27</td>
<td>Expanded Griewank's Function</td>
<td>2500</td>
</tr>
<tr>
<td>28</td>
<td>Expanded Griewank's Function</td>
<td>2600</td>
</tr>
</tbody>
</table>
sen for this experiment is the 30 functions in CEC 2014, and the dimensional choices are $50 - D$, $100 - D$ ($D$ indicates dimension). Simultaneously, the comparison functions chosen for the experiment are WQMSA, WSSA, QMSA, and SMA. In the experiment, the environment, variables, etc. remained constant except for dimensional changes. The results are shown in Appendix 3.

From Appendix 3, it can be seen that WQMSA has more significant advantages in all aspects of the comparison process with other algorithms. This table shows the test results of the algorithm in $50 - D$ and $100 - D$. Combined with Appendix 1 (Appendix 1 is the comparison results of WQMSA and single mechanism algorithm as well as the original algorithm in $30 - D$), it can be seen that with the increase of dimensions, the optimal solution of each algorithm will be worse to a certain extent, because the dimensions become larger, the algorithm processing will become more complex and more challenging. However, the WQMSA comparison results in different dimensions have not changed much, and the first-ranked solution can still be obtained for most functions. Therefore, it can be concluded that WQMSA has good scalability.

### 4.4. Comparison with the well-known algorithms

In this section, WQMSA is compared with some classic and novel meta-heuristics algorithms. The external parameters set as shown in Table 3. We compared these methods: DE, GWO, WOA, MFO, gravitational search algorithm (GSA) (Rashedi, Nezamabadi-pour, & Saryazdi, 2009), SSA, multi-verse optimization algorithm (MVO) (S. Mirjalili, Mirjalili, & Hatamlou, 2016), BA, PSO, sine cosine algorithm (SCA) (Mirjalili, 2016), GA (Holland, 1992), artificial bee colony (ABC) (Karaboga & Basturk, 2007), firefly algorithm (FA) (Yang, 2009).

The results of the comparison of WQMSA with the above algorithms are shown in Appendix 4. The Wilcoxon’s signed-rank test results of WQMSA compared with well-known algorithms are presented in Appendix 5. The data presented in the table are the Avg. and Std. of the algorithms after running them on each function. Also, ARV values and “+”/”−” indicator data are presented at the end of the table.

It can be seen from the table in Appendix 4 that WQMSA can obtain better solutions on 11 of the functions than other algorithms, and the top three solutions can be obtained on 19 of the functions. Combining the ARV values, it can be seen that WQMSA is the only one of all algorithms with a value of less than 3.5. Finally, looking at the “+”/”−” indicator, WQMSA can beat the second-ranked FA algorithm in 15 functions; only ten functions are worse than it, the rest functions are not significant because the p-value is less than 0.05. In comparison, WQMSA is better than the last-ranked SCA in 30 functions.

Fig. 3 shows the convergence of WQMSA with each well-known algorithm throughout 1000 iterations, where two functions of each type are taken as representatives.

On the unimodal function, we use $F_1$ and $F_3$ as representatives for the analysis. For the $F_1$ function, WQMSA can rank first among all the algorithms, and at the same time, the advantage of WQMSA over other algorithms can be clearly seen by the first subgraph in Fig. 3, which is far ahead in terms of convergence speed and accuracy. In terms of $F_1$ function, the convergence effect of DE and ABC is very close to WQMSA. In particular, the DE algorithm’s convergence curve almost coincides with WQMSA, but the subtle difference between them can be seen from the specific data.

The performance of WQMSA is the worst among the four kinds of functions on simple multimodal functions. We take $F_4$ and $F_5$ as the representative to conduct the analysis. For $F_4$ and $F_10$ functions, WQMSA ranks third among all algorithms. On the $F_4$ function, by observing the convergence curve, we can see that the gap between WQMSA and the top two is very small. Therefore, we have enlarged the convergence effect of the last 100 iterations curve to make it easier to observe more clearly. On the $F_10$ function, there is a certain gap between the performance of WQMSA and the top two, especially in comparison with FA, the gap is more obvious. In terms of specific data, WQMSA and the second-ranked DE algorithm are close to each other. Although there is a certain gap between WQMSA and the first-ranked ABC algorithm, WQMSA tends to continue to converge at the end of the 1000th iteration, while ABC has leveled off.

