The Journal of Supercomputing
https://doi.org/10.1007/s11227-024-05909-0

Evolutionary multi-mode slime mold optimization: a hyper-heuristic algorithm inspired by slime mold foraging behaviors

Rui Zhong[1] · **Enzhi Zhang**[1] · **Masaharu Munetomo**[2]

Accepted: 7 January 2024

Abstract
This paper proposes a novel hyper-heuristic algorithm termed evolutionary multi-mode slime mold optimization (EMSMO) for addressing continuous optimization problems. The architecture of a typical hyper-heuristic algorithm comprises two main components: the high-level component and the low-level component. The low-level component contains a set of low-level heuristics (LLHs) and intrinsic problem attributes, while the high-level component manipulates the LLHs to construct the sequence of heuristics. Inspired by the foraging behaviors of slime mold, we designed four easy-implemented search strategies including the search for food, approach food, wrap food, and re-initialization as the LLHs for the low-level component. In the high-level component, we adopt an improvement-based probabilistic selection function that contains two metrics: (1) the probability of improvement and (2) the normalized improvement. The selection function cooperates with the roulette wheel strategy to construct the optimization sequence. To evaluate the performance of our proposal, we implement comprehensive numerical experiments on CEC2013 benchmark functions and three engineering optimization problems. Six classic or advanced evolutionary algorithms and three hyper-heuristic algorithms are applied as competitor algorithms to evaluate the competitiveness of EMSMO. Experimental and statistical results show that EMSMO has broad prospects for solving continuous optimization problems.

Keywords Hyper-heuristics · Bio-inspired optimization algorithm · Improvement-based probabilistic selection function · Continuous optimization

1 Introduction

In recent decades, evolutionary algorithm (EA) as a kind of flexible and efficient optimization methodology has been widely applied to many various fields and achieved great success [1–5]. Meanwhile, thousands of novel EAs have been

Extended author information available on the last page of the article

Published online: 09 February 2024

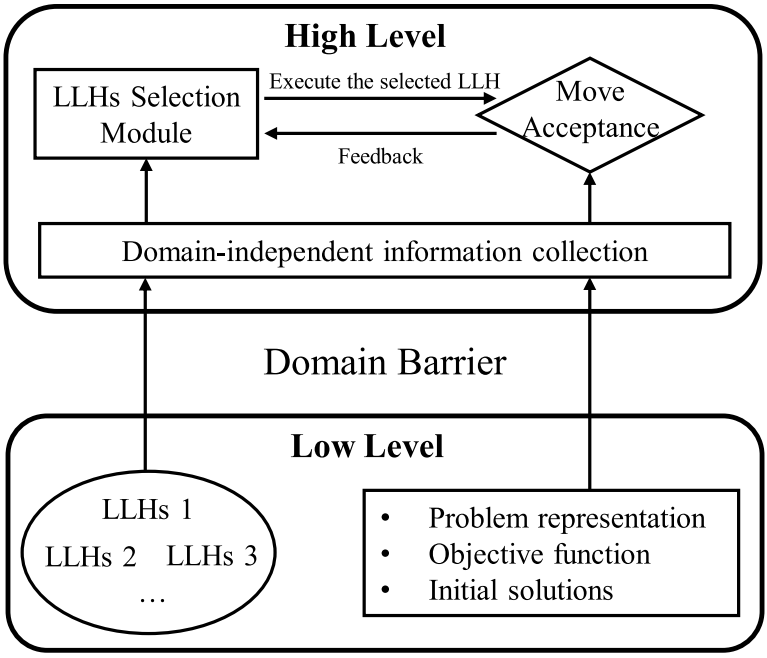

Fig. 1 A general structure of the HHs framework [19, 20]

proposed such as slime mold algorithm (SMA) [6], egret swarm optimization algorithm (ESOA) [7], vegetation evolution (VEGE) [8], and so on. Although the appearance of these new EAs injects vitality into the evolutionary computation (EC) community, researchers are also concerned with some explicit and implicit issues: (1) the novelty claimed in the new proposed EAs is limited [9–11]. (2) The inspiration of the proposed algorithm is even based on supernatural phenomena and the motivation is unreasonable [12]. (3) The validation experiment is poor [13, 14]. The existence of these problems is detrimental to the development of the EC community, and how to promote the advancement of the EC community sustainably becomes a meaningful topic.

Meanwhile, the invention of hyper-heuristics (HHs) promotes the EC community from another perspective. The origin of the HHs can be traced back to the early 1960s [15] and the HHs terminology was first coined in 2000 to describe the operation of "heuristics to choose heuristics" [16]. As an "off-the-peg" technique rather than "made-to-measure" meta-heuristics [17], HHs framework is a high-level automatic methodology to dominate a set of efficient low-level heuristics (LLHs) to produce solutions of acceptable quality [18]. A representative architecture of the HHs framework is shown in Fig. 1.

Two main parts consist of the HHs framework: high-level and low-level components. The high-level component acts as the "brain" of HHs framework to dominate the low-level component and determine the construction of the optimization sequence and the acceptance of the candidate solutions. The low-level component

contains the problem representation, the objective function(s), initial solutions, and a set of LLHs.

In the subsequent decades, the HHs algorithm has achieved rapid development: Burke et al. [21] adopted reinforcement learning (RL) to form the competition of LLHs and constructed the optimization sequence to deal with timetabling and rostering problems. A tabu list of heuristics is also maintained to prevent certain LLHs from being selected continuously. Terashima-Marín et al. [22] proposed a genetic algorithm (GA)-based HHs for the dynamic variable ordering within constraint satisfaction problems. After the training and testing periods, the produced optimal sequence of heuristics has great scalability on most of the benchmark problems. Burke et al. [23] presented a Monte Carlo-based hyper-heuristics to solve capacitated examination timetabling problems. A learning heuristic selection method operates in harmony with a simulated annealing move acceptance method using reheating based on some shared variables, and the experimental results show that simulated annealing with reheating as a hyper-heuristic move acceptance method has great potential. Lin et al. [24] proposed a backtracking search hyper-heuristic (BS-HH) for the distributed assembly permutation flow-shop scheduling problem (DAPFSP). The low-level of HHs contains ten simple but efficient heuristic strategies, a backtracking search is adopted as the high-level to manipulate the LLHs. Zhao et al. [25] presented a cooperative multi-stage hyper-heuristic (CMS-HH) algorithm to solve combinatorial optimization problems including Boolean satisfiability problems, one-dimensional packing problems, permutation flow-shop scheduling problems, personnel scheduling problems, traveling salesman problems, and vehicle routing problems. Lin et al. [26] proposed a Q-learning-based hyper-heuristic (QHH) algorithm to tackle the semiconductor final testing scheduling problem (SFTSP). Q-learning plays a high-level role in the HHs framework to control eight easy-implemented LLHs. However, most research concentrates on combinatorial optimization problems, and research dealing with high-level solvers for continuous optimization is scarce [27].

