Expert Systems With Applications xxx (xxxx) 114529

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: http://ees.elsevier.com

Memetic Harris Hawks Optimization: Developments and perspectives on project scheduling and QoS-aware web service composition☆

ChenYang Li [a], Jun Li [a,*], HuiLing Chen [a], Ali Asghar Heidari [b,c,1]

[a] College of Computer Science and Artificial Intelligence, Wenzhou University, Wenzhou, Zhejiang 325035, China
[b] School of Surveying and Geospatial Engineering, College of Engineering, University of Tehran, Tehran, Iran
[c] Department of Computer Science, School of Computing, National University of Singapore, Singapore, Singapore

ARTICLE INFO

Article history:
Received 28 April 2020
Received in revised form 13 October 2020
Accepted 20 December 2020
Available online xxx

Keywords
Harris hawks optimization
Evolution-based algorithm
Gaussian probability distribution
Numerical optimization functions
QoS-aware web service composition
Resource-constrained project scheduling

ABSTRACT

Harris hawks optimization (HHO) is one of the leading optimization approaches due to its efficacy and multi-choice structure with time-varying components. The HHO has been applied in various areas due to its simplicity and outstanding performance. However,the original HHO can be improved and evolved in terms of convergence trends, and it is prone to local optimization under certain circumstances. Therefore, the performance and robustness of the algorithm need to be further improved. In our research, based on the core principle of evolutionary methods, we first developed an elite evolutionary strategy (EES) and then utilized it to advance HHO's convergence speed and ability to jump out of the local optimum. We named such an enhanced hybrid algorithm EESHHO in this paper. To verify the effectiveness and robustness of the EESHHO, we tested it on 29 numerical optimization test functions, including 23 classic basic test functions and 6 composite test functions from the IEEE CEC2017 special session. Moreover, we apply the EESHHO on resource-constrained project scheduling and QoS-aware web service composition problems to further validate the effectiveness of EESHHO. The experimental results show that proposed EESHHO has faster convergence speed and better optimization performance by comparing it with other mainstream algorithms. The supplementary info and answers to possible queries will be publicly available at https://aliasgharheidari.com/publications/EESHHO.html. Also, the codes and info of HHO are available at: https://aliasgharheidari.com/HHO.html.

1. Introduction

As technological developments grow and economic activities increase, governments face more and more new problems and projects that need to be understood theoretically and practically (Hu et al., 2020; Qiu et al., 2019; Zhang, Chen, Wang, Liu, & Chen, 2021). For instance, such real-world projects in the product oil pipeline have many variables, limited resources, and budgets (Liu, Li, Cai, & Peng, 2019). Another instance is disaster-relief scenarios and tourist industry that we cannot obtain all resources similar to commonplace cases (Fu, Fortino, Li, Pace, & Yang, 2019; Lv, Li, Xu, & Yang, 2020). In such cases, decision support systems also come into the process for subsequent big data analysis, which also includes more factors into the problem (Lv & Qiao, 2020). In the area of routing protocols, we should also optimize many variables regarding the performance of the network's energy and reliability (Fu, Fortino, Pace, Aloi, & Li, 2020). Therefore, based on such examples, real-world problems always have limited resources and budgets, and there is a restriction on their variables (bound-constrained), and searching for some feasible optimal solutions during a reasonable time is required.

Finding feasible and optimal solutions to real-world problems using computationally efficient techniques and valid models is the focus of attention in machine learning, environmental modeling (Wang, Zhang, van Beek, Tian, & Bogaard, 2020), image processing systems (Chao, Kai, & Zhiwei, 2020), geographical information systems (Lv, Li, Lv, & Xiu, 2019), medical expert systems (Xie et al., 2018), healthcare expert systems (Wen, Zhang, Liu, & Lei, 2017), and air pollution monitoring systems (Lv, Liu, Wang, Liu, & Shang, 2020; Zhu, Pang, Chevallier, Wei, & Vo, 2019). In the past decades, various optimization algorithms have been proposed to solve different problems. According to different optimization strategies, optimization algorithms can be divided into two categories: exact algorithm and approximate algorithm (Xue, Zhu, & Wang, 2019; H. Zhang, Qiu, Cao, Abdel-Aty, & Xiong; X. Zhang et al., 2018; Z. Zhang, Liu, Zhou, & Chen, 2020). Exact algorithms have been widely studied and applied in many problems when we have a small sample, or the case's dimension is not high (Zhao & Li, 2020; Wu, Xiong, Cheng, & Xie,

☆ This document is the results of the research project funded by the Science and Technology Plan Project of Wenzhou, China (No. 2020G0055).

* Corresponding author.
E-mail addresses: lcy.yang@foxmail.com (C. Li); omama@wzu.edu.cn (J. Li); chenhuiling.jlu@gmail.com (H. Chen); as_heidari@ut.ac.ir, aliasgha@comp.nus.edu.sg, t0917038@u.nus.edu (A.A. Heidari)
1 https://aliasgharheidari.com

https://doi.org/10.1016/j.eswa.2020.114529
0957-4174/© 2020.

2020). Their design principles are generally based on dynamic programming, branch and bound and backtracking methods (Zeng, Liu, Wang, & Xiao, 2019); consequently, they usually can obtain the optimal solutions (Neapolitan & Naimipour, 2009; Neapolitan & Naimipour, 2009). However, they often cost much more time and space to solve the problem with large search space and complex structure by comparing with approximate algorithms (Chen, Qiao, Xu, Feng, & Cai, 2019) .

Approximation optimization algorithms, which can obtain a feasible solution in large-scale search space with linear time, have been received more and more attention in recent years. Traditional approximation algorithms, such as the greedy algorithm, gradient descent method, and newton method, can be easily implemented and have been successfully applied to many optimization problems (Neapolitan & Naimipour, 2009; Zhang, Qu, Li, Luo, & Xu, 2020). However, the efficiency of such traditional approximation algorithms depends on the mathematical nature of the problem itself, so they are often time-consuming and have poor scalability (Baykasoglu, 2012; Baykasoglu, 2012). Therefore, a more flexible and efficient algorithm is needed to overcome this deficiency (Zhu, Ma, Xie, Chevallier, & Wei, 2018). Based on this motivation, meta-heuristic algorithms that are simulating natural phenomena are receiving more and more attention nowadays (Beheshti, 2013). Although meta-heuristic algorithms are also a kind of approximation algorithms, they are different from the traditional approximation algorithms in that they do not need to consider the mathematical properties of the optimization problem in the process of problem-solving, and they have the characteristics of low complexity, high efficiency and high scalability (Cao et al., 2019; Cao, Fan, et al., 2020; Cao, Zhao, Gu, Ling, & Ma, 2020) .

Meta-heuristic algorithms, which are derived from the inspiration of different natural phenomena, can be mainly divided into three categories: evolution-based, physical-based, and swarm-based (Mirjalili & Lewis, 2016; Sun, Yang, Yang, & Xu, 2019). The evolution of species in nature mainly inspires Evolution-based algorithms. Through the continuous evolution of individuals, individuals with poor fitness are eliminated, and individuals with high fitness are constantly updating the solution. One of the most popular algorithms based on evolutionary methods is Genetic Algorithms (GA) (Goldberg & Holland, 1988; Goldberg & Holland, 1988). The GA simulates the natural selection of the Darwinian evolution and the mechanism of biological evolution in genetics. The GA assumes that each individual in the population is a chromosome, and new individuals are generated through continuous evolution. The new individuals retain old individuals' excellent genes and continue to evolve to achieve the goal of global optimization. Other popular evolution-based algorithms include Evolutionary Strategy (ES) (Rechenberg, 1978; Rechenberg, 1978), Genetic programming (GP) (Banzhaf & Koza, 2000; Banzhaf & Koza, 2000), Differential Evolution (DE) (Storn & Price, 1997; Sun, Xu, & Jiang, 2020), etc.

Physics-based algorithms mainly simulate physical phenomena in nature. Simulated Annealing (SA) (Hwang, 1988) is one of the most popular algorithms. The starting point of SA is based on the annealing process of solid matter in physics, which consists of three parts: heating process, isothermal process, and cooling process. In addition, there are lots of other physics-based algorithms such as Gravitational Search Algorithm (GSA) (Rashedi, Nezamabadi-pour, & Saryazdi, 2009), and Central Force Optimization (CFO) (Formato, 2008).

Swarm-based algorithms are inspired by the social behavior of biological groups in nature. These optimization algorithms are simulated through the complex social group behaviors such as competition and cooperation between groups to achieve the purpose of optimization. A typical representative algorithm based on the swarm method is Particle Swarm Optimization (PSO) algorithm (Kennedy & Eberhart, 2002). PSO mimics birds' collective behavior cooperatively searching food, and each group member constantly changes its search pattern by learning their own and other members' experiences. By comparing with the evolutionary-based and physics-based algorithms, Swarm-based algorithms have been proven to be very competitive and will play a much more important role in future optimization field (Mirjalili & Lewis, 2016). Table 1 lists popular swarm intelligence optimization algorithms from 1995 to 2019.

From Table 1, we can see that the HHO[1] is the latest swarm-based meta-heuristic algorithm which was proposed in 2019. Heidari et al., 2019 valid the effectiveness and robustness of HHO by comparing it with several other meta-heuristic algorithms on a limited number of numerical optimization functions and apply it to several real-world engineering problems. Meanwhile, due to its simplicity, broad applicability, and outstanding performance, HHO has been improved and applied to find viable solutions for many contemporary optimization problems. Jia, Lang, Oliva, Song, and Peng, 2019 proposed a dynamic HHO with a mutation mechanism, named DHHO/M, to solve the problem of satellite image segmentation. Bao, Jia, and Lang, 2019 proposed a hybrid algorithm named HHO-DE, which solves the color image multilevel thresholding segmentation. Chen, Jiao, Wang, Heidari, and Zhao, 2020 proposed an Enhanced HHO (EHHO) based on the chaotic drifts in the vicinity of the best solution and an opposition-based strategy, which can improve the diversity and exploration ability of the algorithm population, thereby effectively identifying unknown parameters in photovoltaic model components. Ridha, Heidari, Wang, and Chen, 2020 presents a Boosted HHO (BHHO) algorithm, which combines the flower pollination algorithm with the strong variation scheme of differential evolution, for the parameter identification of a single diode solar cell model. Wei et al., 2020 proposes an effective predictive model for intelligent entrepreneurial intentions based on an improved HHO optimized Kernel Extreme Learning Machine (KELM) to provide a rational reference for the development of talent development programs and guidance of students' entrepreneurial intentions. Chen et al., 2020 presents a powerful variant of the Harris Hawk optimization algorithm that integrates chaotic strategies, topological multiple group strategies, and differential evolutionary strategies. Although these works play a significant role in promoting the development of HHO, according to the No Free Lunch (NFL) theorem (Wolpert & Macready, 1997), we still need to improve HHO so that we can deal with other complex real-world optimization problems. Besides, when solving some complex optimization problems, the HHO algorithm still has the problem of slow convergence speed or easy to fall into local optimum, which seriously affects the optimization performance.

As an extension of meta-heuristic algorithms, hybrid-based approaches (Fu, Pace, Aloi, Yang, & Fortino, 2020) aims to combine the advantages of different meta-heuristic algorithms to improve the ability to deal with various complex optimization problems. At present, a hybrid-based algorithm is one of the most interesting trends in memetic algorithms (Kang, Li, & Ma, 2011). Therefore, in our work, we utilize the advantages of evolution-based and swarm-based algorithms to present a novel hybrid-based meta-heuristic algorithm (EESHHO) further to improve the optimization performance in some specific scenarios. The main contributions of this paper are as follows:

- A novel exploitation strategy, based on the principle of evolution-based meta-heuristic algorithms named Elite Evolutionary Strategy (EES), is proposed to improve specific swarm-based algorithms' optimization performance.
- By dynamically combining the EES and the original HHO, we propose a novel hybrid-based meta-heuristic algorithm named EESHHO and analyze its computational complexity.

[1] https://aliasgharheidari.com/HHO.html

Table 1
Efficient swarm-based algorithms.

Algorithms	Developers	Inspiration	Year
Particle swarm optimizer (PSO)	Kennedy and Eberhart (2002)	Bird flock	1995
Differential Evolution (DE)	Storn and Price (1997)	Vectors	1997
Ant colony optimization (ACO)	Dorigo et al. (2006)	Ant colony	2006
Artificial Bee Colony (ABC)	Karaboga and Basturk (2007)	Honey Bees	2007
Biogeography-based optimization (BBO)	Simon (2008)	Creation-Combination	2008
Cuckoo Search (CS)	Yang and Deb (2009)	Cuckoo	2009
Bacterial Foraging Optimization (BFO)	Das et al. (2009)	Bacterial life	2009
Fruit fly Optimization (FOA)	Pan (2012)	Fruit fly	2012
Harris Hawks Optimization (HHO) [b]	Heidari et al. (2019)	Harris hawks	2019
Slime mould algorithm (SMA) [a]	Li et al. (2020)	Slime mould	2020

[b] https://aliasgharheidari.com/HHO.html

- To verify the effectiveness and robustness of EESHHO, we evaluate it by solving 29 mathematical optimization problems and then apply it to the resource-constrained project scheduling problem and QoS-aware web service composition problem. Experimental results show that EESHHO is more competitive than other mainstream meta-heuristic algorithms.

The rest of this paper is organized as follows: Section 2 describes the proposed EESHHO. The results of EESHHO in solving different benchmark cases (numerical optimization functions) and two real-world case studies (Resource-constrained project scheduling problem and QoS-aware web service composition problem) are presented in Section 3. Finally, Section 4 summarizes the concluding observations and future work.

2. The proposed HHO-based method

This section presents a novel hybrid-based meta-heuristic algorithm named EESHHO for dealing with the problems described above. To be specific, we first propose the Elite Evolutionary Strategy (EES) based on evolution-based meta-heuristic algorithms' core principles and then combine it with HHO dynamically and intelligently. We present the structure of this variant along with the core equations of the basic HHO.

2.1. Elite evolution strategy

To overcome the HHO algorithm's shortcomings, we designed a new exploitation strategy (EES) for the HHO based on the evolution-based meta-heuristic algorithms' core principle. DE (Storn & Price, 1997) and GA (Goldberg & Holland, 1988) are the two famous evolution-based meta-heuristic algorithms with three same basic operations: selection, crossover, and mutation. However, GA uses binary coding to construct a chromosome, while DE has a more straightforward structure and no coding operation. GA and DE have been proven to be effective in various optimization problems, and show great potential in some specific complex optimization problems (Wang, Zeng, & Chen, 2015; Goldberg, 2008; Wang, Lee, & Ho, 2007). Based on the basic idea of DE and GA, EES is designed to extend the advantages of an evolutionary algorithm to HHO. This strategy includes two different methods: elite natural evolution and elite random mutation. It aims to overcome the HHO's slow convergence speed problem and easily fall into the local optimum.

The pseudo-code of EES is shown in Algorithm. 1and the details are described as follows.