On the hybrid function, we illustrate with $F_{30}$ and $F_{21}$ as representatives. For the $F_{30}$ and $F_{21}$ functions, WQMSA ranks first among all algorithms. Through the graphs, we can find that in both functions, the MVO algorithm poses a greater threat to WQMSA. In the $F_{30}$ function, the final convergence accuracy of the MVO algorithm is 6663, while WQMSA can reach 5535, and the difference between them is as much as 1000. In the $F_{21}$ function, the final convergence accuracy of the MVO algorithm is 21017, while WQMSA is only 19589, which also has a gap of more than 1000, and WQMSA is the only algorithm that achieves convergence accuracy within 20,000 in the $F_{21}$ function.

Finally, the composition function is presented, and WQMSA works best on this type of function. There are eight composition functions in CEC 2014 test set, and WQMSA can rank first among all algorithms in 7 of them. In the convergence curve represented by $F_{24}$ and $F_{30}$, it can be seen that WQMSA is much better than other algorithms both in terms of convergence accuracy and convergence speed; moreover, WQMSA can converge to the optimal value quickly in the first 200 iterations.

### 4.5. Comparison with the advanced algorithms

In this section, to further verify the performance and benefits of WQMSA, we compare it with the advanced functions. The compared methods are:

<table>
<thead>
<tr>
<th>Table 3</th>
<th>Parameters setting for meta-heuristic algorithms.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm</td>
<td>Population size</td>
</tr>
<tr>
<td>DE</td>
<td>30</td>
</tr>
<tr>
<td>GWO</td>
<td>30</td>
</tr>
<tr>
<td>WOA</td>
<td>30</td>
</tr>
<tr>
<td>MFO</td>
<td>30</td>
</tr>
<tr>
<td>GSA</td>
<td>30</td>
</tr>
<tr>
<td>SSA</td>
<td>30</td>
</tr>
<tr>
<td>MVO</td>
<td>30</td>
</tr>
<tr>
<td>BA</td>
<td>30</td>
</tr>
<tr>
<td>PSO</td>
<td>30</td>
</tr>
<tr>
<td>SCA</td>
<td>30</td>
</tr>
<tr>
<td>GA</td>
<td>30</td>
</tr>
<tr>
<td>ABC</td>
<td>30</td>
</tr>
<tr>
<td>FA</td>
<td>30</td>
</tr>
</tbody>
</table>

where $\alpha = 0.5; \beta = 0.2; \gamma = 1$. \( N = 30, \) $F_{2}^{*} = F_{2}(x^{*})$ \( s \in \mathbb{R}^{*}$.
Fig. 3. Convergence curves of WQSMAS compared with well-known algorithms on CEC 2014.
• Self-adaptive differential evolution (SaDE) (Qin & Suganthan, 2005)
• Chaotic local search sine cosine algorithm (CSCA) (Tu et al., 2019)
• Covariance guided sine cosine algorithm (COSCA) (G. Liu et al., 2020)
• Improved grey wolf optimization algorithm (IGWO) (Cai, Gu, Luo, & Zhang, 2019)
• Opposition-based learning grey wolf optimization algorithm (OBLGWO) (Heidari, Abbaspour, & Chen, 2019)
• Comprehensive learning particle swarm optimizer (CLPSO) (J. J. Liang et al., 2006)
• Chaotic bat algorithm (CBA) (Adarsh, Raghunathan, Jayabarathi, & Yang, 2016)
• Bat algorithm with random wind power (RCBA) (Liang, Liu, Shen, Li, & Man, 2018)
• Balanced whale optimization algorithm (BWOA) (Chen, Xu, Wang, & Zhao, 2019)
• An enhanced associative learning-based exploratory whale optimizer (BMWOA) (A. A. Heidari et al., 2019)
• Ensemble particle swarm optimizer (EPSO) (Lynn & Suganthan, 2017)
• A multi-strategy enhanced sine cosine algorithm (MSCA) (H. Chen et al., 2020)
• Particle swarm optimization with an aging leader and challengers (ALPSO) (Chen, Zeng, Lu, & Weng, 2013)