In this paper, we propose a novel HHs algorithm for continuous optimization named evolutionary multi-mode slime mold optimization (EMSMO). Inspired by the foraging behaviors of slime mold, we design four basic search strategies including search for food, approach food, wrap food, and re-initialization as the LLHs of the low-level component in EMSMO. In the high-level component of EMSMO, we design an improvement-based probabilistic selection function to determine the optimization sequence dynamically and smartly. Furthermore, we implement comprehensive experiments to evaluate the performance of our proposed EMSMO adequately. More specifically, the main contribution of this paper is as follows:

(1) This paper proposes a novel HHs algorithm, named EMSMO, to simulate the swarm intelligence of slime mold. In the low-level EMSMO, we design four easy-implemented search strategies inspired by the slime mold foraging behaviors as the LLHs, and in the high-level, we design an improvement-based probabilistic selection function that contains two metrics: (a) the probability of improvement (PI) and (b) the normalized improvement (NI). An unbiased combination of these two metrics consists of the foundation of the selection function and forms the competition of the strategy selection.

(2) We implement a set of numerical experiments on the CEC2013 [28] benchmark functions and three engineering optimization problems to evaluate the performance of our proposed EMSMO. The experiments among EMSMO and six classic or advanced EAs evaluate the overview optimization performance, and the experiments among EMSMO and three conventional HHs frameworks evaluate the performance of the improvement-based probabilistic selection function in optimization sequence determination. Experimental and statistical results show that our proposed EMSMO is competitive with the compared optimizers and has broad prospects to solve various optimization problems.

The remainder of this paper is organized as follows: Sect. 2 introduces the slime mold foraging behaviors and our designed mathematical models. Section 3 introduces our proposal EMSMO in detail. Section 4 contains the numerical experiments. Section 5 analyzes our proposal and lists some open topics. Finally, Sect. 6 concludes this paper.

2 Slime mold foraging behaviors and mathematical models

As the highest-level component of the nervous system, the brain is commonly regarded as the foundation of intelligence. However, the slime mold, as a kind of single-cell organism without a brain, can perform a certain level of swarm intelligence: Nakagaki et al. [29] found that the physarum polycephalum, one category of slime mold, can find the minimum route in a complex maze with the nutrition guide. Tero et al. [30] adopted the slime mold to design the traffic network of the Tokyo area, food sources are placed in 36 geographical locations of cities on the Tokyo map. As the slime mold grows gradually, the topology of the growth of slime mold is highly similar to the real traffic network. Adamatzky et al. [31] conducted similar experiments to approximate the real motorway networks in 14 countries by slime mold and got inspiring experimental results. Boussard et al. [32] considered that although the inner mechanism remains unknown, slime mold can utilize the information from previous experiences to modify its behaviors adaptively, and the oscillations may play an important role in learning and memory of the slime mold. These works reveal that even the simplest organismic structure of slime mold has incredible intelligence.

Figure 2 shows the foraging behavior of slime mold. In the initialization stage, the physarum polycephalum slime mold is placed randomly so that it may be far away from the food source, and the search behavior to unexplored regions is mainly adopted. When the slime mold approaches the food, the slight perturbation of movement allows careful exploitation to reach more nutritious locations. And when the slime mold is very close to the food source such as oat flakes, it will wrap the oat flakes and ingest them. However, when a certain direction of the slime mold cannot obtain sufficient nutrition to survive, the cells in this route will degenerate and die. Inspired by the slime mold behaviors, we design four search strategies to simplify the foraging behavior and realize the LLHs of our

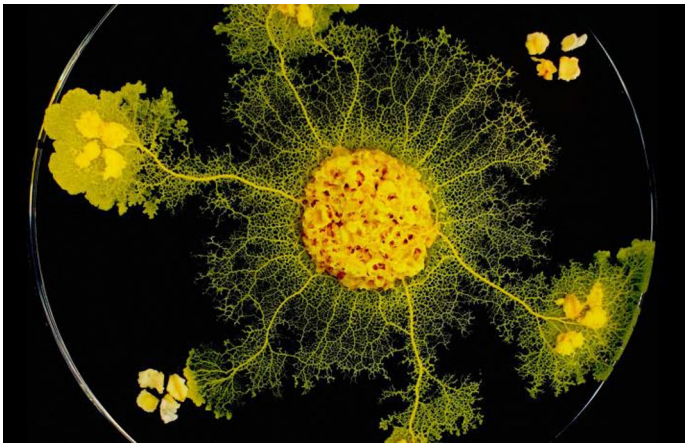

Fig. 2 The foraging behavior of slime mold [33]

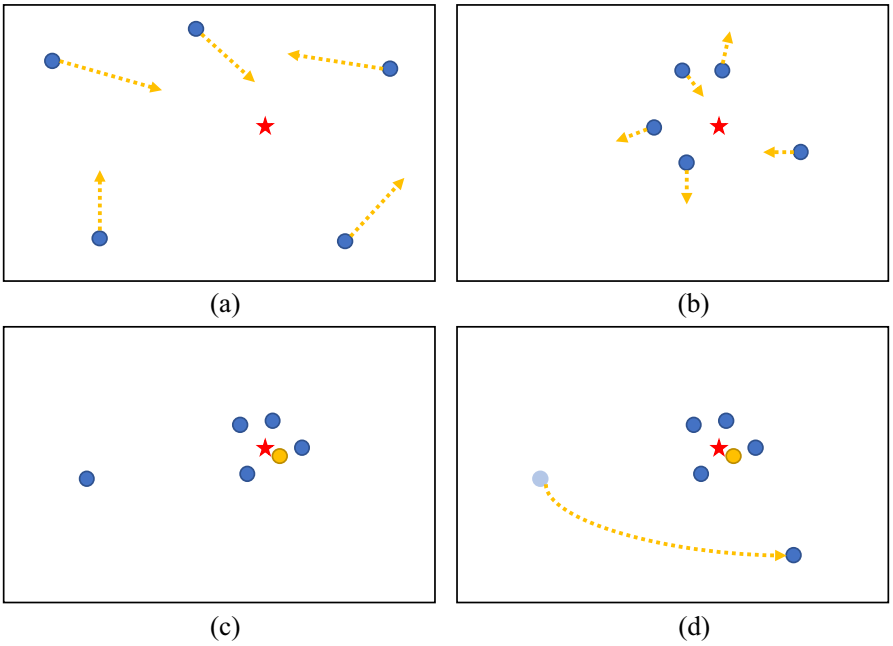

Fig. 3 The simplified search strategies of EMSMO. **a** Search for food. **b** Approach food. **c** Wrap food. **d** Re-initialization

proposed EMSMO: the search for food, approach food, wrap food, and re-initialization. A simple demonstration of these strategies is visualized in Fig. 3. Blue points denote the solutions, orange dotted lines with arrows represent the moving

distances and directions, the red star means the optimal solution, and the orange point is the estimated solution by wrap food operation. The detailed introduction is in the following sections. Noticing that all explanation is under the background of the minimization problem.