Algorithm 1 Elite Evolution Strategy

1: **Input:** a chromosome X
2: Update the value of random number R between $(0,1)$
3: **If**($R \geqslant 0.5$) **then**
4: Use elite natural evolution(described in Section 2.1.1)
5: **Else If**($R<0.5$) **then**
6: Use elite random mutation(described in Section 2.1.2)
7: **End If**
8: **Output:** an evolved chromosome X'

2.1.1. Elite natural evolution

The core principles of Elite natural evolution are gene crossing and gene mutation. Gene crossing depends more on the excellent genes of multiple excellent chromosomes, and gene mutation mainly refers to a small range of local variation. Therefore, this method emphasizes local mining's ability, improving the convergence rate of the original HHO. The conceptual simulation of this method is shown in Fig. 1, and it consists of the following three steps:

- Elitist selection. Three elite solutions (the optimal solutions appearing during the evolution) are retained in the evolution of the algorithm, and they are named $E1, E2$, and $E3$, respectively. The relationship between the fitness values of the three elite solutions is: $f(E1) < f(E2) < f(E3)$ (minimum is optimal).
- Gene cross-recombination of elite chromosomes. Suppose that $E1, E2$, and $E3$ as a chromosome composed of multiple genes, each dimension representing a gene. As you can see in Fig. 1. First, we randomly select 50% genes of $E2$ and 50% genes of $E3$ to generate a new chromosome $N1$ and then randomly select $\lceil 100(1-sp) \rceil$ % genes of $E1$ and $\lceil 100sp \rceil$ % genes of $N1$ to cross to generate a new chromosome $N2$. Its mathematical description is shown below:

$$\begin{gathered} N1 = (50\% \otimes E2) \oplus (50\% \otimes E3) \\ N2 = \lceil 100(1-sp)\% \otimes E1 \rceil \oplus \lceil (100sp)\% \otimes N1 \rceil \end{gathered} \tag{1}$$

where the $\oplus$symbol indicates the cross-recombination operation of chromosomal genes. The $\otimes$ symbol indicates how many genes are randomly selected from each chromosome for cross-recombination. *sp* is a variable that controls the proportion of genes on the $E1$ chromosome.
- Gaussian local mutation. We use a Gaussian sequence ($\mu = 0, \sigma = 0.333$) to locally perturb the chromosome $N2$ to produce a new chromosome X' (refer to Fig. 1). The mathematical description is as follows:

$$X' = N2 + GS|N2 - X| \tag{2}$$

where GS is the sequence vector, which conforms to Gaussian probability distribution ($\mu = 0$ $\sigma = 0.333$). X is the chromosome waiting to be updated. $N2$ is the new chromosome calculated from Eq. (1). The $|\cdot|$ represents the operation of taking the absolute value. X' is the latest updated chromosome.

In this method, three elite chromosomes are selected for cross-recombination of genes, so that the new chromosome can inherit more excellent genes from different parents. However, the new chromosome contains only excellent genes and does not allow it to evolve further. Therefore, we introduce the Gaussian sequence to make local-wide mu-

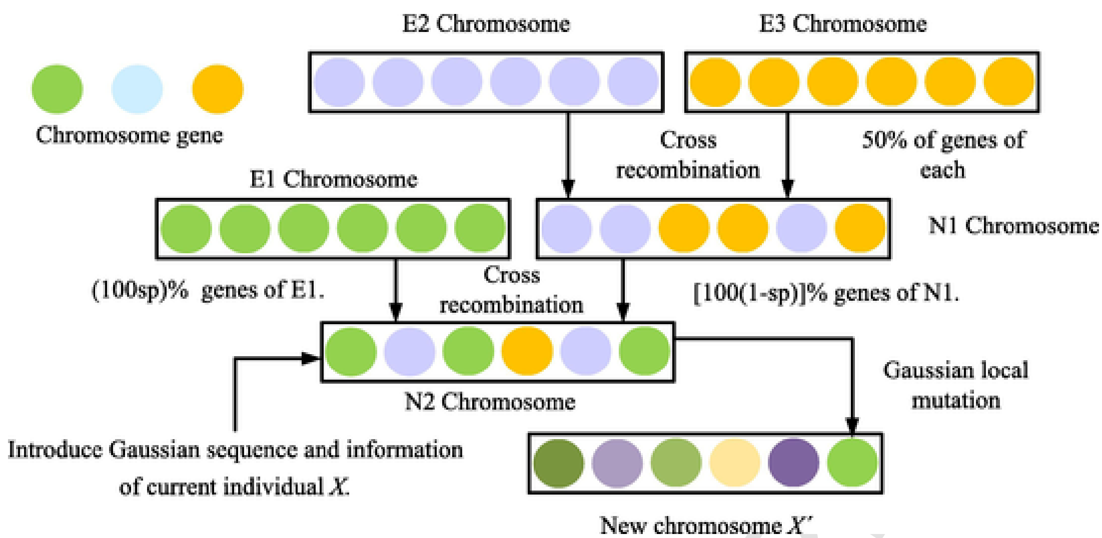


Fig. 1. Schematic illustration of the method of the elite natural evolution.

tations of all chromosome genes, thereby promoting the chromosome's evolutionary efficiency. Gauss probability distribution is a critical and widely used probability distribution in mathematics, physics, and engineering. Meanwhile, it has also been successfully applied in other meta-heuristic algorithms (Luo et al., 2018; Bäck & Schwefel, 1993; Xu et al., 2019). Fig. 2 shows a point set graph (connected by lines) obeying the Gaussian probability distribution with $\mu = 0, \sigma = 0.333$. The graph shows that this point set's distribution range is concentrated around 0, and the generated values are mostly between -0.5 and 0.5. Therefore, we use the distribution characteristics of the Gaussian sequence to provide a variation characteristic of local fluctuation for $N2$ in Eq. (2). In conclusion, this method absorbs the elite chromosome's excellent genes as the basis and introduces Gaussian mutation for local disturbance, emphasizing the exploitation characteristics of elite evolution strategy more.

Next, a numerical example is then introduced to briefly describe the process of elite evolution strategy. Suppose there are three elite individuals $E1 = [1, 4, 8, 10]$, $E2 = [2, 3, 6, 9]$ and $E3 = [1, 4, 9, 10]$ (integers are used for ease of understanding). Meanwhile, enter the individual $X = [2, 3, 1, 1]$ that needs to be updated. First, $E2$ and $E3$ generate $N1 = [2, 3, 9, 10]$ through a cross-genetic recombination operation (see Eq. (1) for details). Then, $N1$ and $E1$ perform a cross-combination operation (see Eq. (1) for details) to generate $N2 = [1, 4, 9, 10]$. Finally, the local mutation of Gaussian is used to generate a new individual X'

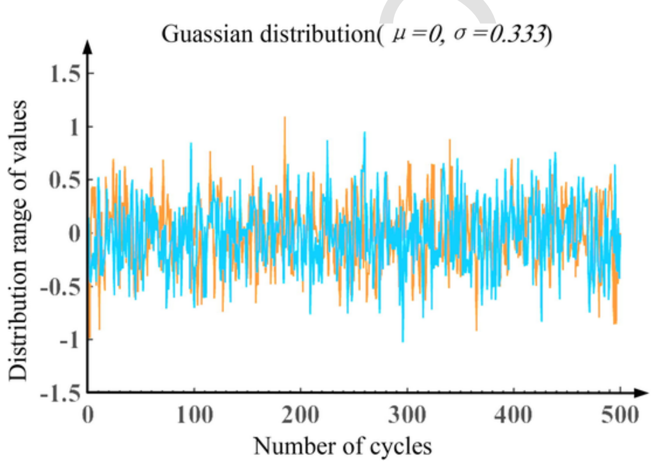


Fig. 2. Gaussian probability distribution ($\mu = 0, \sigma = 0.333$) during two runs and 500 cycles.

$[1.1, 3.9, 13, 11.8]$, which is obtained by Eq. (2). In detail, first a set of Gaussian sequence vectors $GS = [0.1, -0.1, 0.5, 0.2]$ is generated. then vector $S = [1, 1, 8, 9]$ is derived by subtracting individual vector $X = [2, 3, 1, 1]$ from vector $N2 = [1, 4, 9, 10]$ and taking the absolute value. The new individual vector $X' = [1.1, 3.9, 13, 11.8]$ is generated using Eq. (2).

2.1.2. Elite random mutation

The purpose of elite random mutation is to mutate some genes of elite chromosomes within the search scope and provide stronger exploration ability for the algorithm in the later stage of evolution to improve the ability to jump out of local optimum. The conceptual diagram of this method is shown in Fig. 3, and the detailed implementation steps are described below.

- Randomly mutated chromosome. We generate a brand new chromosome in the search space, which is unpredictable and should also maintain a certain relationship with the elite chromosome. Because this can reduce the disadvantages of non-convergence caused by excessive randomness while maintaining exploration. The mathematical formula we designed is described as follows:

$$R1 = CL + (GS_{num}) |CL - E1| \tag{3}$$

where CL represents the center position vector of the search space, $E1$ is the elite chromosome (described in Section 2.1.1). GS_{num} represents a number, which is generated by Gaussian probability distribution ($\mu = 0, \sigma = 1$). It can be seen from Fig. 4 that the only difference between this Gaussian probability distribution and Gaussian probability distribution ($\mu = 0, \sigma = 0.333$) mentioned in the previous section (see Section 2.1.1) is that the value is larger. Therefore, it can provide a wider range of fluctuations in Eq. (3). This random chromosome mutation is unpredictable under the influence of the Gauss number. However, we can find that for a single gene (a single dimension), it is more likely to occur between the central position and the elite chromosome $E1$ (position X in Fig.3), around the elite chromosome (position F), between the central position and the opposite position of $E1$ (position Y), and around the opposite position of $E1$ (position Z).
- Gene cross-recombination. $(100(1 - sp))\%$ of the current elite chromosome $E1$ genes and $(100sp)\%$ of randomly mutated chromosome $R1$ genes are randomly selected for cross recombination. The mathematical description is as follows:

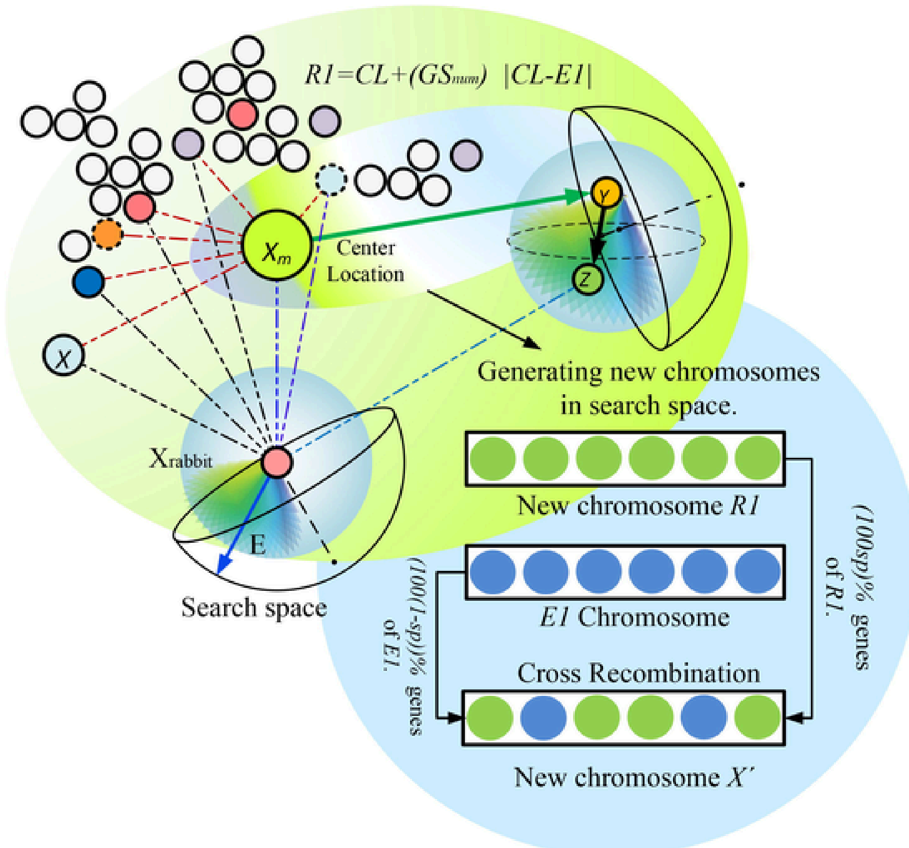


Fig. 3. Schematic illustration of the method of the elite random mutation.

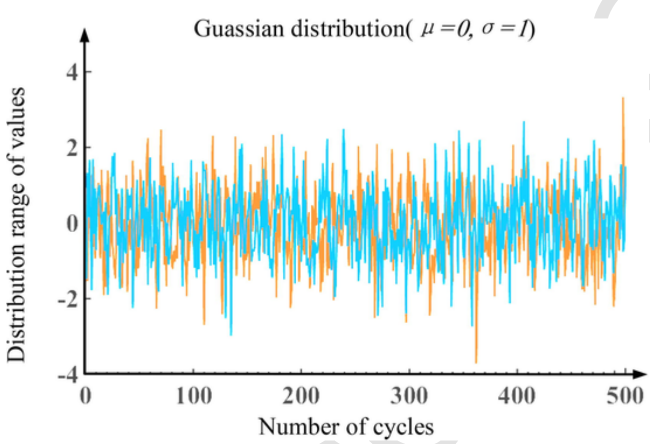


Fig. 4. Gaussian probability distribution ($\mu = 0, \sigma = 1$) during two runs and 500 cycles.

$$X' = \left[(100\,(1-sp))\,\%\otimes E1\right] \oplus \left[(100sp)\,\%\otimes R1\right] \tag{4}$$

where *sp* is a variable that controls the proportion of genes on the $E1$ chromosome, the ⊕symbol indicates the cross-recombination operation of chromosomal genes, The ⊗ symbol indicates how many genes are randomly selected from a chromosome for cross-recombination. $R1$ is obtained by Eq. (3).

The elite random mutation emphasizes more on the potential of exploration, but it will still retain some genes of the elite chromosomes, and the proportion of these excellent genes will continue to adjust with the evolution process to ensure timely convergence. A simple numerical example was used to illustrate the process of elite random mutation. First, we set the center of search space to $CL = [0, 0, 0, 0]$, which has an upper and lower boundary of 20 and -20, respectively. The elite chromosome vector $E1 = [1, 4, 8, 10]$, the Gaussian number $GS_{num} = 1.2$. Subsequently, we generated $R1 = [1.2, 4.8, 9.6, 12]$ by Eq. (3). Finally, we calculated the new individual position vector $X' = [1.2, 4, 9.6, 12]$ by Eq. (4).

2.1.3. Parameter setting

There is a critical control parameter of *sp* in the EES strategy. It controls the proportion of the best parental genes in the entire new chromosome and controls the entire EES transition between exploration and exploitation. When *sp* is larger, producted new chromosomes tend to contain more mutated genes, and conversely, the new chromosomes contain more genes from the best parent. To achieve an adaptive control of the convergence process of the EES, the parameter *sp* is designed, as show in Eq.(5).

$$sp = rand\,(-1,1) \times \left(1 - \frac{t}{T}\right). \tag{5}$$

where $rand\,(-1,1)$ indicates that random number between (-1, 1) and it changes into each evolution. T is the maximum number of evolutions, and t is the current number of evolutions. Fig .5 shows the distribution of points (connected by lines) in the algorithm's evolution. For example, *sp* may appear anywhere between 0 and 1 at the initial point of evolution, and it may appear anywhere between 0 and 0.5 in the middle stage of evolution. In short, with the end of the evolution, *sp* also tends to 0, thus ensuring the convergence of EES.

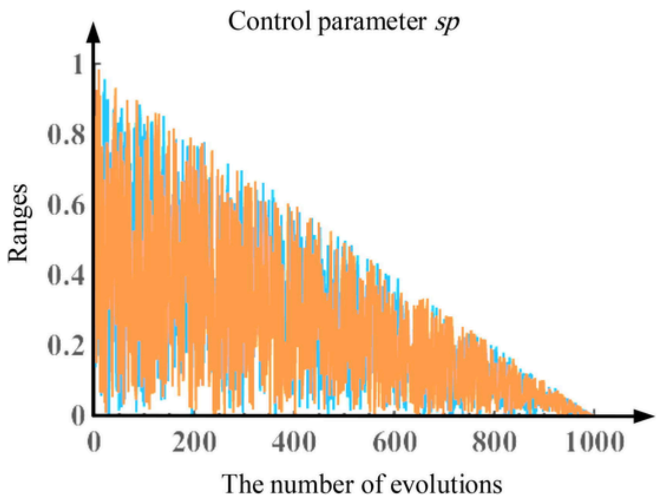


Fig. 5. Distribution of *sp* during two runs and 1000 evolutions.

2.2. Elite Evolution Strategy with Harris Hawks Optimization (EESHHO)

Harris hawks optimization is a new swarm-based stochastic optimizer inspired by the Harris hawks hunting prey (rabbit). It mainly seeks the optimal solution through two exploration strategies and four exploitation strategies. Meanwhile, it uses the parameter E to adaptively switch between the exploration and exploitation stages (Heidari et al., 2019).