Appendices 6 and 7 show the results of the comparison of WQSMA with the advanced function and Wilcoxon’s signed-rank test results, respectively. Fig. 4 shows the trend plot of WQSMA versus advanced algorithms on partial functions.

We first interpret the p-value obtained by the rank test. As mentioned earlier, the results of WQSMA compared with other algorithms are only meaningful when the p-value is less than 0.05. In Appendix 7, it can be seen that the p-values are greater than 0.05, where there are null values, so it can be considered that the difference between WQSMA and the corresponding algorithm is not very significant, and we will not include it in the calculation of the final ranking results.

Then we analyze the table in Appendix 6. At the end of the table, the final ranking and “±/−/+” indicator data are given according to ARV value. It can be seen that WQSMA has the strongest performance among all advanced functions, and its ARV value is less than 3.0, far ahead of the second 4.03. According to the “±/−/+” indicator, WQSMA can surpass any advanced algorithm in at least 9 functions on the CEC 2014 test set.

Finally, do a completed analysis of WQSMA in conjunction with the convergence curves. WQSMA can be ranked first among all algorithms in 9 out of 30 tested functions. From each category of functions, we take two of them as representatives to analyze.

The downward trend of WQSMA and the advantages of other algorithms on the unimodal function are clear. For the F1 function, different algorithms exhibit different performance on the function, and the difference is relatively large. However, WQSMA is still able to stand out from many algorithms and ranked first. Also, the EPSO algorithm is very close to WQSMA in terms of convergence effect, but the data analysis can show that the gap between them is still very large. For F3 WQSMA ranks second among all algorithms, but it is very close to SaDE, which ranks the first in convergence accuracy, with a difference of less than 0.00.

The effect of WQSMA on multimodal functions is worse than that of other types of functions, but it can still be at the forefront of all algorithms. We analyzed functions F5 and F12. For F5 function, the convergence of WQSMA has been completed in the first 200 iterations, and the convergence accuracy is far better than other algorithms. For F12 function, although WQSMA is not as accurate as RCBA, which ranks first in accuracy, the difference between them is only 1%, so it can be said that the performance of the two is the same. Besides, we can also see from the figure that WQSMA has a downward convergence trend at the 1000th iteration.

The last two lines of the figure present the hybrid and composition functions, respectively. Both of these functions are complex and, therefore, naturally more time-consuming and challenging to run than unimodal and simple multimodal functions.

In the figure, we can see that for the F20 function, the convergence accuracy of WQSMA is ranked first. While the accuracy of SaDE is very close to that of WQSMA, the two curves almost overlap at the end of the iteration, so we have zoomed in on this region of the curve to observe the difference more clearly. For the F21 function, WQSMA ranks second, and although there is a gap between it and SaDE, it still deserves recognition in terms of convergence speed, which has been performed in the first 100 iterations.

In terms of composition functions, WQSMA arguably performs the best. Six of the eight composition functions rank first and are also the fastest in terms of convergence speed. In the illustration of the composition function, all algorithms’ optimal solution gaps are not very large. They are therefore superimposed together, so they are enlarged with small plots.

4.6 Balance analysis

In this study, two mechanisms, WC and QRG, are added to the original SMA; therefore, we will focus on the impact of the two mechanisms on the SMA in this section.