2.1 Search for food

Differential-based search strategy has been widely adopted to describe the foraging behaviors of the natural organism [6, 34–37]. Similarly, we adopt the unbiased combination of strategies in Eq. (1) to realize the search for food operator.

$$X_{i+1} = \begin{cases} X_i + a_1(X_{best} - X_i) + a_2(X_{r1} - X_{r2}) \\ X_{best} + a_3(X_{r1} - X_{r2}) \end{cases} \tag{1}$$

where X_i represents the i^{th} solution, X_{best} is the current best solution. $r1$ and $r2$ are two mutually different random values. a_1, a_2, and a_3 are randomly generated from a uniform distribution, which can be calculated by Eq. (2).

$$\begin{aligned} A &= \operatorname{arc}\tanh(1 - t/T) + 0.1 \\ a_i &\sim U(-A, A) \end{aligned} \tag{2}$$

t and T are the current and maximum generation of optimization, respectively, and a_i is a self-adaptive parameter. Supposing the maximum iteration time is 100, and the range of a_i will decrease from 2.75 to 0.1 nonlinearly. Figure 3a demonstrates this search for food operator. Furthermore, the benefit of this differential-based search strategy is that the differential vector between any pairwise solutions constructs a share of the population distribution, which contains a part of the information of distribution. Thus, a differential vector can represent the movement direction of the slime mold.

2.2 Approach food

When the slime mold approaches oatmeal, the bio-oscillator produces a propagating wave and the vein organism becomes veinless or network structures, and the growth speed also degenerates [38]. Therefore, we consider this behavior corresponding to the exploitation operator in algorithm design and formulate it by Eq. (3).

$$X_{i+1} = X_{mean} + R \cdot D \tag{3}$$

where X_{mean} represents the mean location of best-k solutions, and k is randomly sampled from 1 to $N/2$. R is the growth radius of the slime mold, D uniformly sampled from $[-1, 1]$ denotes the extension direction. An example is shown in Fig. 3b.

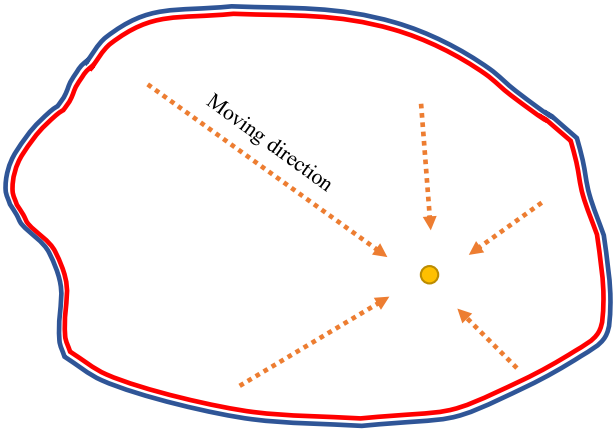

Fig. 4 A demonstration of the wrap food operator in 2-D space

2.3 Wrap food

The slime mold modifies the growth status when reaching the food source and wraps around the surfaces of the oat flakes. Since the contact of the slime mode with the oat flakes is large, it is advantageous for the intake of nutrition to diffuse from the surface of the food source [29]. Figure 3(c) shows a demonstration of the wrap food operation in optimization. To further explain the inspiration for the wrap food operator design, Fig. 4 simulates the food intake process of slime mold.

In the real world, food sources such as oatmeal have a definite volume and cannot be regarded as a point of optimum simply, and the area limited by the red circle in Fig. 4 represents the food source, the blue cycle denotes the slime mold which contacts the food source tightly. The distribution of ingredients in the food source is not uniform so the slime mold has the preference to absorb the food in a certain direction, which corresponds to the concept of fitness in the optimization. Therefore, slime mold can estimate the most nutritious region in the food source area with the biological mechanism, which contributes to the wrap food operator design, and this process can also show a certain level of swarm intelligence of slime mold.

Based on the above explanation, the mathematical model of the wrap food operator is formulated in Eq. (4).

$$X_{i+1} = \sum_{j=1}^{k} w_j X_j, \ \text{s.t.} \ \sum_{j=1}^{k} w_j = 1 \tag{4}$$

where w_j is the corresponding weight to solution X_j, which can be calculated by Eq. (5).

$$w_j = \frac{1/F_j}{\sum_{t=1}^{k} 1/F_t} \tag{5}$$

F_j is the objective value of solution j.

2.4 Re-initialization

In the real-world slime mold, the search direction that fails to find the food will shrink back and leave a trail to avoid re-entry. Similarly, as Fig. 3d shows that a solution is in a poor search direction, which has a probability to be re-initialized. And the re-initialization operation can be described by Eq. (6).

$$X_{i+1}^{d} = LB^{d} + r(UB^{d} - LB^{d}) \tag{6}$$

where LB^d and UB^d represent the lower bound and upper bound of search space in dth dimension, r is a random value from 0 to 1.

3 Our proposal: EMSMO

Section 2 introduces the four LLHs inspired by the foraging behaviors of slime mold, and in this section, we will focus on introducing the high-level component of EMSMO.

The high-level component of EMSMO acts as the brain in the HHs algorithm to manipulate the LLHs. In our proposed EMSMO, we design an improvement-based probabilistic selection function to support the construction of the optimization sequence, which consists of two metrics: the probability of improvement (PI) and the normalized improvement (NI).

Probability of improvement (PI) PI is a metric to describe the probability of a specific search strategy to construct better solutions. Simply, for a search strategy A, PI_A can be calculated by Eq. (7).

$$PI_A = \frac{Times\ of\ A\ achieving\ improved\ solution}{Times\ of\ A\ being\ adopted + \varepsilon} \tag{7}$$

ε is a tiny value to avoid the zero-division error. An example is that if search strategy A is chosen ten times and five improved solutions are obtained, PI_A is approximately equal to 0.5.