In this section, we integrate the Elite Evolution Strategy (EES) into the exploitation phase of HHO to propose an enhanced meta-heuristic algorithm referred to as EESHHO. The improvement point of EESHHO compared with the original HHO is that the EES strategy replaces the two exploitation strategies of HHO, which are the hard besiege strategy with progressive rapid dives and the soft besiege strategy with progressive rapid dives (Heidari et al., 2019). Fig. 7 reveals the core mechanisms of the HHO, which is still preserved in the proposed EESHHO. Fig. 6 shows the flowchart of EESHHO. The detailed description is described below.

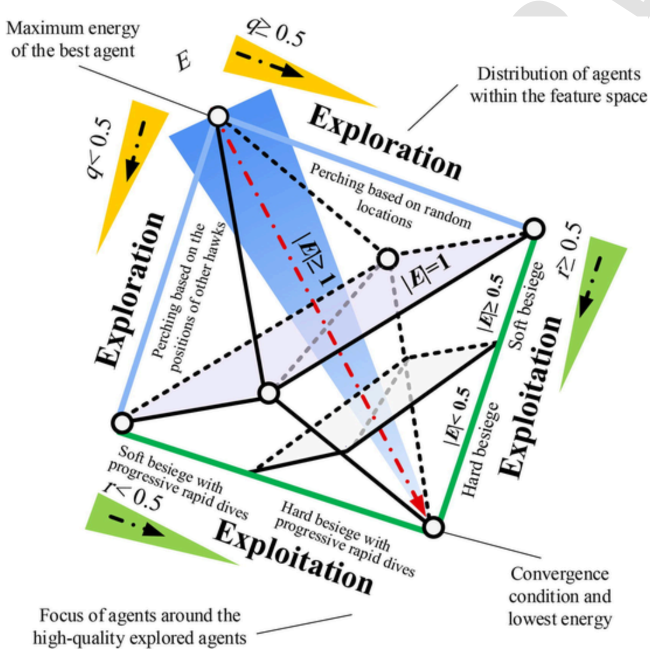


Fig. 7. The core logic of HHO and proposed EESHHO.

2.2.1. Transition between exploration and exploitation

The control parameter E is used to transfer from exploration to exploitation (see Fig. 6). The mathematical description of E is shown below:

$$E = 2E_0\left(1 - \frac{t}{T}\right). \tag{6}$$

where T is the maximum number of evolutions, and t represents the current number of evolutions. E_0 is a random number between (-1,1), and each agent updates this E_0 in each evolution. E can make the algorithm adjust adaptively between exploration and exploitation during the evolutionary process Heidari et al., 2019.

2.2.2. Exploration stage

From Fig. 6, The algorithm entered the exploration stage through the control of E, and Harris hawks use two different exploration strategies to update their position. These exploration strategies can make the Harris hawk explore more unknown areas, thereby increasing the possibility of finding a potential optimal solution. The mathematical description of the two exploration strategies is as follows:*First strategy* Harris hawk conducts random exploration based on information about the location of other family members. Its mathematical modeling is shown below:

$$X_{update} = X_{rand} - r_1\left|X_{rand} - 2r_2X_{current}\right|. \tag{7}$$

where X_{rand} is a randomly selected position vector of the Harris hawk from the Harris hawk population. $X_{current}$ is the current Harris hawk position vector that needs to be updated. $r1$ and $r2$ are random numbers between (0,1). X_{update} is the position vector after $X_{current}$ is updated by Eq. (7).*Second strategy* The Harris hawk uses the rabbit's location information and the scope of the search space for random exploration. The mathematical modeling expression for this behavior is shown below:

$$X_{update} = \left(X_{rabbit} - \overline{X_m}\right) - r_1\left(lb + r_2\left(ub - lb\right)\right) \tag{8}$$

where X_{rabbit} is the position vector of the rabbit(the best solution found so far). $\overline{X_m}$ represents the average position vector of the population. r_1 and r_2 are randomly generated number between 0 and 1. *ub* represents the upper bounds of the search space, and *lb* is the lower bounds. HHO randomly switches between the two exploration strategies with 50% probability, respectively (see Fig.7).

2.2.3. Exploitation stage

From Fig. 6, EESHHO adopts the EES (described in Section 2.1) and two exploitation strategies of the original HHO, which are hard besiege and soft besiege strategies. Suppose r represents the probability of the rabbit escaping from threatening situations. When $r \geqslant 0.5$ (the rabbit did not escape successfully) we adopted the original HHO exploitation strategy, and when r<0.5 (the rabbit successfully escaped) we adopted the Elite Evolution Strategy. Here is a detailed explanation.

The rabbit failed to escape($r \geqslant 0.5$) At this stage we use the two core strategies of soft besiege and hard besiege of the HHO. As follows:

- Hard besiege. When $r \geqslant 0.5$ and $|E|$ <0.5, the rabbit failed to escape from the encirclement and is so exhausted. Therefore, the Harris hawk swooped directly on the rabbit. The mathematical description is shown below:

  $$X_{update} = X_{rabbit} - E\left|X_{rabbit} - X_{current}\right|. \tag{9}$$

  where $X_{current}$ is the position vector of the current Harris hawk. X_{rabbit} is the position vector of the rabbit(the best solution found so far). E is the control parameter (see Section 2.2.1).

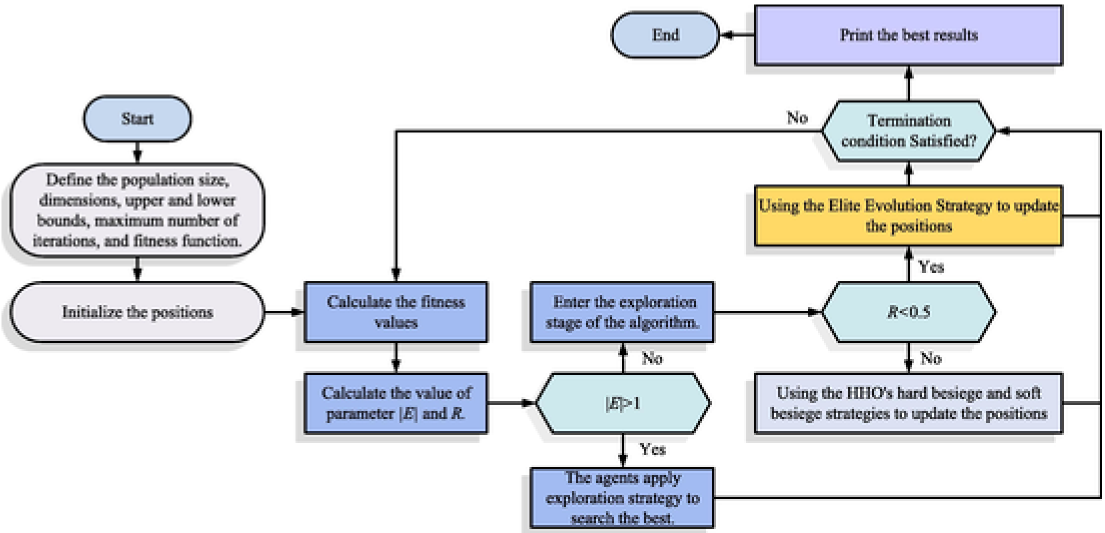


Fig. 6. The entire process of the EESHHO algorithm.

- Soft besiege. When $r \geqslant 0.5$ and $|E| \geq 0.5$, Although the rabbit failed to escape, it still had great energy. Hence, the Harris hawk gently surrounded the rabbit to make it more exhausted, then, the Harris hawk swooped on the rabbit. The mathematical description of the modeling is shown below:

$$X_{update} = \left(X_{rabbit} - X_{current}\right) - E\left|JX_{rabbit} - X_{current}\right|. \tag{10}$$

where J is a random number between $(0, 2)$ and needs to be updated after each evolution. J simulated the nature of rabbit motions.

The rabbit escaped successfully(r<0.5) At this stage, we adopted the EES to update the Harris hawks position. From reference Heidari et al., 2019, the original HHO used levy flight in the strategy to improve the algorithm's exploration ability at a later stage. Levy flight is widely used to enhance the population diversity of meta-heuristic algorithms (Emary, Zawbaa, & Sharawi, 2019). However, if the algorithm relies too much on levy flight, it may cause the algorithm to consume too many evolutionary times for random exploration, which will cause its convergence speed to be too slow. Therefore, the elite natural evolution method of EES is used to improve the convergence speed of EESHHO (detailed in Section 2.1.1). Besides, EES's elite random mutation is used to continue keeping the algorithm's later exploration capabilities (detailed in Section 2.1.2).

2.3. Pseudocode of HHO and computational complexity

The pseudocode of the proposed EESHHO algorithm is reported in Algorithm.2. Meanwhile, it can be noticed that EESHHO mainly includes three processes: initialization, Harris hawk's evolution update mechanism, and fitness evaluation. For EESHHO with a population size of N. The computational complexity of the initialization process is $O(N)$. The computational complexity of the fitness evaluation is $O(T \times N)$, where T is the number of evolutions of the algorithm. Harris hawk's evolution update mechanism is the main structure of the algorithm. Its computational complexity is $O(T \times N \times D)$, where D is the dimension of the optimization problem. Hence, the computational complexity of EESHHO is $O(N \times (T \times D + T + 1))$.

Algorithm 2 The pseudo-code of EESHHO

```
Initialize Harris hawks population X_i (i = 1, 2, ..., N).
Calculate the fitness value of each Harris hawk.
Get the current best solution X_best.
While (t ⩽Max number of evolutions)
 For (Updated each Harris hawk position (X_i))
 Update E, q, r and J
  If (|E| ⩾ 1) then     Exploration phase
   If(q ⩾ 0.5) then
    Use the first exploration strategy.    use Eq. (7).
   Else If(q≤0.5) then
    Use the second exploration strategy.    use Eq. (8).
   End If
  Else If (|E| <1) then    exploitation stage.
   If(r≥0.5) then
    If (|E| <0.5) then
     Use hard besiege.    use Eq. (9).
    If (|E| ⩾ 0.5) then
     Use soft besiege.    use Eq. (10).
    End If
   Else If(r<0.5) then
    Use Elite Evolution Strategy.  See Algorithm 1.
   End If
  End If
 End for
Calculate the fitness value of each Harris hawk.
Update X_best if a better solution is found.
t = t + 1
End while
Return X_best
```

3. Performance study

All experiments were performed on Windows Server 2012 R2 operating system, using Intel (R) Xeon (R) CPU E5-2660 v3 (2.60 GHz) and 16GB RAM. All algorithms were coded and run on MATLAB R2014b software. In this section, we verify the effectiveness of the novel algorithm (EESHHO) through the following experiments:

(1) EESHHO was compared with other popular meta-heuristic algorithms and advanced algorithms (variations of meta-heuristics) in solving 29 mathematical optimization problems to test its numerical efficiency.
(2) Performance comparison between EESHHO and the existing mainstream algorithms in solving the resource-constrained project schedul-

ing (Kim & Ellis, 2010) problem and QoS-aware web service composition (Li, Zheng, Chen, Song, & Chen, 2014) problem.

3.1. Mathematical optimization problems

In this section, 29 mathematical optimization problems were used as test cases, including 23 classical benchmark functions (Mirjalili & Lewis, 2016) and 6 composite functions mentioned in the CEC 2017 special session (Maharana, Kommadath, & Kotecha, 2017). Tables 2–4 summarize the 23 test cases reporting the cost function, class (unimodal, multimodal, and fixed-dimension multimodal), range of optimization variables, and the optimal value. Table 5 gives a brief overview of the remaining 6 composite functions, which are described in detail in reference Maharana et al., 2017. In Table 2–5, It is worth noting that "Dim" indicates the number of dimensions of the design variables, which was set to 30 except for the fixed-dimension multimodal function. Moreover, the population number of all comparison algorithms was set to 30, and the maximum fitness function evaluation number was set to 300,000.

EESHHO was compared with HHO (Heidari et al., 2019), Sine Cosine Algorithm (SCA) (Mirjalili, 2016), Slime Mould Algorithm (SMA)[2] (Li, Chen, Wang, Heidari, & Mirjalili, 2020), modified Weighted Superposition Attraction (mWSA) (Baykasoğlu & Akpinar, 2020), Whale Optimization Algorithm (WOA) (Mirjalili & Lewis, 2016), Grey Wolf Optimizer (GWO) (Mirjalili, Mirjalili, & Lewis, 2014), Particle Swarm Optimization with an Aging Leader (ALCPSO) (Chen et al., 2013), Fuzzy Self-Tuning Particle Swarm Optimization (FSTPSO) (Nobile et al., 2017), DHHO/M (Jia et al., 2019), Hybrid Harris Hawk Optimization Based on Differential Evolution (HHODE) (Birogul, 2019). A brief description of all the above comparison algorithms is as follows:

- EESHHO is a novel advanced meta-heuristic algorithm based on the HHO proposed in this research (see Section 2). The control parameters use an adaptive strategy that does not require additional pre-setting of other fixed parameter values.
- HHO is the original version of EESHHO. It has been compared with many classic meta-heuristic algorithms, including PSO, DE, GA, and so on (more details can be found in Heidari et al., 2019). Hence, these comparison algorithms will no longer include these algorithms. The HHO uses adaptive control parameters without pre-setting additional fixed parameter values.
- SCA is a swarm-based meta-heuristic algorithm that uses a mathematical model of sine and cosine to fluctuate outwards or towards the best solution. It is also a self-adjusting meta-heuristic algorithm, and there is no need to set parameter values in advance.
- SMA is a swarm-based meta-heuristic algorithm that was recently proposed in 2020. This algorithm is inspired by the inherent oscillation pattern of slime molds and proposes a positive and negative feedback mechanism with adaptive weights to explore and exploit the algorithm. It exhibits outstanding performance in different problem search spaces (more details can be found in Li et al., 2020).
- mWSA is a more efficient algorithm proposed by improving the heuristic algorithm Weighted Superposition Attraction (WSA) (Baykasoğlu & Akpinar, 2015) with an operator for the target point superposition determination process. The experimental results show that mWSA is more robust and has better performance than the original WSA in solving complex optimization problems (Baykasoğlu & Akpinar, 2020).
- WOA (Mirjalili & Lewis, 2016) is a viral meta-heuristic algorithm proposed in 2016, which mainly simulates humpback whales' social behavior in nature. The algorithm is comparable to our algorithms in terms of control complexity.
- GWO simulates the leadership and hunting mechanism of the gray wolf. Four different leadership levels provide strong exploration performance for GWO. As with these algorithms mentioned above, it is also unnecessary to set additional fixed parameter values in advance. However, it has a structural defect (Hu et al., 2020) that discovered recently by Niu, Niu, and Chang (2019).
- ALCPSO is an evolutionary version of PSO (Kennedy & Eberhart, 2002). The gradual aging of individuals inspires it in the population in nature. It adds the mechanism of aging leaders and challengers based on the PSO and improves the population's diversity while ensuring the speed of population convergence, thereby overcoming PSO's premature convergence. The parameter $T = 2$, which is used to control how long a challenger temporally leads the swarm.
- FSTPSO is an evolutionary version of PSO proposed in 2017. It is a self-adjusting PSO based on a fuzzy strategy, in which the behavior of each particle is dynamically and automatically adjusted during the optimization process.
- DHHO/M is an evolutionary variant of the HHO algorithm, which uses dynamic control parameter strategy and mutation operators to enhance the ability of the original HHO to jump out of the local optimum. There are mainly two control parameters set to $\alpha = 2.5$ and $SF = 0.5$.
- HHODE is an evolved version of HHO developed based on the mutation strategy of the DE (Storn & Price, 1997), and the advantages of DE and HHO are used in this algorithm, in which DE uses the mutation strategy of DE/current-to-best/2 (Birogul, 2019).

Table 2
Description of unimodal test functions $F1$-$F7$.