We have selected five of the 30 functions from CEC 2014 that are representative and have more pronounced effects as examples to dissect, namely F13, F15, F26, F27 and F28. The study results are shown in Fig. 5, where each row of the graph represents a specific result on a particular function. Also, the first column (a) is a three-dimensional plot of the corresponding function, and the second column (b) is a two-dimensional plot of the position of the algorithm at each search on the function. The third column (c) is the trajectory of WQSMA, and the last column shows the average fitness of WQSMA. The images in columns (a) and (b) correspond to each other. The optimal solution of the function can be found in a certain region by (a), and then determine whether the red dot in column (b) falls in the region (the red dot is the position of the optimal solution obtained by WQSMA on the function).

As shown in the column in Fig. 5 (b), the historical search range of WQSMA is full of the whole solution space. Meanwhile, in the vicinity of the optimal solution, the number of searches is more than that of other regions, reflecting the strong search ability of WQSMA and increasing the population’s diversity. In Fig. 5 (c), the magnitude of the trajectory change of the population is shown. It can be seen that overall, the fluctuations vary from large to small. The early fluctuations are large, indicating that the WQSMA has good search capabilities.

In contrast, the later fluctuations become slower but still present, indicating that throughout the process, WQSMA is continuously looking for the optimal solution, demonstrating the superiority of its search. The average fitness value during each iteration when the last column of images is taken. It generally becomes progressively more stable, but at some points in the last two images, there is a significant increase, which can be phenomenally reflected in the WQSMA jumps out at the local optimal solution during the search. Since the local optimal solution tends to have high values of nearby adaptation, the average adaptation value rises sharply in the pop-out’s initial phase. Still, its value must drop to a minimum at a later stage, moving closer to the optimal solution.

To better understand the development trend of exploration and exploitation in the process of searching for the optimal solution, the SMA...
Fig. 4. Convergence curves of WQSMO compared with advanced algorithms on CEC 2014.
composed of different mechanisms is analyzed below. Fig. 6 shows a balanced analysis of the four algorithms. There are three curves in each picture. The red curve represents the exploration process of the algorithm, the blue curve represents the exploitation process of the algorithm, and the green curve represents the incremental-decremental curves. As can be seen from the figure, when the exploration effect (percentage) is better than the exploitation effect, the green curve increases; otherwise, it decreases. When the curve falls to a negative number, it is set to 0. Therefore, according to the green curve, we can clearly understand how the algorithm balances the relationship between exploration and exploitation in the whole iterative process. If the value of the green curve is high, it means that the exploration effect is good at this time; otherwise, it means that the exploitation effect is good at this time. Moreover, when the curve is at its highest, it means that the two are in equilibrium.

As you can see from the figure, all three algorithms, except WSMA, spend much time on the exploitation process to get a better solution. The exploration process is often much shorter than the exploitation process. For WSMA, the WC mechanism is added to the original SMA, and the primary function of the WC mechanism, as mentioned above, is to increase the search range of the algorithm in the solution space, so it makes sense that the exploration process is always larger than the exploitation process for WSMA. As can be seen from the original SMA’s effect figure, the exploration part only played a role in the first few iterations and has been in the exploitation process since then, which is also the key reason why SMA will fall into the local optimum. Finally, QSMA and WQSMA have little difference in effect. The primary purpose of QSMA is to increase the exploitation process, but at the same time, it also promotes the exploration process. WQSMA absorbs WSMA and QSMA’s advantages to the greatest extent; it makes the algorithm keep the balance of exploitation and exploration.

4.7 Application to engineering problems

Through the above analysis, we obtain that the proposed WQSMA has better superiority than other algorithms, reflecting the existence of
better performance of WQSMA at the theoretical level. In this section, we apply WQSMA to three constrained application problems: tension/compression string problem, welded beam design problem (WDB), and pressure vessel design problem (PVD), to verify the performance embodiment of WQSMA at the practical application level. In life, a practical engineering problem is often constrained by many realistic conditions, such as how to maximize benefits with minimal expense. Translating this into a mathematical model requires taking all these factors into account and adding constraints. Therefore, finding a way to deal with these constraints becomes essential. We have studied some penalty functions introduced in (Coello Coello, 2002), such as co-evolutionary, death penalty, etc. We are free, but here the death penalty is considered a moderate function to construct the mathematical model's initial target value to be processed. Therefore, we tested WQSMA with the death penalty function.