Normalized improvement (NI) NI is a quantitative metric to describe the certainty of improvement. Considering that the absolute improvement between the early and late stages of optimization will have a huge difference (i.e., commonly the improvement of solutions in the early stage is large and in the late stage is relatively tiny), it is unfair to compare the improvement directly. To eliminate the effect of dimension,

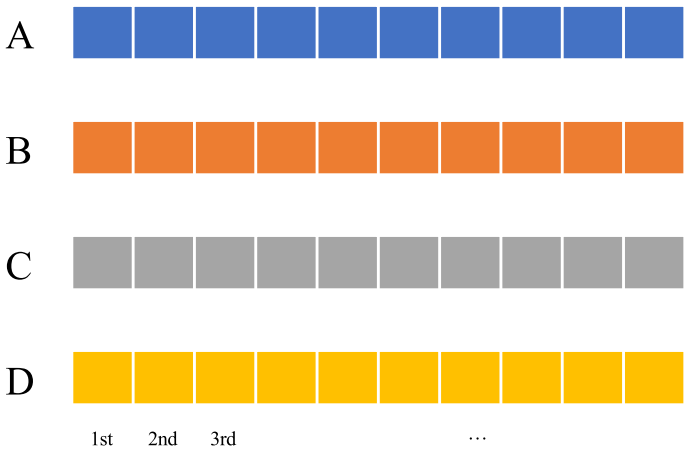

Fig. 5 The structure of score matrix

we normalize the improvement in every generation of optimization. Similarly, the *NI* of strategy *A* can be computed by Eq. (8).

$$NI_A = \frac{\sum_A \max(0, f(O_A) - f(P_A))}{\varepsilon + \sum_{i=1}^{|Pop|} \max(0, f(O_i) - f(P_i))} \tag{8}$$

where $|Pop|$ denotes the population size, $f(\cdot)$ means the objective function. O_i and P_i represent the ith offspring and parent solution, respectively. And the corresponding score of strategy A can be defined in Eq. (9).

$$Score_A = w \cdot PI_A + (1 - w) \cdot NI_A \tag{9}$$

where w is set to 0.5 in our experiments. Figure 5 shows the structure of the score matrix.

A, *B*, *C*, and *D* correspond to our designed four LLHs. The values in a row represent the scores of a certain strategy in the continuous generation of optimization, and values in a column denote the scores of the corresponding strategies in a generation, each value in the score matrix is calculated by Eq. (9). We sum up all previous scores of a certain strategy in the score matrix to contribute to the strategy selection in the next generation. Roulette wheel strategy (RWS) [39] is applied to choose the search strategy for each individual to construct the offspring, and the chosen probability of strategy A can be calculated by Eq. (10).

$$P_A = \frac{Score_A}{\sum_{i=\{A,B,C,D\}} Score_i} \tag{10}$$

In summary, the pseudocode of EMSMO is shown in Algorithm 1.

Algorithm 1 EMSMO

Require: Population size: S, Dimension: D, Maximum iteration: T, Lower and upper bound: LB, UB
Ensure: Optimum: O

```
X ← initialPop(S, D, LB, UB) # Population initialization
O ← best(X)
Initialize score matrix SM
while t = 0 and t < T do
    Sum up the previous scores
    for i = 0 to S do
        Select a search strategy by RWS # Eq. (10)
        Construct offspring OS_i for X_i
        Update X by OS with greedy survival
    end for
    Update the score matrix # Eq. (9)
    O ← best(X)
    t ← t + 1
end while
return O
```

4 Numerical experiments

In this section, we implement a set of experiments to evaluate our proposed EMSMO. Section 4.1 introduces the experiment settings, and Sect. 4.2 shows the experimental results.

4.1 Experiment settings

4.1.1 Benchmark functions

We evaluate the performance of EMSMO on the CEC2013 benchmark functions and three engineering optimization problems. The CEC2013 suite contains 29 functions with various characteristics such as multimodality, asymmetry, and nonseparability. In addition, all functions are shifted and rotated, and the detailed information is listed in Table 1.

We adopt 10-D, 30-D, and 50-D CEC2013 benchmark functions to evaluate EMSMO. Besides, the explanation of three engineering optimization problems is as follows:

Corrugated Bulkhead Design (CBD) This problem aims to minimize the weight of a corrugated bulkhead in a chemical tanker [40], where the decision variables are the width (x_1), depth (x_2), length (x_3), and plate thickness (x_4). The mathematical model of this optimization problem is given as follows.

Table 1 Summary of the CEC2013 Suite: *Uni.* unimodal function, *Multi.* multimodal function, *Comp.* composition function

Fun	Description	Feature	Optimum
f_1	Sphere Function	Uni.	− 1400
f_2	Rotated High Conditioned Elliptic Function		− 1300
f_3	Rotated Bent Cigar Function		− 1200
f_4	Rotated Discus Function		− 1100
f_5	Different Powers Function		− 1000
f_6	Rotated Rosenbrock's Function	Multi.	− 900
f_7	Rotated Schaffers F7 Function		− 800
f_8	Rotated Ackley's Function		− 700
f_9	Rotated Weierstrass Function		− 600
f_{10}	Rotated Griewank's Function		− 500
f_{11}	Rastrigin's Function		− 400
f_{12}	Rotated Rastrigin's Function		− 300
f_{13}	Non-Continuous Rotated Rastrigin's Function		− 200
f_{14}	Schwefel's Function		− 100
f_{15}	Rotated Schwefel's Function		100
f_{16}	Rotated Katsuura Function		200
f_{17}	Lunacek Bi-Rastrigin Function		300
f_{18}	Rotated Lunacek Bi-Rastrigin Function		400
f_{19}	Expanded Griewank's plus Rosenbrock's Function		500
f_{20}	Expanded Scaffer's F6 Function		600
f_{21}	Composition Function 1 ($n = 5$, Rotated)	Comp.	700
f_{22}	Composition Function 2 ($n = 3$, Unrotated)		800
f_{23}	Composition Function 3 ($n = 3$, Rotated)		900
f_{24}	Composition Function 4 ($n = 3$, Rotated)		1000
f_{25}	Composition Function 5 ($n = 3$, Rotated)		1100
f_{26}	Composition Function 6 ($n = 5$, Rotated)		1200
f_{27}	Composition Function 7 ($n = 5$, Rotated)		1300
f_{28}	Composition Function 8 ($n = 5$, Rotated)		1400