Function	Dim	Range	Optimum
$F_1(x) = \sum_{i=1}^{n} x_i^2$	30	[−100,100]	0
$F_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[−10,10]	0
$F_3(x) = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j \right)^2$	30	[−100,100]	0
$F_4(x) = \max_i \{ \lvert x_i \rvert, \quad 1 \leqslant i \leqslant n \}$	30	[−100,100]	0
$F_5(x) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2 \right]$	30	[−30,30]	0
$F_6(x) = \sum_{i=1}^{n} \left(\lfloor x_i + 0.5 \rfloor \right)^2$	30	[−100,100]	0
$F_7(x) = \sum_{i=1}^{n} i x_i^4 +$ random $[0,1)$	30	[−1.28,1.28]	0

All comparison algorithms run independently 30 times for each test case, in which each time starting from different populations randomly generated, and then the average result of these runs was obtained as the final result. Such a condition is to avoid biased and unfair comparisons (Lv & Qiao, 2020; Yang et al., 2019; Shi, Wang, Tang, & Zhong, 2020). Statistical results are reported in Tables 6–8, in which "*AVG*" represents the average of 30 independent runs per test case, and "*STD*" represents the standard deviation of 30 independent runs. The best results of each test case mark in bold, and these results are expressed in scientific notation, which only shows the decimal part's first four digits. Also, note that some values in the table that are the same as the best value display but do not have a bold block are because the decimal parts that are not displayed are not the same. Meanwhile, we evaluated the comprehensive performance of all algorithms based on all test cases, and the comprehensive performance metrics are denoted by "*CP*" . Among them, we performed statistical analysis of the experimental results using the Friedman test method (Friedman, 1937) and the Wilcoxon sign rank test (García, Fernández, Luengo, & Herrera, 2010) to ensure that our algorithm is statistically significant. "*ARV*" represents the comprehensive ranking obtained by the algorithm through statistical analysis and performance comparison based on all

[2] https://aliasgharheidari.com/SMA.html

Table 3
Description of multimodal test functions $F8$-$F13$.

Function	Dim	Range	Optimum
$F_8(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	30	[−500,500]	-418.9829×5
$F_9(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	30	[−5.12,5.12]	0
$F_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	30	[−32,32]	0
$F_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	[−600,600]	0
$F_{12}(x) = \frac{\pi}{n}\left\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2\left[1 + 10\sin^2(\pi y_{i+1})\right] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$	30	[−50,50]	0
$y_i = 1 + \frac{x_i + 1}{4} u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m x_i > a \\ 0 - a < x_i < a \\ k(-x_i - a)^m x_i < -a \end{cases}$			
$F_{13}(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2\left[1 + \sin^2(3\pi x_i + 1)\right] + (x_n - 1)^2\left[1 + \sin^2(2\pi x_n)\right]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30	[−50,50]	0

Table 4
Description of fixed-dimension multimodal test functions $F14$-$F23$.

Function	Dim	Range	Optimum
F14x=1500+∑j=1251j+∑i=12xi-aij6-1	2	[-65,65]	1
$F_{15}(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1\left(b_i^2 + b_i x_2\right)}{b_i^2 + b_i x_3 + x_4}\right]^2$	4	[-5,5]	0.00030
$F_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	[-5,5]	−1.0316
$F_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	2	[-5,5]	0.398
$F_{18}(x) = \left[1 + (x_1 + x_2 + 1)^2\left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right] \times \left[30 + (2x_1 - 3x_2)^2 \times \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right]$	2	[-2,2]	3
$F_{19}(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$	3	[1,3]	−3.86
$F_{20}(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$	6	[0,1]	−3.32
$F_{21}(x) = -\sum_{i=1}^{5}\left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4	[0,10]	−10.1532
$F_{22}(x) = -\sum_{i=1}^{7}\left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4	[0,10]	−10.4028
$F_{23}(x) = -\sum_{i=1}^{10}\left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4	[0,10]	−10.5363

test cases, and "$+/=/-$" is used to display the details of test results (obtained by statistical analysis), among which "$+$" indicates the number of test cases where EESHHO performs better than another comparison algorithm. Similarly, "$-$"/"$=$" indicate the number of test cases that the performance of EESHHO is lower than/equal to the other one.

3.1.1. Evaluation of exploitation and exploration capabilities

These test cases ($F1$-$F7$) in Table 2 are unimodal functions, which contain only one global optimal solution. The investigated meta-heuristic algorithms do not worry about the risk of falling into local optimum in these test functions but only focus on their exploitation capability performance. From the "*ARV*" index in Table 6, we can see that EESHHO has achieved the first comprehensive ranking, which is very competitive compared with other comparative algorithms.

In particular, from the comparison results of EESHHO and HHO, they have obtained the same results on the $F1, F2, F3, F4$ test functions, and are optimal. Although HHO shows better results than EESHHO on the $F5$ test function; EESHHO's results are not bad either. Also, EESHHO has better performance than HHO on the $F6$ and $F7$ test functions. The results demonstrated that our algorithm not only retains the original HHO exploitation performance but also further improves it. This is because the core strategy of HHO is retained in the process of developing EESHHO (see Algorithm 2), so its exploitation ability is not weakened. Meanwhile, the Elite Evolution Strategy (EES) retains the most excellent genes and provides Gaussian local mutation (see Section 2.1.1), which significantly enhances the local exploitation ability of EESHHO.

Table 5
Summary of composition functions $F24$-$F29$.

ID(CEC2017-ID)	Properties	Dim	Range	Optimum
F_{24}(C22)	MM,NS,A,DO	30	[−100,100]	2200
F_{25}(C23)	MM,NS,A,DO	30	[−100,100]	2300
F_{26}(C25)	MM,NS,A,DO	30	[−100,100]	2500
F_{27}(C28)	MM,NS,A,DO	30	[−100,100]	2800
F_{28}(C29)	MM,NS,A,DO,DS	30	[−100,100]	2900
F_{29}(C30)	MM,NS,A,DO,DS	30	[−100,100]	3000

* MM:Multi-modal, NS:Non-separable, NS:Non-separable, A:Asymmetrical.
* DO:Different properties around different local optima.
* DS:Different properties for different variables sub components.

Table 3 shows the multimodal functions, and Table 4 shows the fixed-dimension multimodal functions. They are different from unimodal functions in that they contain a large number of locally optimal solutions and only one global optimal solution. Hence, these test cases ($F8$-$F23$) can be used to test the algorithm's exploration ability. It should be pointed out that the fixed-dimension multimodal functions are different from multimodal functions because their dimension variable (see "Dim" index in the Table 4) cannot be changed, but they provide different search space and function mathematical properties from multimodal function. Table 7 reports the experimental results of the comparison algorithm on multimode functions ($F8$-$F23$). It can be seen from the comprehensive index "*ARV*" that EESHHO has achieved the first achievement, followed by ALCPSO, SMA, DHHO/M, HHODE, HHO, WOA, GWO, FSTPSO, mWSA, SCA. Moreover, it can be seen from the "+/=/-" indicators that EESHHO outperforms HHO, SCA, SMA, mWSA, WOA, GWO, FSTPSO, DHHO/M, and HHODE in at least 11 test functions out of 16 test functions. Although EESHHO outperformed ALCPSO only in 8 test functions, it did not appear inferior to ALCPSO. These results demonstrate that the EESHHO has a good exploration ability, not only better than the original HHO but also better than other comparison algorithms. This is mainly due to the addition of elite random mutation mechanism in EES (see Section 2.1.2), which provides more opportunities for EESHHO to explore, leading this algorithm towards global optimality.

3.1.2. Balanced performance evaluation between exploration and exploitation

Table 5 shows 6 composite functions. Like multimode functions, they have a large number of local optimal values and only one global optimal value. However, optimizing a composite function is a more challenging task than a multimode function. It requires the algorithm's exploration and exploitation ability to be strong enough and requires a proper balance between exploration and exploitation to allow local optimization to be avoided.

The optimization results of all comparison algorithms on these composite test functions ($F24$-$F29$) are reported in Table 8. From "+/=/-" index, it can be clearly seen that the optimization results obtained by EESHHO on all 6 composite test functions are better than HHO, SCA, mWSA, WOA, FSTPSO, DHHO/M, HHODE. Although GWO has achieved better results than EESHHO on the $F25$ and $F28$ test functions, EESHHO is still better than GWO in terms of comprehensive indicators. Also, EESHHO is better than or equal to ALCPSO in all test cases except the $F25$ test function. These experimental results prove that EESHHO can well balance the exploration and exploitation stages, which is much stronger than HHO's performance in this regard. Such ability derives from the adaptive strategy for controlling the proportion of EES's outstanding genes (see Section 2.1.3).

3.1.3. Evaluation of convergence performance

Fig. 8 shows the convergence curve of EESHHO and other comparison algorithms in some of the test functions, where "*Average Best-so-far*" represents the average of the optimal values obtained in each evaluation evolution of 30 runs so far, and "*FES*" indicates the number of fitness evaluation (the maximum fitness evaluation is 3×10^5). It can be

Table 6
Comparison of optimization results obtained for the unimodal benchmark functions(F_1-F_7).

Algorithms	$F1$		$F2$		$F3$		$F4$	
	AVG	*STD*	*AVG*	*STD*	*AVG*	*STD*	*AVG*	*STD*
EESHHO	**0.0000E+00**	0.0000E+00	**0.0000E+00**	0.0000E+00	**0.0000E+00**	0.0000E+00	**0.0000E+00**	0.0000E+00
HHO	0.0000E+00	0.0000E+00	**0.0000E+00**	0.0000E+00	**0.0000E+00**	0.0000E+00	**0.0000E+00**	0.0000E+00
SCA	1.5092E-54	8.1565E-54	1.2432E-57	6.1761E-57	8.9022E-01	2.0635E+00	3.2585E-03	9.4607E-03
SMA	**0.0000E+00**	0.0000E+00	**0.0000E+00**	0.0000E+00	**0.0000E+00**	0.0000E+00	**0.0000E+00**	0.0000E+00
mWSA	7.6719E-133	4.0244E-132	5.5254E-67	1.0743E-66	7.9881E-132	2.6165E-131	7.8409E-68	1.4439E-67
WOA	**0.0000E+00**	0.0000E+00	**0.0000E+00**	0.0000E+00	2.4127E+01	7.3936E+01	3.9687E+00	1.0534E+01
GWO	**0.0000E+00**	0.0000E+00	**0.0000E+00**	0.0000E+00	2.9589E-180	0.0000E+00	1.3786E-152	3.9903E-152
ALCPSO	9.5653E-50	5.2391E-49	2.9696E-05	1.5774E-04	3.0887E-11	7.5484E-11	3.8977E-05	5.3755E-05
FSTPSO	4.4751E+03	1.2849E+03	2.2936E+01	1.1108E+01	9.4325E+03	5.0963E+03	2.6900E+01	4.2929E+00
DHHO/M	**0.0000E+00**	0.0000E+00	1.2077E-257	0.0000E+00	**0.0000E+00**	0.0000E+00	1.2674E-233	0.0000E+00
HHODE	**0.0000E+00**	0.0000E+00	**0.0000E+00**	0.0000E+00	**0.0000E+00**	0.0000E+00	**0.0000E+00**	0.0000E+00
	$F5$		$F6$		$F7$		*CP*	
	AVG	*STD*	*AVG*	*STD*	*AVG*	*STD*	+/=/-	*ARV*
EESHHO	2.6822E-04	6.9776E-04	5.1639E-19	6.6998E-19	1.2192E-05	1.4710E-05	N/A	**1**
HHO	8.7569E-05	8.8091E-05	1.2975E-06	2.0710E-06	2.3470E-05	2.6968E-05	2/4/1	3
SCA	2.7373E+01	7.1361E-01	3.5530E+00	2.8591E-01	1.3912E-03	1.1244E-03	7/0/0	9
SMA	1.8703E-03	1.4628E-03	9.5592E-06	3.1015E-06	1.1143E-05	1.1525E-05	2/4/1	3
mWSA	2.8956E+01	1.9055E-02	6.4653E+00	4.8906E-01	**3.1071E-06**	4.2301E-06	6/0/1	7
WOA	2.4260E+01	2.6308E-01	5.6667E-06	1.8474E-06	1.5271E-04	1.8244E-04	5/2/0	6
GWO	2.6388E+01	6.8458E-01	4.3542E-01	3.3413E-01	6.0386E-05	4.4058E-05	5/2/0	5
ALCPSO	4.5893E+01	3.2837E+01	**2.6758E-31**	5.1946E-31	9.3594E-02	3.6838E-02	6/0/1	8
FSTPSO	9.9541E+05	5.8474E+05	3.6523E+03	1.1366E+03	2.9703E-01	1.6947E-01	7/0/0	10
DHHO/M	**2.8167E-05**	4.8775E-05	7.5434E-07	9.4845E-07	2.2963E-05	2.3856E-05	4/2/1	4
HHODE	8.2239E-05	1.2719E-04	8.7012E-07	1.6085E-06	1.2446E-05	1.5174E-05	2/4/1	2

Table 7
Comparison of optimization results obtained for the multimodal(F_8-F_{13}), and fixed-dimension multimodal benchmark functions(F_{14}-F_{23}).