4.7.1. Tension/Compression string problem

This problem aims to minimize the tension/compression spring weight. The model contains the following three parameters:

- wire diameter (d)
- mean coil diameter (D)
- number of active coils (N)

Its mathematical model is as follows:

Consider \( \vec{x} = [x_1, x_2, x_3] = [D, D, N] \)

Minimize \( f(\vec{x}) = x_1^2 + x_2^2 (x_1 + 2) \)

Subject to \( g_1(\vec{x}) = 1 - \frac{4x_2}{1085x_1} \leq 0 \)

\( g_2(\vec{x}) = \frac{4x_1^2 - x_1 x_2}{12566(x_1^2 - x_3^2)} + \frac{1}{5108x_3^2} \leq 0 \)

\( g_3(\vec{x}) = 1 - \frac{140.45x_3}{x_2^2 x_3} \leq 0 \)

\( g_4(\vec{x}) = \frac{x_1 + x_2}{1.5} - 1 \leq 0 \)

Variable range: \( 0.05 \leq x_1 \leq 2, 0.25 \leq x_2 \leq 1.3, 2 \leq x_3 \leq 150 \)

To confirm the strength of the algorithm, we compared it to the following five methods:
• DE (Huang, Wang, & He, 2007)
• Evolution strategies (ES) (Mezura-Montes & Coello, 2008)
• PSO (He & Wang, 2007)
• GWO (Mirjalili, et al., 2014)
• WOA (Mirjalili & Lewis, 2016)

Table 4 shows the results of the comparison. From the table, it can be seen that when \( d = 0.051795 \), \( D = 359279 \), \( N = 11.140358 \), WQSMA yields the smallest cost of 0.012665 on this model. Simultaneously, this cost is also the smallest of all the solutions derived from the algorithm, so it can be shown that WQSMA has good results in solving tension/compression string problems.

4.7.2. Welded beam design problem

The WBD is a kind of composite beam, which is proposed by Coello (Coello Coello, 2000). The WBD problem is also one of the more classic engineering problems used to test the performance of the algorithm, intending to find the next four factors to minimize the price of welding beams:

• shear stress (\( \tau \))
• bending stress (\( \theta \))
• buckling load (\( P_c \))
• deflection (\( \delta \))

When the problem is decoded into a mathematical model, the following four parameters are included:

• welding seam thickness (\( h \))
• welding joint length (\( l \))
• beam width (\( t \))
• beam thickness (\( b \))

The mathematical model is as follows:
Consider \( \bar{x} = [x_1, x_2, x_3, x_4] = [h t b] \)
Minimize \( f(\bar{x}) = 1.10471x_1^2 + 0.04811x_3x_4(14.0 + x_4) \)
Subject to:
\[ g_1(\bar{x}) = \tau(\bar{x}) - \tau_{\text{max}} \leq 0 \]
\[ g_2(\bar{x}) = \sigma(\bar{x}) - \sigma_{\text{max}} \leq 0 \]
\[ g_3(\bar{x}) = x_1 - x_4 \leq 0 \]
\[ g_4(\bar{x}) = P - P_c(\bar{x}) \leq 0 \]
\[ g_5(\bar{x}) = 0.125 - x_1 \leq 0 \]
\[ g_7(\bar{x}) = 1.10471x_1^2 + 0.04811x_3x_4(14.0 + x_4) - 5.0 \leq 0 \]