Search space: $[-100, 100]^D$

minimize

$$f(X) = \frac{5.885x_4(x_1 + x_3)}{x_1 + \sqrt{|x_3^2 - x_2^2|}} \tag{11}$$

subject to

$$g_1(X) = -x_4x_2(0.4x_1 + \frac{x_3}{6}) + 8.94(x_1 + \sqrt{|x_3^2 - x_2^2|}) \leq 0$$

$$g_2(X) = -x_4x_2^2(0.2x_1 + \frac{x_3}{12}) + 2.2(8.94(x_1 + \sqrt{|x_3^2 - x_2^2|}))^{4/3} \leq 0$$

$$g_3(X) = -x_4 + 0.0156x_1 + 0.15 \leq 0$$

$$g_4(X) = -x_4 + 0.0156x_3 + 0.15 \leq 0$$

$$g_5(X) = -x_3 + x_2 \leq 0$$

where

$$0 \leq x_1, x_2, x_3 \leq 100, 1.05 \leq x_4 \leq 5$$

Tension/Compression Spring Design (TCSD) Tension/Compression Spring Design is described in [40] as an optimization problem, the objective is to minimize the weight of a tension/compression spring under the constraints of minimum deflection, shear stress, surge frequency, and outside diameter limitation. Equation (12) shows the mathematical model of this problem.

$$
\begin{aligned}
&\text{minimize} \\
&f(X) = (x_3 + 2)x_2x_1^2 \\
&\text{subject to} \\
&g_1(X) = 1 - \frac{x_2^3 x_3}{71785x_1^4} \leq 0 \\
&g_2(X) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \leq 0 \\
&g_3(X) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0 \\
&g_4(X) = \frac{x_1 + x_2}{1.5} - 1 \leq 0 \\
&\text{where} \\
&0.05 \leq x_1 \leq 2, 0.25 \leq x_2 \leq 1.3, 2 \leq x_3 \leq 15
\end{aligned}
\tag{12}
$$

Pressure Vessel Design (PVD) This problem is a classic engineering optimization problem that has been adopted to evaluate the performance of many meta-heuristic algorithms [41–43], and the objective is to minimize the cost of the pressure vessel with the thickness of the shell (x_1), the thickness of the head (x_2), the inner radius (x_3), and the length of the cylindrical section of the vessel (x_4), which can be described by Eq. (13).

$$
\begin{aligned}
&\text{minimize} \\
&f(X) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \\
&\text{subject to} \\
&g_1(X) = -x_1 + 0.0193x_3 \leq 0 \\
&g_2(X) = -x_2 + 0.00954x_3 \leq 0 \\
&g_3(X) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \leq 0 \\
&g_4(X) = x_4 - 240 \leq 0 \\
&\text{where} \\
&0 \leq x_1, x_2 \leq 99, 10 \leq x_3, x_4 \leq 200
\end{aligned}
\tag{13}
$$

More details and visualization of these engineering problems can be found in [43].

Table 2 The compared EAs and parameters

Algorithm	Parameters	Value
EMSMO	Growth radius R	2
DE (1995)	Scale factor F	0.7
	Crossover rate Cr	0.9
	Mutation strategy	DE/rand/1/bin
PSO (1995)	Constant b	1
	Random number l	[-1, 1]
	Coefficient vector $\vec{A}$ and $\vec{C}$	[0, 2]
WOA (2016)	Parameter-free	
SLO (2019)	Constant b	1
	Random number l	[-1, 1]
	Coefficient vector $\vec{A}$ and $\vec{C}$	[0, 2]
SMA (2020)	Re-initialization rate z	0.03
AOA (2021)	Exploitation accuracy α	5
	Control parameter μ	0.5
	MOA_{min} and MOA_{max}	0.2 and 0.9

Table 3 The compared hyper-heuristic algorithms and descriptions

HHs alg	Description
Simple random (SR) [18, 47]	Select the subsequent LLHs with uniform probability
Random descent (RD) [48]	Initially select a LLHs randomly and repeat
	It as long as the improvement is achieved
Random permutation (RP) [49]	Randomly construct a sequence of heuristics

4.1.2 Compared methods and parameters

We first evaluate the overview optimization performance of EMSMO. Six classic or advanced EAs including differential evolution (DE) [44], particle swarm optimization (PSO) [45], whale optimization algorithm (WOA) [42], sea lion optimization algorithm (SLO) [45], slime mold algorithm (SMA) [6], and arithmetic optimization algorithm (AOA) [46] are employed to the optimization comparison, and the detailed information is listed in Table 2.

The second experiment is to evaluate the efficiency of our proposed improvement-based probabilistic selection function in optimization sequence construction. Three conventional HHs frameworks are employed as competitor algorithms which are summarized in Table 3. The LLHs for these compared HHs frameworks are consistent with EMSMO.

All compared EAs and CEC2013 benchmark functions are provided by MEALPY [50] and OpFuNu [51]. For all compared algorithms, the population size is 100, the maximum fitness evaluations for CEC2013 benchmark functions and engineering optimization problems are $500 * D$ and 20,000, respectively, and the independent trial run for each optimization method is 30. Holm multiple comparison test [52] is applied to determine the statistical significance between every pair of algorithms, and we use $+$, $\approx$, and $-$ to denote that our proposed EMSMO is significantly better, with no significance, and significantly worse with the compared method. Win-Tie-Loss information is summarized based on the statistical results, and the average rank of each optimization technique is also calculated. The best value is in bold.

4.2 Experimental results

4.2.1 Experimental results on CEC2013

This section shows the experimental and statistical results of EMSMO and compared EAs on CEC2013 benchmark functions. Tables 4, 5, and 6 summarize the optimization results among EMSMO and compared EAs, and convergence curves of representative functions are visualized in Fig. 6. Tables 7, 8, and 9 show the results between EMSMO and other HHs frameworks.

4.2.2 Experimental results on engineering optimization problems

Table 10 summarizes the optimization results of compared EAs on three engineering optimization problems. Given all the engineering problems contain constraints, the original EMSMO and compared EAs cannot deal with the constrained optimization problem. For the sake of simplicity, we equip the EMSMO and compared EAs with a death penalty function [53], which allocates an extremely bad fitness value to infeasible solutions. And if more than 1 of 30 trial runs cannot find a feasible solution at the end of optimization, we manually fill the mean, std, and the worst with *NaN* in Table 10.

5 Discussion

In this section, we first provide the computational complexity analysis of EMSMO, and then the performance of EMSMO based on the experimental results is discussed. Then, we provide some open topics to further improve the performance of EMSMO at the end of this section.