Algorithms	*F*8		*F*9		*F*10		*F*11	
	AVG	*STD*	*AVG*	*STD*	*AVG*	*STD*	*AVG*	*STD*
EESHHO	**−1.2569E+04**	1.9877E-11	**0.0000E+00**	0.0000E+00	**8.8818E-16**	0.0000E+00	**0.0000E+00**	0.0000E+00
HHO	−1.2569E+04	1.1229E-02	**0.0000E+00**	0.0000E+00	**8.8818E-16**	0.0000E+00	**0.0000E+00**	0.0000E+00
SCA	−4.4281E+03	2.4368E+02	4.1213E+00	1.3415E+01	1.1248E+01	9.0603E+00	1.4026E-15	7.6823E-15
SMA	−1.2569E+04	3.2611E-04	**0.0000E+00**	0.0000E+00	**8.8818E-16**	0.0000E+00	**0.0000E+00**	0.0000E+00
mWSA	−3.1796E+03	6.0164E+02	**0.0000E+00**	0.0000E+00	**8.8818E-16**	0.0000E+00	**0.0000E+00**	0.0000E+00
WOA	−1.2470E+04	3.0222E+02	**0.0000E+00**	0.0000E+00	2.7830E-15	1.8027E-15	2.1984E-03	1.2041E-02
GWO	−6.3168E+03	7.6188E+02	**0.0000E+00**	0.0000E+00	7.6383E-15	1.0840E-15	**0.0000E+00**	0.0000E+00
ALCPSO	−1.1533E+04	3.4459E+02	2.1889E+01	7.3174E+00	1.3944E+00	7.4939E-01	1.4968E-02	2.0150E-02
FSTPSO	−5.1212E+03	6.7136E+02	1.6505E+02	2.9367E+01	1.2538E+01	1.1231E+00	3.7419E+01	1.3005E+01
DHHO/M	−1.2569E+04	1.5606E-03	**0.0000E+00**	0.0000E+00	**8.8818E-16**	0.0000E+00	**0.0000E+00**	0.0000E+00
HHODE	−1.2569E+04	1.1471E-02	**0.0000E+00**	0.0000E+00	**8.8818E-16**	0.0000E+00	**0.0000E+00**	0.0000E+00
	*F*12		*F*13		*F*14		*F*15	
	AVG	*STD*	*AVG*	*STD*	*AVG*	*STD*	*AVG*	*STD*
EESHHO	**1.6843E-20**	2.9413E-20	**4.3351E-19**	7.2061E-19	**9.9800E-01**	2.2584E-16	**3.0749E-04**	1.6949E-14
HHO	5.4964E-08	7.6732E-08	5.5296E-07	6.5750E-07	9.9800E-01	2.1208E-11	3.1145E-04	4.3398E-06
SCA	3.2797E-01	8.5057E-02	1.9925E+00	1.3249E-01	9.9800E-01	8.4157E-07	5.6506E-04	4.0557E-04
SMA	9.4534E-06	1.0297E-05	5.5624E-06	3.8945E-06	9.9800E-01	5.3044E-16	3.1826E-04	5.5152E-05
mWSA	9.2760E-01	1.6722E-01	2.9888E+00	3.5455E-02	3.2373E+00	2.6432E+00	1.9582E-03	1.4055E-03
WOA	1.1851E-04	6.4354E-04	3.8903E-04	2.0062E-03	1.1964E+00	6.0541E-01	3.8352E-04	2.3192E-04
GWO	3.4374E-02	1.8528E-02	4.1542E-01	1.9799E-01	5.5553E+00	4.9363E+00	3.6501E-03	7.6022E-03
ALCPSO	3.7020E-02	6.8999E-02	6.0051E-03	7.5697E-03	**9.9800E-01**	1.1662E-16	3.9909E-04	2.7951E-04
FSTPSO	4.2431E+04	8.6941E+04	9.7259E+05	8.1306E+05	5.9357E+00	3.6198E+00	6.6875E-03	1.2276E-02
DHHO/M	1.6448E-08	2.1878E-08	2.0629E-07	2.1369E-07	9.9800E-01	1.2384E-12	3.1057E-04	3.9860E-06
HHODE	6.9854E-08	9.6700E-08	8.9989E-07	1.7802E-06	9.9800E-01	1.3064E-11	3.1043E-04	3.5949E-06
	*F*16		*F*17		*F*18		*F*19	
	AVG	*STD*	*AVG*	*STD*	*AVG*	*STD*	*AVG*	*STD*
EESHHO	**−1.0316E+00**	5.3761E-16	**3.9789E-01**	0.0000E+00	3.0000E+00	1.8217E-15	**−3.8628E+00**	2.3744E-15
HHO	−1.0316E+00	2.2504E-14	3.9789E-01	5.8942E-10	3.0000E+00	1.7621E-11	−3.8627E+00	9.6938E-05
SCA	−1.0316E+00	2.5854E-06	3.9796E-01	5.4215E-05	3.0000E+00	2.4497E-07	−3.8556E+00	2.3401E-03
SMA	−1.0316E+00	1.5288E-14	3.9789E-01	8.5497E-13	3.0000E+00	1.1275E-14	−3.8628E+00	7.0951E-11
mWSA	−1.0218E+00	1.4005E-02	4.2301E-01	8.4634E-02	3.2218E+00	9.2450E-01	−3.7637E+00	1.1019E-01
WOA	−1.0316E+00	5.3542E-15	3.9789E-01	2.1053E-10	3.0000E+00	2.4841E-08	−3.8620E+00	2.4023E-03
GWO	−1.0316E+00	2.7309E-11	3.9789E-01	9.3782E-10	3.0000E+00	1.2745E-07	−3.8625E+00	1.4528E-03
ALCPSO	**−1.0316E+00**	5.9752E-16	**3.9789E-01**	0.0000E+00	3.0000E+00	2.1630E-15	**−3.8628E+00**	2.6117E-15
FSTPSO	**−1.0316E+00**	6.7752E-16	**3.9789E-01**	0.0000E+00	**3.0000E+00**	1.3374E-15	**−3.8628E+00**	2.7101E-15
DHHO/M	−1.0316E+00	6.9853E-16	3.9789E-01	2.0299E-11	3.0000E+00	8.3833E-14	−3.8628E+00	6.0133E-06
HHODE	−1.0316E+00	1.6483E-15	3.9789E-01	2.0292E-11	3.0000E+00	3.5140E-12	−3.8628E+00	5.1712E-06
	*F*20		*F*21		*F*22		*F*23	
	AVG	*STD*	*AVG*	*STD*	*AVG*	*STD*	*AVG*	*STD*
EESHHO	−3.2824E+00	5.7005E-02	**−1.0153E+01**	5.4001E-15	**−1.0403E+01**	8.7273E−16	**−1.0536E+01**	2.4240E−15
HHO	−3.2304E+00	6.6832E−02	−5.3940E+00	1.2895E+00	−5.2648E+00	9.7041E−01	−5.4889E+00	1.3718E+00
SCA	−2.9275E+00	2.5921E−01	−3.2554E+00	2.8916E+00	−4.9723E+00	2.2361E+00	−4.8124E+00	2.7023E+00
SMA	−3.2229E+00	4.5066E−02	−1.0153E+01	1.1359E−06	−1.0403E+01	1.4123E−06	−1.0536E+01	1.3696E−06
mWSA	−2.4044E+00	3.9204E−01	−4.3150E+00	1.1682E+00	−3.9567E+00	5.8156E−01	−3.9361E+00	7.2667E−01
WOA	−3.2638E+00	7.6506E−02	−1.0153E+01	1.3456E−06	−1.0403E+01	5.7089E−07	−1.0536E+01	1.2703E−06

Algorithms	*F*8			*F*9			*F*10			*F*11	
	AVG	*STD*		*AVG*	*STD*		*AVG*	*STD*		*AVG*	*STD*
GWO	−3.2610E+00	6.8092E−02		−9.6449E+00	1.5509E+00		−1.0403E+01	8.7629E−07		−1.0358E+01	9.7874E−01
ALCPSO	−3.2625E+00	6.0463E−02		−9.1208E+00	2.0513E+00		−9.3298E+00	2.1419E+00		−9.6414E+00	2.0356E+00
FSTPSO	**−3.2938E+00**	5.1862E−02		−5.1435E+00	3.2123E+00		−5.7639E+00	3.4470E+00		−4.7915E+00	3.3333E+00
DHHO/M	−3.2621E+00	6.9364E−02		−1.0153E+01	1.1686E−04		−1.0403E+01	1.7177E−04		−1.0536E+01	2.2841E−04
HHODE	−3.2224E+00	8.1070E−02		−1.0152E+01	9.6067E−04		−1.0402E+01	1.4491E−03		−1.0535E+01	1.2308E−03
CP	EESHHO	HHO	SCA	SMA	mWSA	WOA	GWO	ALCPSO	FSTPSO	DHHO/M	HHODE
+/=/-	N/A	13/3/0	14/2/0	13/3/0	13/3/0	14/2/0	14/2/0	8/8/0	11/5/0	12/4/0	13/3/0
ARV	**1**	6	11	3	10	7	8	2	9	4	5

Table 8
Comparison of optimization results obtained for the composite benchmark functions (F_{24}-F_{29}).

Algorithms	$F24$		$F25$		$F26$		$F27$	
	AVG	*STD*	*AVG*	*STD*	*AVG*	*STD*	*AVG*	*STD*
EESHHO	**4516.079175**	2165.49847	2838.705634	45.51988901	2895.122052	11.42102482	**3195.162893**	43.06245093
HHO	6542.02607	1527.540171	3177.761774	125.4992381	2909.68814	16.93351827	3267.689571	36.70142322
SCA	8100.954706	2509.770232	2985.004547	22.7433969	3219.793884	58.08508899	3850.586836	182.5784693
SMA	5681.613474	850.3205581	**2741.043834**	26.8713464	**2887.248306**	2.365572384	3222.768223	50.55585564
mWSA	9846.343147	560.9177017	3852.673848	191.0922944	5963.026046	652.1478017	8047.23587	783.9281287
WOA	6860.749094	2235.076952	3052.880703	99.66516147	2945.509013	29.89842298	3297.415443	28.11785503
GWO	5080.418142	1232.642694	2745.08302	27.54470558	2986.731553	60.08833189	3422.718283	109.5304787
ALCPSO	5353.483612	1562.458697	2795.896223	51.11463433	2902.006762	20.66201859	3238.409277	28.45413434
FSTPSO	7638.661452	1467.646203	3318.791849	176.1770835	3898.324787	310.0506547	4777.0864	453.2765507
DHHOM	6503.74572	2016.337093	3086.263999	92.42761074	2912.85731	17.46978447	3259.06818	39.37503833
HHODE	5982.145363	2163.102578	3116.070845	123.9470167	2905.371407	16.57750524	3253.588993	21.43974962
	$F28$		$F29$		CP			
	AVG	*STD*	*AVG*	*STD*	+/=/-		*ARV*	
EESHHO	3916.568663	262.7993431	**12462.8787**	9355.759021	N/A		**1**	
HHO	4411.160613	273.1791471	1765583.539	1015822.794	6//0//0		7	
SCA	4644.560624	216.0572195	73186436.47	20429799.84	6//0//0		9	
SMA	3812.790301	163.3519005	16681.97012	4707.529439	3//1//2		2	
mWSA	11421.93258	7371.837512	2808946039	1375991287	6//0//0		11	
WOA	4837.291147	484.1190401	10203286.38	5568561.77	6//0//0		8	
GWO	**3728.697389**	149.0688107	6098613.54	6952598.975	3//1//2		4	
ALCPSO	3930.728102	212.0652639	16277.40501	6157.888819	2//3//1		3	
FSTPSO	5290.637653	402.7803798	29393339.53	34187269.57	6//0//0		10	
DHHOM	4327.235406	344.9879671	1605689.543	714353.3557	6//0//0		6	
HHODE	4296.751484	352.5739514	1531646.441	830060.8559	6//0//0		5	

seen from the figure that EESHHO has strong competitiveness compared with other comparison algorithms.

As shown in Fig. 8, EESHHO shows three different convergence curve states when optimizing these test functions. First, EESHHO can quickly converge to the optimal solution, which is shown in $F1, F3$, and $F4$ test functions. This is because the EES EESHHO algorithm can make full use of the information of many excellent solutions. Once the algorithm finds the region's evolutionary direction containing the optimal solution, it can quickly locate its location. Secondly, compared with other algorithms, EESHHO can achieve higher convergence accuracy in fewer evolution times, which is shown in $F10, F14, F21, F23$, and $F24$ test functions (including multimode functions and composite functions). This may be that EESHHO has reached a proper balance in the exploration and exploitation strategies in the process of optimizing these functions; after the exploration strategy is quickly located in the most promising area, the exploitation strategy is adopted to conduct a local search for the area promptly. This is due to the algorithm's full adaptive adjustment strategy. Thirdly, EESHHO gradually converges to the optimal solution at the later stage of evaluation, which is shown in $F12, F13, F27$, and $F29$ test functions. This may be because EESHHO did not find a suitable solution at the beginning of the evaluation and fell into the local optimum, but EESHHO is still evolving and converging towards the global optimum in the final stage algorithm convergence. This is due to the gene random mutation mechanism in EESHHO, which allows the algorithm to jump out of the local optimum at the later evolution stage.

In conclusion, the comprehensive performance of EESHHO is the best among all the comparison algorithms. First of all, EESHHO has a high exploration ability because it integrates the exploration performance of original HHO (see Section 2.2.2) and elite random mutation mechanism in EES (see Section 2.1.2). It is worth noting that EESHHO mainly adopts EES in the middle and later stages of evolution, which provides the opportunity to jump out of local optimum for the later evolution of the algorithm. Secondly, EESHHO shows strong exploitation ability because it further adopts the elite evolution mechanism of EES (see Section 2.1.1) in the exploitation strategy of the original HHO. Meanwhile, EESHHO keeps the adaptive adjustment strategy of the original HHO to maintain a proper balance between exploration and exploitation. The experimental results prove that EESHHO shows high convergence speed and local optima avoidance. In the next section, verify the performance of EESHHO in more challenging real-world problems (resource-constrained project scheduling and QoS-aware web service composition).

3.2. EESHHO for resource-constrained project scheduling problem

Resource constrained project scheduling problem (RCPSP) is a classical combinatorial optimization problem and an NP-hard problem (Kim & Ellis, 2010; Huang & Yang, 2019). The RCPSP is often considered one of the benchmarks for testing discrete search optimization algorithms (Baykasoğlu & Şenol, 2019). In the RCPSP, a project containing N activities can be represented as $V = \{1, 2, \ldots, N\}$, where 1 and N denote the two virtual activities of project start and end, respectively. Meanwhile, all activities cannot be interrupted after they have started to be executed. K renewable resources are required for the implementation of the project, and the amount of each resource available in each time interval is denoted as $R_k, k = 1, 2, \ldots, K$. Also, each activity has a duration that it needs to run for, during which time the activity cannot be terminated. For example, the duration of activity V_i is expressed as d_i, and the demand for the *k-th* resource is $r_{i,k}$. Virtual activities 1 and N both have durations and resource requirements of 0. In addition to having resource quantity constraints, there are also precedence constraints between activities. For each activity, $i \in V$, there is a set of preceding and succeeding activities P_i and S_i. Activity V_i must not begin execution until all of its predecessor activities $j \in P_i, j = 1, 2, \ldots, |P_i|$ are completed. Fig. 9.(a) shows the precedence constraint relationship for a project containing 7 activities. The number above each activity represents the duration, the number below indicates the

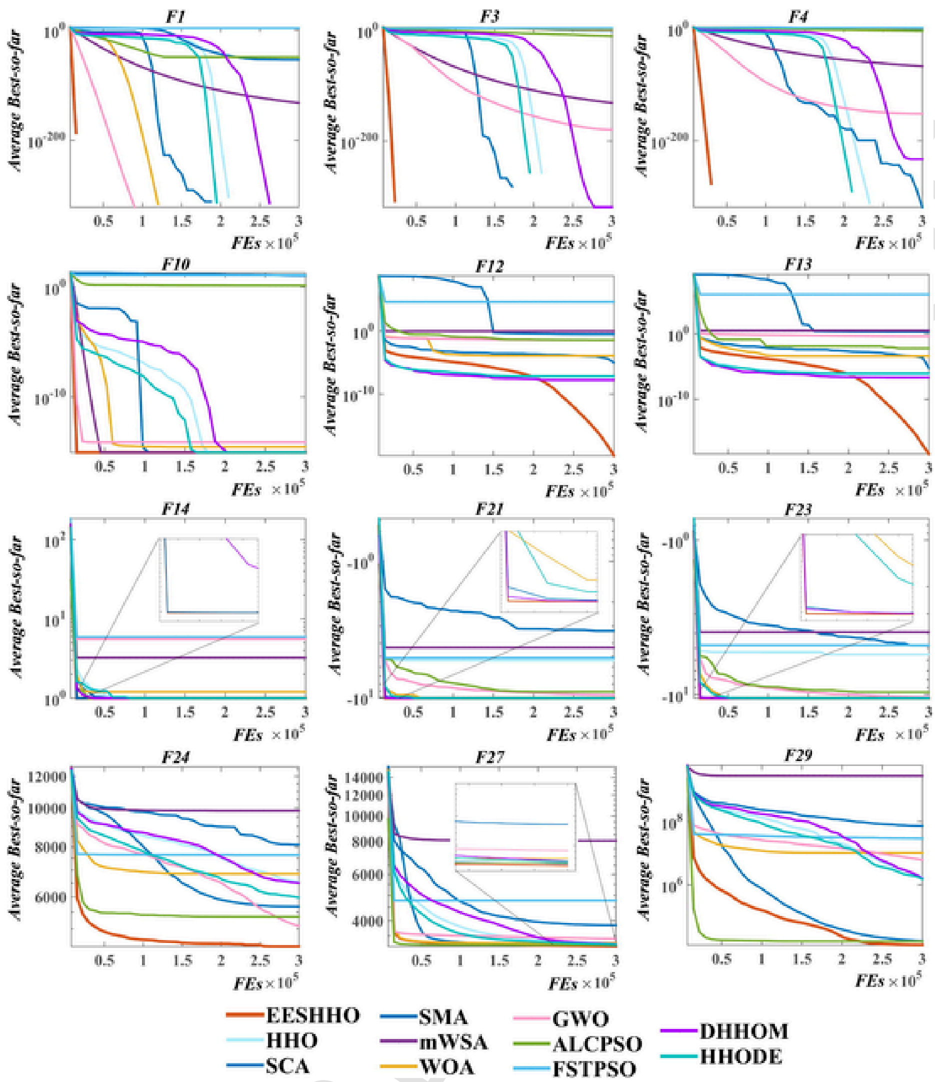
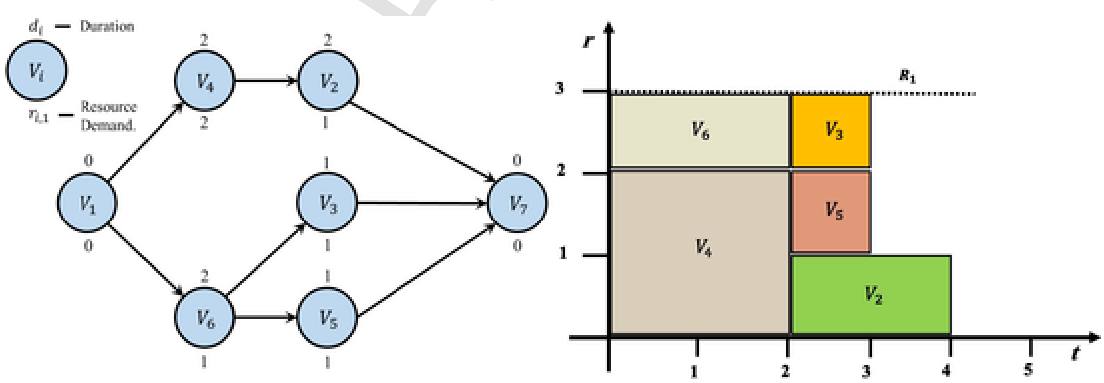


Fig. 8. Convergence curves of all compared algorithms in some of test functions.