Table 4
Comparison results for tension/compression string problem.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>d</th>
<th>D</th>
<th>N</th>
<th>Best Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>WQSMA</td>
<td>0.051795</td>
<td>0.359279</td>
<td>11.140358</td>
<td>0.012665</td>
</tr>
<tr>
<td>DE</td>
<td>0.051609</td>
<td>0.354714</td>
<td>11.41083</td>
<td>0.012670</td>
</tr>
<tr>
<td>ES</td>
<td>0.051989</td>
<td>0.363965</td>
<td>10.890522</td>
<td>0.0126810</td>
</tr>
<tr>
<td>PSO</td>
<td>0.015728</td>
<td>0.357644</td>
<td>11.244543</td>
<td>0.0126747</td>
</tr>
<tr>
<td>WOA</td>
<td>0.051207</td>
<td>0.345215</td>
<td>12.0043032</td>
<td>0.0126763</td>
</tr>
</tbody>
</table>

Table 5
Comparison results for WBD problem.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>The optimal value for variables</th>
<th>Best Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>WQSMA</td>
<td>0.18850, 3.56850, 9.10685, 20.0542</td>
<td>1.72129</td>
</tr>
<tr>
<td>SMA</td>
<td>0.20270, 3.52170, 8.89310, 0.21260</td>
<td>1.75350</td>
</tr>
<tr>
<td>MFO</td>
<td>0.15600, 4.004099, 9.862099, 0.214648</td>
<td>1.810680</td>
</tr>
<tr>
<td>CDE</td>
<td>0.203137, 3.542998, 9.033498, 0.206179</td>
<td>1.733462</td>
</tr>
<tr>
<td>GWO</td>
<td>0.20570, 3.47840, 9.03680, 0.20580</td>
<td>1.726240</td>
</tr>
</tbody>
</table>
4.7.3. Pressure vessel design problem

The last is the PVD design problem, which aims to minimize the pressure vessel shell's thickness to withstand the maximum pressure at the specified temperature. In the case of the modeling problem, the problem is mainly influenced by the following factors:

- shell \( (T_s) \)
- head \( (T_h) \)
- the inner radius \( (r) \)
- the range of the section minus head \( (l) \)
- The constraints are as follows:

The constraints are as follows:

\[
\begin{align*}
\vec{x} & = [x_1, x_2, x_3, x_4] = [T_s, T_h, R, L] \\
\text{Objective} & = f(\vec{x}) = 0.6224x_1x_2 + 1.7781x_3x_4^2 + 3.1661x_4x_2^2 + 19.84x_3^2 \\
\text{Subject to} & \quad g_1(\vec{x}) = -x_1 + 0.0193x_3 \leq 0 \\
& \quad g_2(\vec{x}) = -x_1 + 0.0095x_4 \leq 0 \\
& \quad g_3(\vec{x}) = -x_4 - 240 \leq 0 \\
\text{Variable ranges} & \quad 0 \leq x_2 \leq 99 \\
& \quad 10 \leq x_3 \leq 200 \\
& \quad 10 \leq x_4 \leq 200
\end{align*}
\]

Algorithms compared to WQSMA include:

- SMA
- BA (X. S. Yang & Sowmya, 2010)
- PSO
- GA (Coelho, 2002)
- GWO

Table 6 shows the final comparative results. It can be seen that the WQSMA can obtain the optimal spend value of 6059.9 when \( T_s = 0.81250 \) and \( T_h = 0.43750 \). In addition, the optimal solution is also much improved to 176.65344 in comparison with the original SMA. Except that, the optimal solution of the other four algorithms is more than 6060.