Table 4 Experimental and statistical results between EMSMO and compared EAs on 10-D CEC2013 Suite

Func	DE		PSO		WOA		SLO		SMA		AOA		EMSMO	
	mean	SD	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
f_1	1.44e+03 +	5.91e+02	1.79e+03 +	2.71e+03	− 1.14e+03 +	2.69e+02	− 1.10e+03 +	3.33e+02	− 7.20e+02 +	4.78e+02	6.78e+03 +	1.84e+03	**− 1.40e+03**	**3.12e− 01**
f_2	1.79e+07 +	6.55e+06	3.12e+07 +	3.14e+07	5.34e+06 +	3.56e+06	6.29e+06 +	5.59e+06	1.41e+07 +	7.70e+06	6.66e+07 +	3.16e+07	**2.45e+06**	**1.48e+06**
f_3	1.81e+09 +	5.48e+08	6.73e+09 +	3.22e+09	**7.44e+08** ≈	**1.01e+09**	8.10e+08 ≈	1.04e+09	3.90e+09 +	2.78e+09	1.59e+11 +	6.45e+11	7.79e+08	1.83e+09
f_4	4.07e+04 +	1.34e+04	6.78e+04 +	3.17e+04	3.56e+04 +	1.80e+04	4.00e+04 +	1.65e+04	4.24e+04 +	1.39e+04	8.34e+04 +	1.20e+05	**9.76e+03**	**3.74e+03**
f_5	− 3.46e+02 +	1.48e+02	1.85e+03 +	3.49e+03	− 6.12e+02 ≈	3.40e+02	− 3.43e+02 +	5.19e+02	− 4.80e+02 +	3.64e+02	8.75e+03 +	4.32e+03	**− 7.52e+02**	**2.53e+02**
f_6	− 7.46e+02 +	4.44e+01	− 6.63e+02 +	1.90e+02	− 8.06e+02 +	4.31e+01	− 7.87e+02 +	5.60e+01	− 8.01e+02 ≈	4.72e+01	− 2.16e+02 +	2.40e+02	**− 8.31e+02**	**2.61e+01**
f_7	6.75e+03 ≈	9.13e+02	9.00e+03 +	3.01e+03	8.25e+03 +	3.17e+03	8.69e+03 +	3.06e+03	**− 6.85e+02** −	**3.82e+01**	2.60e+04 +	2.09e+04	5.84e+03	1.78e+03
f_8	**− 6.79e+02** ≈	**1.12e− 01**	− 6.79e+02 ≈	1.11e−01	− 6.79e+02 ≈	1.03e−01	− 6.79e+02 ≈	8.92e−02	− 6.79e+02 ≈	9.65e−02	− 6.79e+02 ≈	1.55e−01	− 6.79e+02	9.01e−02
f_9	− 5.90e+02 +	8.89e−01	− 5.90e+02 +	1.30e+00	− 5.91e+02 +	1.43e+00	− 5.91e+02 +	1.37e+00	− 5.92e+02 +	1.58e+00	− 5.88e+02 +	1.39e+00	**− 5.93e+02**	**1.51e+00**
f_{10}	− 1.29e+02 +	1.05e+02	1.07e+02 +	2.74e+02	− 4.32e+02 +	4.89e+01	− 4.17e+02 +	6.59e+01	− 3.15e+02 +	1.23e+02	5.20e+02 +	3.03e+02	**− 4.86e+02**	**1.37e+01**
f_{11}	− 3.08e+02 +	1.16e+01	− 2.91e+02 +	2.80e+01	− 3.22e+02 +	2.85e+01	− 3.20e+02 +	3.93e+01	− 3.35e+02 +	1.93e+01	− 2.19e+02 +	2.66e+01	**− 3.52e+02**	**1.61e+01**
f_{12}	− 1.92e+02 +	9.97e+00	− 1.79e+02 +	3.29e+01	− 2.12e+02 +	3.52e+01	− 2.10e+02 +	3.70e+01	− 2.32e+02 +	2.06e+01	− 1.30e+02 +	2.39e+01	**− 2.57e+02**	**1.10e+01**
f_{13}	− 1.02e+02 +	9.76e+00	− 1.01e+02 +	2.60e+01	− 1.03e+02 +	2.82e+01	− 1.01e+02 +	3.10e+01	− 1.22e+02 ≈	2.63e+01	− 4.05e+01 +	2.47e+01	**− 1.38e+02**	**1.59e+01**
f_{14}	2.00e+03 +	1.90e+02	2.12e+03 +	2.49e+02	1.25e+03 ≈	4.23e+02	1.23e+03 ≈	2.90e+02	1.39e+03 +	2.74e+02	2.33e+03 +	3.00e+02	**1.03e+03**	**2.31e+02**
f_{15}	2.09e+03 +	2.00e+02	2.06e+03 +	2.44e+02	1.39e+03 +	3.46e+02	1.42e+03 +	3.37e+02	1.61e+03 +	2.49e+02	2.50e+03 +	2.03e+02	**9.68e+02**	**2.64e+02**
f_{16}	2.02e+02 ≈	3.28e−01	2.02e+02 ≈	4.78e−01	**2.01e+02** −	**4.07e-01**	2.01e+02 ≈	4.19e−01	2.02e+02 ≈	5.41e−01	2.03e+02 +	9.11e−01	2.02e+02	4.83e−01
f_{17}	5.20e+02 +	2.91e+01	4.17e+02 +	3.64e+01	4.09e+02 +	3.45e+01	4.07e+02 +	3.19e+01	3.88e+02 +	1.68e+01	4.62e+02 +	2.41e+01	**3.45e+02**	**8.48e+00**

Table 4 (continued)