(a). Precedence constraint relationship.

(b). Feasible scheduling table.

Fig. 9. An example of an RCPSP containing 7 activities.

number of resources required, and there is a precedence constraint relationship between the two activities connected by a directional arrow. For example, activity V_4 must begin after activity V_2 ends, and similarly, activity V_5 must begin after activity V_3 ends. If the project has only one resource, R_1, the total amount of its resources is 3. Fig. 9.(b) shows the feasible scheduling table for this project, where the horizontal axis shows the duration of the project, and the vertical axis is the resources consumed. It can be seen that the project duration is 4 while meeting the resource and preference constraints.

The scheduling table $T=\{T_1, T_2, \ldots, T_N\}$ defines the start time of a set of all activities, and the ultimate goal of solving RCPSP is to minimize the duration of the project while satisfying the resource and activity precedence constraints. The mathematical model is as follows:

$$\begin{array}{ll} Min & T_N \\ subject\ to & T_i \geqslant T_j + d_j, \forall j \in P_i \\ & \sum_{i \in A_i} r_{ik} \leqslant R_k, k = 1, 2, \ldots, K \\ & T_i \geqslant 0, i = 1, 2, \ldots, N \end{array} \tag{11}$$

where $A_i = \{i | T_i \leqslant t \leqslant T_i + d_i, i = 1, 2, \ldots N\}$ denotes the activity being performed at moment t. T_N denotes the completion time of the last virtual activity, which is also the objective function to be minimized. $(T_i \geqslant T_j + d_j, \forall j \in P_i)$ denotes the activity precedence constraint. $\left(\sum_{i \in A_i} r_{ik} \leqslant R_k, k = 1, 2, \ldots, K\right)$ denotes the resource constraint. $(T_i \geqslant 0, i = 1, 2, \ldots, N)$ denotes that the start time of all activities cannot be negative.

In solving RCPSP using the EESHHO algorithm, we encode the solution using "priority list, PL" (Baykasoğlu & Şenol, 2019) and decode it using "serial scheduling generation scheme, SSGS" (Baykasoğlu & Şenol, 2019). Fig. 10 shows this encoding and decoding process. First, we use the EESHHO algorithm to generate a vector $\vec{X}_V^{cont}$ whose upper and lower bounds are 1 and 0, respectively. Each dimension of the vector $\vec{X}_V^{cont}$ uniquely corresponds to an activity, and each dimension's value is the priority number of that activity. Subsequently, $\vec{X}_V^{cont}$ is sorted from smallest to largest, and the order of each activity in the sorting process also changes with the priority value of the corresponding dimension. Finally, the precedence constraints are also taken into account when generating the result, and if the constraints are not satisfied, a substitution operation (Kadam & Mane, 2015) is performed until all precedence constraints are satisfied. SSGS can generate a table of feasible schedules. It schedules each activity in order of sequence *act* within the precedence and resource constraints. And, The feasible scheduling table of in Fig. 10 has been described in Fig. 9.

3.2.1. Experimental results of RCPSP

This section tests the performance of EESHHO using the J30, I60, and J120 datasets mentioned in the literature (Baykasoğlu & Şenol, 2019), where J30, J60, and J120 contain 30, 60, and 120 activities, respectively. Also, the two datasets J30 and J60, contain 48 sets of data, each containing 10 test cases. Moreover, J120 contains 60 sets of data, each containing 10 test cases. In this experiment, we measure the performance of our algorithm using the evaluation index of average deviation (*AvgDev*), which is also used in many pieces of literature

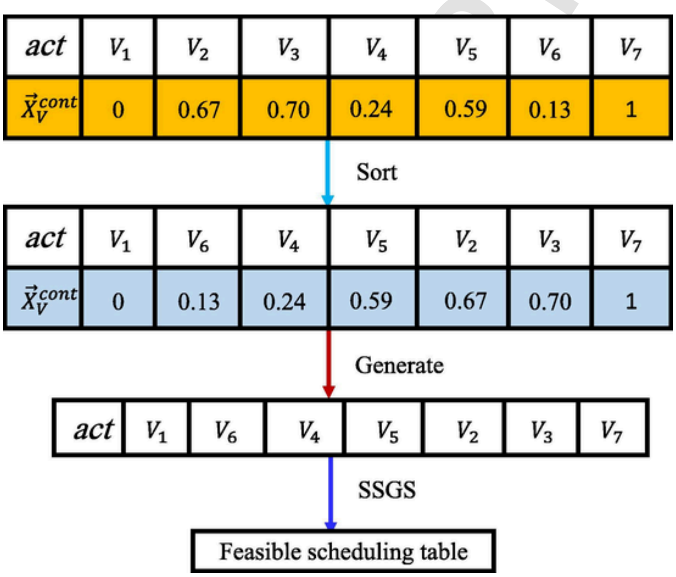


Fig. 10. An example RCPSP with solution encoding and decoding.

(Baykasoğlu & Şenol, 2019; Kadam & Mane, 2015; Kim & Ellis, 2010). The average deviation of J30 is calculated differently from that of J60 and J120. Since the optimal solution for each test case of the J30 data is already known, the average deviation from the optimal solution is used, which is mathematically described in Eq. (12). The optimal solution for each test case of the J60 and J120 data is unknown, but the lower bound for each test case has been obtained using the critical path method(CPM) (Baykasoğlu & Şenol, 2019), so the average deviation from this lower bound is used, and mathematically described in Eq. (13).

$$AvgDev_{opt} = \frac{\sum_{\text{instances}} \left(\frac{\text{Obtained}-\text{Opt}}{\text{Opt}} \times 100\%\right)}{\text{instances}} \tag{12}$$

$$AvgDev_{lb} = \frac{\sum_{instances} \left(\frac{\text{Obtained}-\text{Lb}}{\text{Lb}} \times 100\%\right)}{\text{instances}} \tag{13}$$

In this section, 18 mainstream algorithms that have been used for applications to the RCPSP are used as comparison algorithms whose test results about the RCPSP are derived from their original literature. Also, to reduce our algorithms' bias in optimizing RCPSP due to uncertainties, we run each test case independently 10 times and take the average of the 10 times as the final optimization result. The algorithm's termination condition was set to 1000 fitness function (See Eq. (11).) evaluations and 5000 fitness function evaluations.

The results of this average deviation for J30, J60, and J120 are described in Table 9, and these results are in the form of percentages. Meanwhile, The best results for each scenario are indicated in bold. The results obtained by EESHHO on the J30 dataset are not very competitive compared to other comparison algorithms, with average deviations of 1.5% and 0.89% for the 1000 and 5000 evaluation scenarios, respectively. The number of test cases in which EESHHO reached optimum in 480 test cases under 1000 evaluation scenarios was 320, and the number of test cases that reached optimum under 5000 evaluation scenarios was 358. From the results, EESHHO still has room for further optimization in solving certain test cases in the J30 dataset. As the number of evaluations increases, EESHHO can hopefully successfully resolve more test cases in the J30 dataset. For the J60 dataset, EESHHO took the best test results among all comparison algorithms for the 1000 evaluation scenarios, achieving an average deviation of 3.8%. Although not the best performance in the 5000 evaluation scenario, it only lagged behind the opposition based cWSA algorithm by 0.17%. EESHHO achieved an average deviation of 3.1% on the J60 dataset in the 5000 evaluation scenario. For the J120 dataset, EESHHO still takes the best test results among all comparison algorithms, reaching an average deviation of 13.07% and 11.50% for the 1000 and 5000 evaluation scenarios, respectively. Because the optimal value for each test case in the J120 dataset is unknown, and the lowest lower bound based on CPM is known. So EESHHO has 66 test cases out of 600 test cases that reach the lowest lower bound in the 1000 evaluation scenarios and 90 test cases that reach the lowest lower bound in the 5000 evaluation scenarios. In summary, EESHHO's comprehensive test results are highly competitive compared to the above comparison algorithm and can be used as a viable algorithm to solve such optimization problems. To further test the effectiveness of the EESHHO, in the next section, we apply EESHHO to the QoS-aware web service composition optimization problem, which is a practical engineering problem that has been gaining attention in recent years with the rapid development of web services.

3.3. EESHHO for QoS-aware web service composition

In cloud manufacturing and the internet of things, we may face a challenging service composition problem that can be utilized in a wide variety of applications (Lv & Xiu, 2020; Lv & Song, 2019; Lv & Kumar, 2020). QoS-aware web service composition problem is usually

Table 9
the average deviation results of EESHHO and other comparison algorithms for J30, J60 and J120.

Algorithms	J30		J60		J120	
	1000	5000	1000	5000	1000	5000
EESHHO (Present work)	1.5	0.89	**3.8**	3.1	**13.07**	**11.50**
Opposition based cWSA (Baykasoğlu & Şenol, 2019)	0.59	0.16	4.28	**2.93**	15.48	14.72
cWSA (Baykasoğlu & Şenol, 2019)	0.66	0.28	4.58	3.12	16.11	15.32
COAs (Elsayed et al., 2017)	0.04	**0**	11.13	10.77	34.04	32.9
MAOA (Zheng & Wang, 2015)	0.17	0.06	11.67	10.84	33.87	32.64
GA-MBX (Zamani, 2013)	0.14	0.04	11.33	10.94	34.02	32.89
GANS (Proon & Jin, 2011)	1.83	1.27	11.35	10.53	33.35	31.51
ACO + SS (Chen et al., 2010)	0.14	0.06	11.75	10.98	35.19	32.48
SFLA (Fang & Wang, 2012)	0.36	0.21	11.44	10.87	34.83	33.2
PSO–HH (Koulinas et al., 2014)	0.26	0.04	11.74	11.13	35.2	32.59
GA (Mendes et al., 2009)	0.06	0.02	11.72	11.04	35.87	33.03
GANN (Agarwal et al., 2011)	0.13	0.1	11.51	11.29	34.65	34.15
HEDA (Wang & Fang, 2012)	0.38	0.14	11.97	11.43	35.44	33.61
PSO (Chen, 2011)	0.29	0.14	12.03	11.43	35.71	33.88
JPSO (Chen, 2011)	0.29	0.14	12.03	11.43	35.71	35.88
BA(Ziarati et al., 2011)	0.42	0.19	12.55	12.04	37.72	36.76
GA-activitylist (Hartmann, 1998)	0.54	0.25	12.68	11.89	39.37	36.74
TS (Klein, 2000)	0.46	0.16	12.97	12.18	40.86	37.88
GLSA (Kadam & Mane, 2015)	**0.03**	N/A	4.02	N/A	15.6	N/A

formulated as a combinatorial optimization problem, which is an NP-hard problem (Li, Li, & Chen, 2020). Suppose N is the number of abstract services (service composition tasks) in the process of service composition, and $S=\{T_1,T_2,T_3,\ldots,T_N\}$ is defined as composite services. Meanwhile, suppose M is the number of concrete services (candidate services) for each abstract service, and $C_u=\{c_u^1,c_u^2,c_u^3,\ldots,c_u^M\}$ is the concrete services for T_u. D is the number of QoS (Quality of Service) attributes (Li et al., 2014). The mathematical model of this combinatorial optimization problem is expressed as follows:

$$\begin{array}{ll} \textit{Maximize} & \sum_{k=1}^{D} w_k f_k\left(\left\{\sum_{i=1}^{N}\sum_{j=1}^{M} x_{ij} c_i^j\right\}\right) \\ \textit{subject to} & f_k\left(\left\{\sum_{i=1}^{N}\sum_{j=1}^{M} x_{ij} q'_{ijk}\right\}\right) \geqslant L_k, \\ & \sum_{k=1}^{D} w_k = 1, x_{ij}\in\{0,1\}, \\ & \sum_{j=1}^{M} x_{ij} = 1, \\ & k=1,2,\ldots,D, \\ & i=1,2,..N, j=1,2,\ldots M. \end{array} \quad (14)$$

where w_k is the weight of the *k-th* QoS attribute. $f(\cdot)$ represents the aggregation function (Li et al., 2014), and $f_k(\cdot)$ is used to calculate the combined value of attribute *k*. x_{ij} is a concrete service selection operation (each abstract service allows only one concrete service to be selected), which indicates whether concrete service *j* is selected for abstract service *i*. L_k indicates the lower bound of the *k-th* QoS attribute, and the upper bound can be transformed into the lower bound (Wang, Xu, Sheng, Wang, & Yao, 2019). When the problem is optimized, it must not only satisfy this global QoS attribute constraint but also reach the maximum value of Eq. (14).

It is important to note that the aggregate function $f(\cdot)$ varies with the composition workflow's structural patterns, which have four basic types of structural patterns, including sequence combination pattern, loop pattern, parallel pattern, and conditional pattern (Jaeger, Rojec-Goldmann, & Muhl, 2004). For example, for the QoS attribute of *response time*, the aggregate function is the sum of the response time of all components in the sequential combination pattern while takes the maximum in the parallel pattern. Sine previous researchers have conducted in-depth research on this subject, and this loop pattern, parallel pattern, and conditional pattern can be reduced or converted to sequential modes using techniques that handle multiple execution paths (Cardoso, Sheth, Miller, Arnold, & Kochut, 2004), so in our work, we only consider the sequential pattern.

The EESHHO application to the QoS-aware web service composition optimization problem uses an integer coding approach, which is the approach adopted by most of the literature (Gavvala, Jatoth, Gangadharan, & Buyya, 2019; Chandra & Niyogi, 2019; Huang, Li, & Tao, 2014). Fig. 11 shows the process of composing services under the integer coding approach. Enter a composite service $S=\{T_1,T_2,T_3,\ldots,T_N\}$, which is the sequential combination pattern. Each task corresponds to an abstract service, e.g., T_1 corresponds to the abstract service C_1. Each abstract service contains *m* number of concrete services, e.g., C_1 contains many concrete services that have the same function but different QoS. Subsequently, one concrete service is selected from many concrete services corresponding to one abstract service, and similarly, one concrete service is selected for each abstract service. Finally, these concrete services are combined to reach the optimal value in Eq. (14). Each abstract service corresponds to all concrete services with their own unique number in this combination of integer-coded services. EESHHO solves this problem by performing an integer operation on the final solution, thus enabling the service combination operation. For example, a solution $S=\{\#3,\#5,\#10,\#8\}$, where #3 represents the first abstract service corresponding to the concrete service numbered 3, #5 is the second abstract service corresponding to the concrete service numbered 5, and so on.