5. Conclusions and future work

In this study, an enhanced SMA called WQSMA is proposed by introducing two QRG and WC mechanisms into SMA, which improves the exploration and exploitation ability of the original SMA. This is because the QRG can enhance the original SMA's ability to search locally, while the WC can increase populations' diversity during the exploration phase. To fully prove the effectiveness of the proposed algorithm, the following experiments are carried out. First, we tested the effect of different SMA mechanisms, comparing the four algorithms of original SMA, WSMA, QSM A, and WQSMA; we concluded that SMA could reach its optimal potential under the joint action of both mechanisms. Secondly, the efficiency of WQSMA is tested in multi-dimension, and the effect is also verified. Thirdly, the proposed WQSMA method's effectiveness was confirmed by comparing it with an inclusive set of recognized algorithms, including DE, GWO, WOA, MFO, GSA, SSA, MVO, BA, PSO, SCA, GA, ABC, and FA. The comparative results reveal that WQSMA can attain more precise agents and meaningfully outperform all the other test participants. This superior performance of WQSMA can also be seen in convergence leannings. Then, WQSMA was compared with multiple advanced MAs, including SaDE, CSA, COSCA, IGWO, OBLGWO, CLPSO, CBA, RCBA, BWOA, BMWOA, EPSO, MSGA, and ALCPSO. We found that WQSMA finds better solutions and has faster convergence conduct than the other optimizers to realize the studied problems. Finally, in the process of WQSMA running, the balance between exploration and exploitation is analyzed. In addition to the above verification tests, we also applied WQSMA to engineering problems to test its operational ability in practical problems. The experimental results prove that WQSMA can also achieve good results in solving engineering problems, significantly improving the previous solutions.

While WQSMA has proven to be an effective solution to the optimization problem, it still has some limitations to be considered as a future attempt; the main one is time complexity. Compared with the basic SMA, the new approach uses more amount of time to reach a convergence state. Many areas need further research. For example, we can utilize the proposed WQSMA for mechanical optimization (Cai, Gao, et al., 2020), smart grid (Wang, Liu, & Choo), intelligent damage detection (Kordestani & Zhang, 2020; Kordestani, Zhang, & Shadabfar, 2020; Mousavi, Zhang, Masri, & Gholiopour, 2020), image and video processing (Hinjoosa, et al., 2020; Jiang, et al., 2018; W. Liu et al., 2020; T. Wang, Zhao, Huang, & Xu, 2021; C. Yan, et al., 2020; M. Yang & Sowmya, 2015; X. Zhang, J. Wang, et al., 2020; X. Zhang, Wang, Luo, & Huang, 2020), social evolution modeling (Xue, Wang, Zhan, Feng, & Guo, 2019), image editing (Huang, et al., 2015; X. Li, Huang, Zhao, Wang, & Hu, 2020; X. Li, et al., 2016; Y. Yang, et al., 2017; H. Zhao, et al., 2018; Y. Zhao, et al., 2014), image retrieval (Zenggang, Zhiwen, & Xiaowen, 2021), fault diagnosis (Deng et al., 2020; Gao, Tang, Xiang, & Zhong, 2018; Liu, Huang, & Xiang, 2020; Song, Xiang, & Zhong, 2018; Wang & Xiang, 2020; Wang, Xiang, Zhong, & Zhou, 2018; Huimin Zhao, Liu, Xu, & Deng, 2019), and remote sensing (Yang, et al., 2018). Also, the idea of the multi-objective and many-objective variants of WQSMA is our next plans.

Table 6

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>The optimal value for variables</th>
<th>Best Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>WQSMA</td>
<td>0.81250 0.43750 0.80790 0.176544</td>
<td>6059.87991</td>
</tr>
<tr>
<td>SMA</td>
<td>0.81250 0.43750 0.74436 0.18230219</td>
<td>6131.79252</td>
</tr>
<tr>
<td>BA</td>
<td>0.81250 0.43750 0.0721 0.17690437</td>
<td>6062.40627</td>
</tr>
<tr>
<td>PSO</td>
<td>0.81250 0.43750 0.9217 0.17674650</td>
<td>6061.07770</td>
</tr>
<tr>
<td>GA</td>
<td>0.93750 0.50000 0.98200 0.1126790</td>
<td>6410.38110</td>
</tr>
<tr>
<td>GWO</td>
<td>0.81250 0.43750 0.09445 0.17670224</td>
<td>6060.57791</td>
</tr>
</tbody>
</table>

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (62076185, U1809209), Zhejiang Provincial Natural Science Foundation of China (LY21F020001), Wenzhou Science & Technology Bureau (ZG20200026), the Natural Science Foundation of Hel-