Func	DE		PSO		WOA		SLO		SMA		AOA		EMSMO	
	mean	SD	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
f_{18}	6.14e+02 +	2.89e+01	6.10e+02 +	8.69e+01	5.75e+02 +	6.19e+01	5.77e+02 +	6.65e+01	**4.84e+02** ≈	**3.00e+01**	8.22e+02 +	7.47e+01	5.04e+02	4.25e+01
f_{19}	9.89e+02 +	4.81e+02	1.94e+03 +	4.77e+03	5.70e+02 +	1.02e+02	7.49e+02 +	7.89e+02	5.17e+02 +	1.64e+01	1.33e+04 +	2.05e+04	**5.07e+02**	**2.69e+00**
f_{20}	6.04e+02 ≈	1.39e−01	6.04e+02 ≈	2.94e−01	6.04e+02 ≈	2.91e−01	6.04e+02 ≈	2.80e−01	6.04e+02 ≈	3.65e−01	6.05e+02 +	2.41e−01	**6.04e+02**	**3.46e-01**
f_{21}	1.43e+03 +	6.45e+01	1.20e+03 +	1.33e+02	1.13e+03 +	7.82e+01	1.16e+03 +	9.63e+01	1.13e+03 +	5.68e+01	1.47e+03 +	8.99e+01	**1.10e+03**	**3.52e+01**
f_{22}	3.01e+03 +	2.31e+02	3.23e+03 +	2.55e+02	2.30e+03 ≈	3.51e+02	2.25e+03 ≈	4.38e+02	2.53e+03 +	3.15e+02	3.60e+03 +	1.78e+02	**2.13e+03**	**3.92e+02**
f_{23}	2.94e+03 +	2.04e+02	2.87e+03 +	3.08e+02	2.42e+03 ≈	3.69e+02	2.59e+03 ≈	2.75e+02	2.76e+03 +	3.28e+02	3.55e+03 +	2.77e+02	**2.37e+03**	**3.07e+02**
f_{24}	1.22e+03 ≈	2.38e+00	1.23e+03 ≈	1.05e+01	1.22e+03 +	2.59e+01	1.22e+03 ≈	2.06e+01	1.23e+03 ≈	3.44e+00	1.25e+03 +	1.32e+01	**1.22e+03**	**2.18e+01**
f_{25}	1.33e+03 ≈	1.41e+00	1.32e+03 ≈	1.34e+01	**1.32e+03** ≈	**2.61e+01**	1.32e+03 ≈	2.31e+01	1.32e+03 ≈	4.12e+00	1.35e+03 +	8.61e+00	1.32e+03	2.14e+01
f_{26}	1.40e+03 +	7.57e−01	1.40e+03 +	1.08e+01	1.40e+03 +	2.95e+01	1.41e+03 +	4.13e+01	1.41e+03 +	5.35e+01	1.44e+03 +	3.93e+01	**1.38e+03**	**2.82e+01**
f_{27}	1.89e+03 +	1.76e+01	1.89e+03 +	3.49e+01	1.90e+03 +	4.61e+01	1.89e+03 +	4.47e+01	1.91e+03 +	6.14e+01	2.01e+03 +	4.68e+01	**1.81e+03**	**4.00e+01**
f_{28}	2.27e+03 +	8.92e+01	2.08e+03 ≈	1.85e+02	1.98e+03 ≈	1.93e+02	**1.97e+03** ≈	**1.60e+02**	2.20e+03 +	1.27e+02	2.43e+03 +	1.39e+02	1.99e+03	1.73e+02
+/≈/−	22/6/0		22/6/0		18/9/1		18/10/0		19/8/1		27/1/0			
Ave. rank	4.82		5.25		2.5		3.32		3.67		6.96		**1.46**	

5.1 Computational complexity analysis of EMSMO

Similar to most EAs, EMSMO has three periods that need to be analyzed: initialization stage, iterative search stage, and parameter update stage. Supposing the population size is N, the dimension of the problem is D, the maximum iteration time is T, the number of the search strategy is a constant so that the complexity is $O(1)$, and the computational complexity of the initialization stage can be simply calculated to $O(ND)$. For one generation of the search, the complexity of each strategy is $O(D)$, and N individuals need to be updated, thus the computational complexity of one generation of the search is $O(ND)$. In the parameter update stage, the score matrix cooperates with the roulette wheel strategy to identify the search strategy for each individual, where the computational complexity is $O(N)$. After the evaluation, the necessary computation for updating the score matrix is $O(N)$ as well. In summary, the time complexity of EMSMO is

$$O(ND) + O(T(N + ND + N)) := O(ND) + O(TND) := O(TND) \tag{14}$$

5.2 Performance analysis on CEC2013 benchmark functions

In the realm of optimization algorithm design, the evaluation of an algorithm's exploitation ability, exploration ability, and the balance struck between these two aspects holds paramount importance. In our analysis, we commence by delving into the exploitation prowess of EMSMO.

Given that the functions f_1 to f_5 within the CEC2013 benchmark are unimodal, the optimization performance across these functions can directly reflect the exploitation capacity of the algorithm. From the comprehensive numerical experiments and statistical analyses, EMSMO outperforms the six compared optimization techniques on most functions. This notable superiority is maintained across the majority of functions, with the exception of 10-D f_3. Moreover, this competitiveness is amplified as the dimension of problems increases, which shows the excellent exploitation capacity of EMSMO even while facing the challenges of higher-dimensional problems.

f_6 to f_{28} are multimodal and composition functions in CEC2013 suite. These functions have multiple optima and complex fitness landscapes, which serve as challenging testbeds for evaluating the performance of algorithms.

Through the experimental and statistical results, our proposed EMSMO remains exceptionally competitive in comparison with the other six optimization techniques. This conclusion is verified through both statistical analyses and the average rank of performance. However, it is important to note that there are instances of slight deterioration in EMSMO's performance, as observed in functions f_{24} and f_{25}.

To explain this phenomenon reasonably, the No Free Lunch Theorem [54] is responsible for this degeneration. This theory states that the average performance of any two stochastic optimization algorithms is equivalent across all possible problems. Consequently, if an algorithm outperforms in a specific class of problems, it must perform poorly in the rest class of problems in order to maintain the same average performance. In light of this theory, we can infer that EMSMO may exhibit

Table 5 Experimental and statistical results between EMSMO and compared EAs on 30-D CEC2013 Suite