3.3.1. Experimental setup and metrics

In this section, EESHHO was compared with several mainstream algorithms, which have been used to solve the problem of QoS-aware web service composition. Meanwhile, the original HHO was also tested simultaneously, even though it has not been used to solve the problem so far. All the compared algorithms, including EESHHO, Eagle Strategy with Whale Optimization Algorithm (ESWOA) (Gavvala et al., 2019), modified Artificial Bee Colony(mABC) (Chandra & Niyogi, 2019), Chaos Control Optimal Algorithm(CCOA) (Huang et al., 2014), and HHO runs in the same experimental environment. The parameter set-

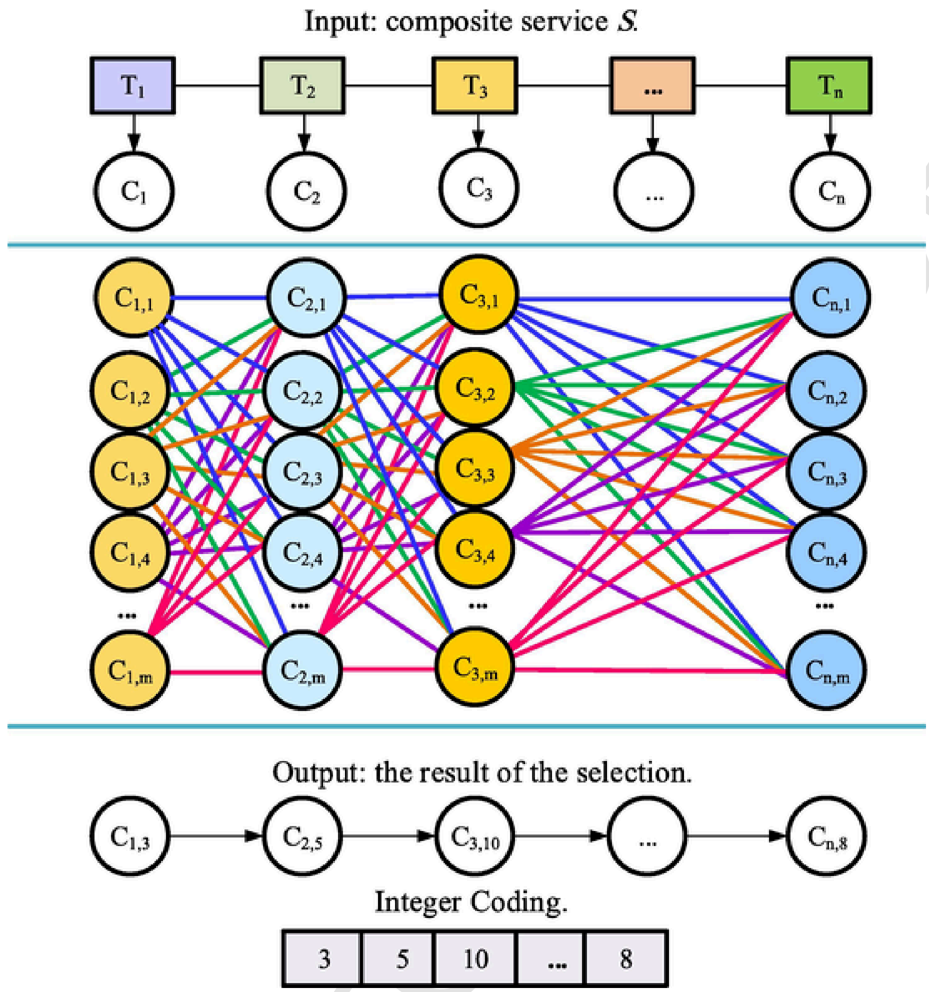


Fig. 11. A brief process of Qos-aware web service composition.

tings of EESHHO and HHO have been described (see Section 3.1). ESWOA, CCOA, and mABC are described below:

- ESWOA is an evolved version of WOA proposed in 2018 and has been proven to have a good performance in solving QoS-aware web service composition problem. ESWOA has a fixed parameter P_e to balance the transition between the exploration and exploitation stages, and is set to $P_e = 0.2$ according to the original literature requirements (Gavvala et al., 2019).
- mABC introduces the opposite learning method and differential evolution strategy based on chaos into the ABC algorithm to solve the QoS-aware web service composition problem. It has an important parameter *limit* to control the execution frequency of the exploration bee, which is set to $limit = (mn/2)$ (where *m* is the population size and *n* is the dimension size).
- CCOA is a meta-heuristic algorithm specifically designed for combinatorial optimization problems. It uses chaos to control the entire algorithm's search process, thereby improving the search efficiency in large-scale spaces. It is more effective than GA, PSO, and other meta-heuristic algorithms on combinatorial optimization problems (Huang et al., 2014). The setting parameter *A* is set to 3.

All comparison algorithms' population size was set to 60, and the maximum fitness evaluation was set to 15000. We adopted the QWS2.0 dataset (Al-Masri & Mahmoud, 2007), which contains 2507 real-world services with their 9 QoS attribute information. In this study, we only discuss 3 QoS attributes of each service: *response time*, *reliability*, and *latency*. The global constraints of all test cases are generated dynamically in the following ways: we first calculate the average QoS value of each attribute based on all concrete services in each abstract service, then obtain their aggregate values through the existing QoS attribute aggregation functions. Finally, each QoS attribute's global constraint is defined by taking 0.9 times of its aggregate value.

We define these test cases through *M* and *N*, where *M* represents the number of abstract services, and *N* represents the number of concrete services per abstract service. For example, #($N = 30$, $M = 50$), #($N = 30$, $M = 350$) and #($N = 60$, $M = 350$) are different test cases. We evaluate the algorithms' performance in solving QoS-aware web service composition problem using *csQos* and *CT*metrics. *csQos* is the value of the objective test function calculated by Eq. (14), which falls within the range of (0,1) after standardization. *CT* represents the computation time of the algorithm to get the final solution. All the experimental results are the average of the results after each algorithm runs 30 times to ensure reliable evaluation.

3.3.2. Evaluate comparison algorithms

Table 10 shows the performance of the different algorithms in all test cases (varying *N* and *M*). The challenge of this optimization task increases with the increase of *N*, which is the reason why the optimization value is generally low when $N = 90$. It is worth noting that EESHHO still achieves the best results in this complicated test case. Moreover, EESHHO achieved the best performance of all comparison algorithms in all test cases (only for *csQos* metric). This may be because the excellent genes of elite individuals preserved in EESHHO provide a

Table 10
Results of the comparison algorithms are in different test cases(QoS-aware web service composition)

Metric	Algorithm	$N = 30$				$N = 60$				$N = 90$			
		$M = 50$	$M = 350$	$M = 650$	$M = 950$	$M = 50$	$M = 350$	$M = 650$	$M = 950$	$M = 50$	$M = 350$	$M = 650$	$M = 950$
csQos	EESHHO	0.712	0.820	0.821	0.814	0.686	**0.691**	**0.692**	**0.692**	**0.682**	**0.690**	**0.690**	**0.691**
	ESWOA	0.699	0.755	0.744	0.738	0.682	0.689	0.688	0.689	0.677	0.687	0.686	0.687
	mABC	0.673	0.693	0.699	0.695	0.667	0.682	0.684	0.684	0.663	0.681	0.682	0.682
	CCOA	0.686	0.714	0.718	0.720	0.678	0.685	0.686	0.687	0.675	0.682	0.683	0.684
	HHO	0.679	0.702	0.700	0.698	0.670	0.684	0.684	0.685	0.665	0.681	0.682	0.683
CT(s)	EESHHO	0.304	0.313	0.324	0.319	0.323	0.332	**0.339**	0.335	**0.343**	0.351	0.359	**0.359**
	ESWOA	**0.295**	**0.308**	0.303	**0.308**	0.347	0.347	0.348	0.355	0.367	0.383	0.386	0.389
	mABC	0.300	0.315	0.310	0.313	0.367	0.364	0.374	0.381	0.398	0.414	0.412	0.417
	CCOA	0.435	0.413	0.417	0.421	0.513	0.423	0.440	0.446	0.505	0.432	0.436	0.439
	HHO	0.436	0.459	0.465	0.452	0.481	0.501	0.497	0.501	0.514	0.542	0.538	0.532

potential incentive for optimizing this problem to promote the algorithm to find a better solution.

EESHHO mostly obtains the shortest computation time. Meanwhile, the computation time of EESHHO is more affected by *N*, but not sensitive to *M*, and the above comparison of meta-heuristic algorithms shows a similar phenomenon, which is that compared with *M*, the influence of *N* will be more significant. The reason for this phenomenon is that *M* determines the search space for this problem. Simultaneously, the meta-heuristic method relies on the global exploration strategy and local exploitation strategy to optimize in the search space. As long as the maximum fitness evaluation is fixed, no matter how the search space changes, the calculation time will not be significantly affected. Therefore, these algorithms' computing time is more affected by the algorithm's complexity, the number of *N*, and the algorithm code's optimization. Our algorithm's computing time is very competitive in most test cases under a similar coding environment and the same number of *N*. This result can be explained by the simple structure of our algorithm and its low computational complexity.

As a summary, the result of optimizing this QoS-aware web service composition problem shows that the proposed EESHHO algorithm has the potential competitiveness in solving this kind of real-world combinatorial optimization problem.

4. Conclusion and future work

In this study, we proposed a novel meme algorithm, named EESHHO, based on the defects (for some optimization cases) of the meta-heuristic algorithms HHO as an entry point and comprehensively improving the performance of the original HHO. We were first inspired by the meta-heuristic algorithm based on evolution and proposed a novel Elite Evolutionary Strategy (EES) to deal with the shortcomings of the original HHO, which is slow to converge and easily fall into the local optimum. Second, the original HHO and EES are deeply fused to maximize their performance. Third, an extensive study of ESHHO was performed on 29 numerically optimized test functions (including 23 classic basic test functions and 6 composite test functions from the CEC2017 special session to analyze its ability to exploitation, exploration, the balance of exploration and exploitation and convergence. The experimental results show that EESHHO exhibits the best overall performance compared to other comparative meta-inspired algorithms. Finally, to further evaluate the performance of EESHHO, we used two real-world cases as benchmarks for the experiment: the resource-constrained project scheduling problem and the QoS-aware web service composition problem. Extensive experimental results show that EESHHO is specifically more competitive than other mainstream meta-heuristic algorithms on the above two NP-hard combinatorial optimization problems (QoS-aware web service composition and the resource-constrained project scheduling). In future work, we will try to use EESHHO to solve more real-world optimization problems. The source code for EESHHO in this paper can be found at https://www.researchgate.net/profile/Chenyang_Li39/research and https://aliasgharheidari.com/publications/EESHHO.html.

CRediT authorship contribution statement

ChenYang Li: Conceptualization, Methodology, Resources, Software, Writing - original draft, Investigation, Formal analysis. **Jun Li:** Writing - review & editing, Resources, Investigation, Supervision, Project administration, Funding acquisition. **HuiLing Chen:** Validation, Software, Investigation, Formal analysis, Data curation, Resources, Software. **Ali Asghar Heidari:** Visualization, Software, Investigation, Formal analysis, Resources, Writing - review & editing.

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 work was supported by the Science and Technology Plan Project of Wenzhou, China (No. 2020G0055).

References

Agarwal, A., Colak, S., & Erenguc, S. (2011). A neurogenetic approach for the resource-constrained project scheduling problem. Computers & Operations Research, 38, 44–50.
Al-Masri, E., & Mahmoud, Q. (2007). Qos-based discovery and ranking of web services. 2007 16th International Conference on Computer Communications and Networks (pp. 529–534).
Banzhaf, W., & Koza, J. (2000). Genetic programming. IEEE Intelligent Systems and their Application, 15. P.74–84.
Bao, X., Jia, H., & Lang, C. (2019). A novel hybrid harris hawks optimization for color image multilevel thresholding segmentation. IEEE Access, 7, 76529–76546.
Baykasoglu, A. (2012). Design optimization with chaos embedded great deluge algorithm. Applied Soft Computing, 12, 1055–1067.
Baykasoğlu, A., & Akpinar, Şener (2015). Weighted superposition attraction (wsa): A swarm intelligence algorithm for optimization problems – part 1: Unconstrained optimization. Applied Soft Computing, 37, 520–540.
Baykasoğlu, A., & Akpinar, Şener (2020). Enhanced superposition determination for weighted superposition attraction algorithm. Soft Computing, 24, 15015–15040.
Baykasoğlu, A., & Şenol, M.E. (2019). Weighted superposition attraction algorithm for combinatorial optimization. Expert Systems With Applications, 138. 112792.
Beheshti, Z. (2013). A review of population-based meta-heuristic algorithm. In International Journal of Advances in Soft Computing and its Applications (pp. 1–35). volume 5.
Birogul, S. (2019). Hybrid harris hawk optimization based on differential evolution (hhode) algorithm for optimal power flow problem. IEEE Access, 7, 184468–184488.
Bäck, T., & Schwefel, H.-P. (1993). An overview of evolutionary algorithms for parameter optimization. Evolutionary Computation, 1, 1–23.
Cardoso, J., Sheth, A.P., Miller, J.A., Arnold, J., & Kochut, K.J. (2004). Quality of service for workflows and web service processes. Journal of Web Semantics, 1, 281–308.
Chandra, M., & Niyogi, R. (2019). Web service selection using modified artificial bee colony algorithm. IEEE Access, 7, 88673–88684.
Chao, M., Kai, C., & Zhiwei, Z. (2020). Research on tobacco foreign body detection device based on machine vision. Transactions of the Institute of Measurement and Control. (p. 0142331220929816)..
Chen, H., Heidari, A.A., Chen, H., Wang, M., Pan, Z., & Gandomi, A.H. (2020). Multi-population differential evolution-assisted harris hawks optimization: Framework and case studies. Future Generation Computer Systems, 111, 175–198.
Chen, H., Jiao, S., Wang, M., Heidari, A.A., & Zhao, X. (2020). Parameters identification of photovoltaic cells and modules using diversification-enriched harris hawks optimization with chaotic drifts. Journal of Cleaner Production, 244. 118778.
Chen, R.-M. (2011). Particle swarm optimization with justification and designed mechanisms for resource-constrained project scheduling problem. Expert Systems with Applications, 38, 7102–7111.
Chen, R.-M. (2011). Particle swarm optimization with justification and designed mechanisms for resource-constrained project scheduling problem. Expert Systems With Applications, 38, 7102–7111.
Chen, W., Shi, Y.-J., Teng, H.-F., Lan, X.-P., & Hu, L.-C. (2010). An efficient hybrid algorithm for resource-constrained project scheduling. Information Sciences, 180, 1031–1039.
Chen, W.-N., Zhang, J., Lin, Y., Chen, N., Zhan, Z.-H., Chung, H.S.-H., … Shi, Y.-H. (2013). Particle swarm optimization with an aging leader and challengers. IEEE Transactions on Evolutionary Computation, 17, 241–258.
Das, S., Biswas, A., Dasgupta, S., & Abraham, A. (2009). Bacterial foraging optimization algorithm: theoretical foundations, analysis, and applications. Foundations of computational intelligence volume 3 (pp. 23–55). Springer.
Dorigo, M., Birattari, M., & Stützle, T. (2006). Ant colony optimization: Artificial ants as a computational intelligence technique. IEEE Computational Intelligence Magazine, 1, 28–39.
Elsayed, S., Sarker, R., Ray, T., & Coello, C.C. (2017). Consolidated optimization algorithm for resource-constrained project scheduling problems. Information Sciences, 418, 346–362.
Emary, E., Zawbaa, H.M., & Sharawi, M. (2019). Impact of lèvy flight on modern meta-heuristic optimizers. Applied Soft Computing, 75, 775–789.
Fang, C., & Wang, L. (2012). An effective shuffled frog-leaping algorithm for resource-constrained project scheduling problem. Computers & Operations Research, 39, 890–901.
Formato, R.A. (2008). Central force optimization: A new nature inspired computational framework for multidimensional search and optimization. NICSO, 129, 221–238.
Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. Journal of the American Statistical Association, 32, 675–701.

Fu, X., Fortino, G., Li, W., Pace, P., & Yang, Y. (2019). Wsns-assisted opportunistic network for low-latency message forwarding in sparse settings. Future Generation Computer Systems, 91, 223–237.

Fu, X., Fortino, G., Pace, P., Aloi, G., & Li, W. (2020). Environment-fusion multipath routing protocol for wireless sensor networks. Information Fusion, 53, 4–19.

García, S., Fernández, A., Luengo, J., & Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. Information Sciences, 180, 2044–2064.

Gavvala, S.K., Jatoth, C., Gangadharan, G., & Buyya, R. (2019). Qos-aware cloud service composition using eagle strategy. Future Generation Computer Systems, 90, 273–290.

Goldberg (2008). Genetic Algorithms.

Goldberg, D.E., & Holland, J.H. (1988). Genetic algorithms and machine learning. Machine Learning, 3, 95–99.

Hartmann, S. (1998). A competitive genetic algorithm for resource-constrained project scheduling. Naval Research Logistics, 45, 733–750.

Heidari, A.A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M.M., & Chen, H. (2019). Harris hawks optimization: Algorithm and applications. Future Generation Computer Systems, 97, 849–872.

Huang, B., Li, C., & Tao, F. (2014). A chaos control optimal algorithm for qos-based service composition selection in cloud manufacturing system. Enterprise Information Systems, 8, 445–463.

Huang, X., & Yang, L. (2019). A hybrid genetic algorithm for multi-objective flexible job shop scheduling problem considering transportation time. International Journal of Intelligent Computing and Cybernetics.

Hwang, C.R. (1988). Simulated annealing: Theory and applications. Acta Applicandae Mathematica, 12, 108–111.

Jaeger, M., Rojec-Goldmann, G., & Muhl, G. (2004). Qos aggregation for web service composition using workflow patterns. In Proceedings. Eighth IEEE International Enterprise Distributed Object Computing Conference, 2004. EDOC 2004. (pp. 149–159).

Jia, H., Lang, C., Oliva, D., Song, W., & Peng, X. (2019). Dynamic harris hawks optimization with mutation mechanism for satellite image segmentation. Remote Sensing, 11, 1421.

Kadam, S.U., & Mane, S.U. (2015). A genetic-local search algorithm approach for resource constrained project scheduling problem. 2015 International Conference on Computing Communication Control and Automation (pp. 841–846).

Kang, F., Li, J., & Ma, Z. (2011). Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions. Information Sciences, 181, 3508–3531.

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. Journal of Global Optimization, 39, 459–471.

Kennedy, J., & Eberhart, R. (2002). Particle swarm optimization. In Neural Networks, 1995. Proceedings., IEEE International Conference on (pp. 1942–1948). volume 4.

Kim, J.-L., & Ellis, R.D. (2010). Comparing schedule generation schemes in resource-constrained project scheduling using elitist genetic algorithm. Journal of Construction Engineering and Management-asce, 136, 160–169.

Klein, R. (2000). Project scheduling under time-varying resource constraints. International Journal of Production Research, 38, 3937–3952.

Koulinas, G., Kotsikas, L., & Anagnostopoulos, K. (2014). A particle swarm optimization based hyper-heuristic algorithm for the classic resource constrained project scheduling problem. Information Sciences, 277, 680–693.

Li, C., Li, J., & Chen, H. (2020). A meta-heuristic based approach for qos-aware service composition. IEEE Access. 1–1.

Li, J., Zheng, X.-L., Chen, S.-T., Song, W.-W., & Chen, D.-R. (2014). An efficient and reliable approach for quality-of-service-aware service composition. Information Sciences, 269, 238–254.

Li, S., Chen, H., Wang, M., Heidari, A.A., & Mirjalili, S. (2020). Slime mould algorithm: A new method for stochastic optimization. Future Generation Computer Systems, 111, 300–323.

Liu, E., Li, W., Cai, H., & Peng, S. (2019). Formation mechanism of trailing oil in product oil pipeline. Processes, 7, 7.

Luo, J., Chen, H., zhang, Q., Xu, Y., Huang, H., & Zhao, X. (2018). An improved grasshopper optimization algorithm with application to financial stress prediction. Applied Mathematical Modelling, 64, 654–668.

Lv, Q., Liu, H., Wang, J., Liu, H., & Shang, Y. (2020). Multiscale analysis on spatiotemporal dynamics of energy consumption co2 emissions in china: Utilizing the integrated of dmsp-ols and npp-viirs nighttime light datasets. Science of the Total Environment, 703. 134394.

Lv, X., Li, N., Xu, X., & Yang, Y. (2020). Understanding the emergence and development of online travel agencies: a dynamic evaluation and simulation approach. Internet Research. doi:10.1108/INTR-11-2019-0464.

Lv, Z., & Kumar, N. (2020). Software defined solutions for sensors in 6g/ioe. Computer Communications, 153, 42–47.

Lv, Z., Li, X., Lv, H., & Xiu, W. (2019). Bim big data storage in webvrgis. IEEE Transactions on Industrial Informatics, 16, 2566–2573.

Lv, Z., & Qiao, L. (2020). Analysis of healthcare big data. Future Generation Computer Systems, 109, 103–110. doi:10.1016/j.future.2020.03.039.

Lv, Z., & Qiao, L. (2020). Deep belief network and linear perceptron based cognitive computing for collaborative robots. Applied Soft Computing. (p. 106300).

Lv, Z., & Song, H. (2019). Mobile internet of things under data physical fusion technology. IEEE Internet of Things Journal, 7, 4616–4624.

Lv, Z., & Xiu, W. (2020). Interaction of edge-cloud computing based on sdn and nfv for next generation iot. IEEE Internet of Things Journal, 7, 5706–5712. doi:10.1109/JIOT.2019.2942719.

Maharana, D., Kommadath, R., & Kotecha, P. (2017). Dynamic yin-yang pair optimization and its performance on single objective real parameter problems of cec 2017. 2017 IEEE Congress on Evolutionary Computation (CEC) (pp. 2390–2396).

Mendes, J.J., Gonçalves, J.F., & Resende, M.G. (2009). A random key based genetic algorithm for the resource constrained project scheduling problem. Computers & Operations Research, 36, 92–109.

Mirjalili, S. (2016). Sca: A sine cosine algorithm for solving optimization problems. Knowledge Based Systems, 96, 120–133.

Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. Advances in Engineering Software, 95, 51–67.

Mirjalili, S., Mirjalili, S.M., & Lewis, A. (2014). Grey wolf optimizer. Advances in Engineering Software, 69, 46–61.

Neapolitan, R., & Naimipour, K. (2009). Foundations of algorithms (fourth ed.). Jones & Bartlett Learning.

Niu, P., Niu, S., & Chang, L., et al. (2019). The defect of the grey wolf optimization algorithm and its verification method. Knowledge-Based Systems, 171, 37–43.

Nobile, M.S., Cazzaniga, P., Besozzi, D., Colombo, R., Mauri, G., & Pasi, G. (2017). Fuzzy self-tuning pso: A settings-free algorithm for global optimization. Swarm and Evolutionary Computation, 39, 70–85.

Pan, W.-T. (2012). A new fruit fly optimization algorithm: Taking the financial distress model as an example. Knowledge Based Systems, 26, 69–74.

Proon, S., & Jin, M. (2011). A genetic algorithm with neighborhood search for the resource-constrained project scheduling problem. Naval Research Logistics (NRL), 58, 73–82.

Rashedi, E., Nezamabadi-pour, H., & Saryazdi, S. (2009). Gsa: A gravitational search algorithm. Information Sciences, 179, 2232–2248.

Rechenberg, I. (1978). Evolutionsstrategien. Berlin Heidelberg: Springer.

Ridha, H.M., Heidari, A.A., Wang, M., & Chen, H. (2020). Boosted mutation-based harris hawks optimizer for parameters identification of single-diode solar cell models. Energy Conversion and Management, 209. 112660.

Shi, K., Wang, J., Tang, Y., & Zhong, S. (2020). Reliable asynchronous sampled-data filtering of t–s fuzzy uncertain delayed neural networks with stochastic switched topologies. Fuzzy Sets and Systems, 381, 1–25.

Simon, D. (2008). Biogeography-based optimization. IEEE Transactions on Evolutionary Computation, 12, 702–713.

Storn, R., & Price, K. (1997). Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 11, 341–359.

Sun, G., Xu, G., & Jiang, N. (2020). A simple differential evolution with time-varying strategy for continuous optimization. Soft Computing, 24, 2727–2747.

Sun, G., Yang, B., Yang, Z., & Xu, G. (2019). An adaptive differential evolution with combined strategy for global numerical optimization. Soft Computing, 1–20.

Wang, H.-C., Lee, C.S., & Ho, T.H. (2007). Combining subjective and objective qos factors for personalized web service selection. Expert Systems With Applications, 32, 571–584.

Wang, L., & Fang, C. (2012). A hybrid estimation of distribution algorithm for solving the resource-constrained project scheduling problem. Expert Systems with Applications, 39, 2451–2460.

Wang, L., Zeng, Y., & Chen, T. (2015). Back propagation neural network with adaptive differential evolution algorithm for time series forecasting. Expert Systems With Applications, 42, 855–863.

Wang, S., Zhang, K., van Beek, L.P., Tian, X., & Bogaard, T.A. (2020). Physically-based landslide prediction over a large region: Scaling low-resolution hydrological model results for high-resolution slope stability assessment. Environmental Modelling & Software, 124. 104607.

Wang, X., Xu, X., Sheng, Q.Z., Wang, Z., & Yao, L. (2019). Novel artificial bee colony algorithms for qos-aware service selection. IEEE Transactions on Services Computing, 12, 247–261.

Wei, Y., Lv, H., Chen, M., Wang, M., Heidari, A.A., Chen, H., & Li, C. (2020). Predicting entrepreneurial intention of students: An extreme learning machine with gaussian barebone harris hawks optimizer. IEEE Access, 8, 76841–76855.

Wen, D., Zhang, X., Liu, X., & Lei, J. (2017). Evaluating the consistency of current mainstream wearable devices in health monitoring: a comparison under free-living conditions. Journal of Medical Internet Research, 19. e68.

Wolpert, D., & Macready, W. (1997). No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation, 1, 67–82.

Wu, T., Xiong, L., Cheng, J., & Xie, X. (2020). New results on stabilization analysis for fuzzy semi-markov jump chaotic systems with state quantized sampled-data controller. Information Sciences, 521, 231–250.

Xie, J., Wen, D., Liang, L., Jia, Y., Gao, L., & Lei, J. (2018). Evaluating the validity of current mainstream wearable devices in fitness tracking under various physical activities: Comparative study. JMIR mHealth and uHealth, 6. e94.

Xu, Y., Chen, H., Heidari, A.A., Luo, J., Zhang, Q., Zhao, X., & Li, C. (2019). An efficient chaotic mutative moth-flame-inspired optimizer for global optimization tasks. Expert Systems With Applications, 129, 135–155.

Yang, S., Deng, B., Wang, J., Li, H., Lu, M., Che, Y., … Loparo, K.A. (2019). Scalable digital neuromorphic architecture for large-scale biophysically meaningful neural network with multi-compartment neurons. IEEE Transactions on Neural Networks and Learning Systems, 31, 148–162.

Yang, X.-S., & Deb, S. (2009). Cuckoo search via lèvy flights. In 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC) (pp. 210–214).

Zamani, R. (2013). A competitive magnet-based genetic algorithm for solving the resource-constrained project scheduling problem. European Journal of Operational Research, 229, 552–559.

Zeng, H.-B., Liu, X.-G., Wang, W., & Xiao, S.-P. (2019). New results on stability analysis of systems with time-varying delays using a generalized free-matrix-based inequality. Journal of the Franklin Institute, 356, 7312–7321.

Zhang, H., Qu, S., Li, H., Luo, J., & Xu, W. (2020). A moving shadow elimination method based on fusion of multi-feature. IEEE Access, 8, 63971–63982.

Zhao, C., & Li, J. (2020). Equilibrium selection under the bayes-based strategy updating rules. Symmetry, 12, 739.

Zheng, X.-L., & Wang, L. (2015). A multi-agent optimization algorithm for resource constrained project scheduling problem. Expert Systems with Applications, 42, 6039–6049.

Zhu, B., Ma, S., Xie, R., Chevallier, J., & Wei, Y.-M. (2018). Hilbert spectra and empirical mode decomposition: A multiscale event analysis method to detect the impact of economic crises on the european carbon market. Computational Economics, 52, 105–121.

Zhu, B., Pang, R., Chevallier, J., Wei, Y.-M., & Vo, D.-T. (2019). Including intangible costs into the cost-of-illness approach: a method refinement illustrated based on the pm 2.5 economic burden in china. The European Journal of Health Economics, 20, 501–511.

Ziarati, K., Akbari, R., & Zeighami, V. (2011). On the performance of bee algorithms for resource-constrained project scheduling problem. Applied Soft Computing, 11, 3720–3733.

Fu, X., Pace, P., Aloi, G., Yang, L., & Fortino, G. (2020). Topology Optimization Against Cascading Failures on Wireless Sensor Networks Using a Memetic Algorithm. Computer Networks, 177, 107327. https://doi.org/10.1016/j.comnet.2020.107327.

Cao, B., Zhao, J., Gu, Y., Ling, Y., & Ma, X. (2020). Applying graph-based differential grouping for multiobjective large-scale optimization. Swarm and Evolutionary Computation, 53, 100626. doi:https://doi.org/10.1016/j.swevo.2019.100626.

Cao, B., Fan, S., Zhao, J., Yang, P., Muhammad, K., & Tanveer, M. (2020). Quantum-enhanced multiobjective large-scale optimization via parallelism. *Swarm and Evolutionary*. Computation, 57, 100697. https://doi.org/10.1016/j.swevo.2020.100697.

Cao, B., Zhao, J., Yang, P., Gu, Y., Muhammad, K., … Rodrigues, J. J. (2019). Multiobjective 3-D Topology Optimization of Next-Generation Wireless Data Center Network. IEEE Transactions on Industrial Informatics, 16(5), 3597–3605.

Chen, H., Qiao, H., Xu, L., Feng, Q., & Cai, K. (2019). A Fuzzy Optimization Strategy for the Implementation of RBF LSSVR Model in Vis–NIR Analysis of Pomelo Maturity. IEEE Transactions on Industrial Informatics, 15(11), 5971–5979.

XueQ. ZhuY. WangJ. 2019 Joint Distribution Estimation and Naïve Bayes Classification under Local Differential Privacy," inIEEE Transactions on Emerging Topics in Computing10.1109/TETC.2019.2959581

Zhang, X., Wang, Y., Chen, X., Su, C. Y., Li, Z., Wang, C., & Peng, Y. (2018). Decentralized adaptive neural approximated inverse control for a class of large-scale nonlinear hysteretic systems with time delays. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 49(12), 2424–2437.

ZhangH. QiuZ. CaoJ. Abdel-AtyM. XiongL. Nov. 2020 Event-Triggered Synchronization for Neutral-Type Semi-Markovian Neural Networks With Partial Mode-Dependent Time-Varying Delays," inIEEE Transactions on Neural Networks and Learning Systems31114437445010.1109/TNNLS.2019.2955287

Zhang, Z., Liu, M., Zhou, M., & Chen, J. (2020). Dynamic reliability analysis of nonlinear structures using a Duffing-system-based equivalent nonlinear system method. International Journal of Approximate Reasoning, 126, 84–97.

ZhangC. ChenZ. WangJ. LiuZ. ChenC. L. P. 2021 Fuzzy Adaptive Two-Bit-Triggered Control for a Class of Uncertain Nonlinear Systems With Actuator Failures and Dead-Zone ConstraintIEEE Transactions on Cybernetics51121022110.1109/TCYB.2020.2970736

Hu, J., Chen, H., Heidari, A. A., Wang, M., Zhang, X., Chen, Y., & Pan, Z. (2020). Orthogonal learning covariance matrix for defects of grey wolf optimizer: Insights, balance, diversity, and feature selection. Knowledge-Based Systems, 213, 106684. https://doi.org/10.1016/j.knosys.2020.106684.

Qiu, T., Shi, X., Wang, J., Li, Y., Qu, S., Cheng, Q., & Sui, S. (2019). Deep learning: A rapid and efficient route to automatic metasurface design. Advanced Science, 6(12), 1900128.

Hu, J., Zheng, B., Wang, C., Zhao, C., Hou, X., Pan, Q., & Xu, Z. (2020). A survey on multi-sensor fusion based obstacle detection for intelligent ground vehicles in off-road environments. Frontiers Inf. Technol. Electron. Eng., 21(5), 675–692.