Func	DE		PSO		WOA		SLO		SMA		AOA		EMSMO	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
f_1	3.23e+04 +	4.47e+03	3.07e+04 +	1.07e+04	1.58e+04 +	5.45e+03	1.35e+04 +	6.06e+03	2.64e+04 +	6.73e+03	5.33e+04 +	3.58e+03	**-1.38e+03**	**4.34e+00**
f_2	3.91e+08 +	9.57e+07	3.91e+08 +	2.44e+08	3.31e+08 +	1.45e+08	2.11e+08 +	9.72e+07	3.88e+08 +	1.68e+08	1.18e+09 +	5.21e+08	**1.90e+07**	**8.24e+06**
f_3	1.30e+11 +	2.28e+10	6.98e+13 +	1.72e+14	7.62e+15 +	2.85e+16	6.58e+11 +	9.47e+11	3.09e+14 +	1.46e+15	4.62e+18 +	1.43e+19	**1.43e+10**	**1.55e+10**
f_4	1.73e+05 +	2.66e+04	1.98e+05 +	5.37e+04	1.28e+05 +	3.31e+04	1.06e+05 +	2.85e+04	1.45e+05 +	3.10e+04	1.26e+05 +	7.24e+04	**4.97e+04**	**8.14e+03**
f_5	1.20e+04 +	2.30e+03	5.11e+04 +	2.14e+04	3.25e+03 +	1.78e+03	9.18e+03 +	5.01e+03	5.92e+03 +	3.16e+03	9.04e+04 +	3.24e+04	**-6.61e+02**	**1.46e+02**
f_6	2.41e+03 +	6.79e+02	4.30e+03 +	3.26e+03	1.12e+03 +	8.02e+02	6.20e+02 +	7.39e+02	2.51e+03 +	1.39e+03	1.07e+04 +	1.84e+03	**-7.93e+02**	**3.40e+01**
f_7	2.45e+05 ≈	2.27e+04	2.84e+06 +	4.69e+06	4.14e+04 −	1.03e+05	2.45e+07 +	5.24e+07	**6.03e+03** −	**9.65e+03**	7.35e+08 +	8.43e+08	9.06e+05	1.93e+06
f_8	− 6.79e+02 ≈	6.12e⁻02	− 6.79e+02 ≈	5.69e⁻02	− 6.79e+02 +	7.97e⁻02	− 6.79e+02 ≈	5.94e⁻02	− 6.79e+02 +	5.72e⁻02	− 6.79e+02 +	7.52e⁻02	**-6.79e+02**	**5.50e-02**
f_9	− 5.58e+02 +	1.19e+00	− 5.62e+02 +	2.96e+00	− 5.59e+02 +	2.60e+00	− 5.61e+02 +	3.28e+00	− 5.60e+02 +	2.45e+00	− 5.55e+02 +	2.52e+00	**-5.66e+02**	**2.78e+00**
f_{10}	3.73e+03 +	7.49e+02	5.00e+03 +	1.76e+03	2.27e+03 +	6.35e+02	1.48e+03 +	6.52e+02	3.50e+03 +	9.16e+02	7.68e+03 +	1.17e+03	**-4.58e+02**	**1.80e+01**
f_{11}	2.13e+02 +	3.87e+01	3.74e+02 +	1.43e+02	2.53e+02 +	1.16e+02	3.02e+02 +	1.38e+02	2.11e+02 +	1.24e+02	4.76e+02 +	6.87e+01	**1.06e+02**	**6.37e+01**
f_{12}	3.68e+02 +	6.44e+01	4.44e+02 +	1.31e+02	3.68e+02 +	1.03e+02	3.78e+02 +	1.26e+02	2.80e+02 +	1.03e+02	5.44e+02 +	6.72e+01	**1.51e+02**	**8.39e+01**
f_{13}	4.22e+02 +	4.66e+01	5.02e+02 +	1.29e+02	5.06e+02 +	1.23e+02	5.51e+02 +	1.45e+02	4.16e+02 +	1.13e+02	6.21e+02 +	7.69e+01	**3.02e+02**	**5.89e+01**
f_{14}	6.63e+03 +	3.69e+02	8.22e+03 +	4.80e+02	7.52e+03 +	7.15e+02	6.27e+03 +	7.55e+02	7.43e+03 +	5.50e+02	9.40e+03 +	4.44e+02	**5.44e+03**	**8.22e+02**
f_{15}	8.33e+03 +	4.03e+02	8.65e+03 +	3.77e+02	7.83e+03 +	8.02e+02	6.53e+03 +	1.00e+03	7.67e+03 +	7.17e+02	9.20e+03 +	4.94e+02	**5.13e+03**	**6.74e+02**

Table 5 (continued)

Func	DE		PSO		WOA		SLO		SMA		AOA		EMSMO	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
f_{16}	2.03e+02 +	3.98e−01	2.03e+02 +	4.27e−01	2.04e+02 +	8.17e−01	2.04e+02 +	6.71e−01	2.04e+02 +	5.21e−01	2.05e+02 +	9.13e−01	**2.02e+02**	**5.42e-01**
f_{17}	2.16e+03 +	1.60e+02	1.50e+03 +	2.63e+02	1.13e+03 +	1.06e+02	1.23e+03 +	2.34e+02	1.11e+03 +	1.14e+02	1.36e+03 +	5.29e+01	**8.01e+02**	**7.34e+01**
f_{18}	2.03e+03 +	1.97e+02	2.09e+03 +	4.39e+02	1.19e+03 −	9.52e+01	2.27e+03 +	3.83e+02	**1.19e+03** −	**1.28e+02**	2.86e+03 +	1.60e+02	1.43e+03	1.95e+02
f_{19}	3.14e+05 +	1.33e+05	3.28e+05 +	3.88e+05	4.56e+04 +	4.07e+04	5.67e+04 +	7.87e+04	1.89e+05 +	1.59e+05	1.67e+06 +	9.57e+05	**5.42e+02**	**8.40e+00**
f_{20}	**6.14e+02** −	**2.67e-01**	6.15e+02 ≈	4.09e−01	6.15e+02 +	7.46e−02	6.15e+02 +	2.10e−01	6.15e+02 +	5.92e−02	6.15e+02 +	2.91e−08	6.15e+02	3.96e−01
f_{21}	4.60e+03 +	3.04e+02	3.62e+03 +	6.16e+02	2.78e+03 +	1.55e+02	2.65e+03 +	1.61e+02	3.20e+03 +	1.76e+02	3.49e+03 +	2.07e+02	**1.91e+03**	**6.43e+01**
f_{22}	7.72e+03 +	3.61e+02	1.00e+04 +	3.97e+02	8.85e+03 +	5.14e+02	8.02e+03 +	9.16e+02	9.02e+03 +	6.56e+02	1.06e+04 +	4.38e+02	**7.02e+03**	**8.54e+02**
f_{23}	9.39e+03 +	3.13e+02	9.49e+03 +	4.48e+02	9.25e+03 +	6.38e+02	8.23e+03 +	9.17e+02	9.06e+03 +	5.14e+02	1.05e+04 +	5.03e+02	**7.39e+03**	**9.79e+02**
f_{24}	1.31e+03 ≈	4.13e+00	1.31e+03 ≈	7.61e+00	1.36e+03 +	1.86e+01	1.32e+03 +	1.17e+01	1.33e+03 +	1.39e+01	1.42e+03 +	5.77e+01	**1.31e+03**	**1.06e+01**
f_{25}	**1.42e+03** −	**4.40e+00**	1.42e+03 −	8.56e+00	1.48e+03 +	2.24e+01	1.44e+03 ≈	1.38e+01	1.45e+03 +	1.39e+01	1.53e+03 +	1.65e+01	1.44e+03	1.61e+01
f_{26}	1.56e+03 ≈	5.22e+01	1.58e+03 ≈	4.65e+01	1.54e+03 ≈	9.27e+01	1.60e+03 +	3.73e+01	**1.53e+03** ≈	**8.73e+01**	1.68e+03 +	9.40e+01	1.58e+03	5.39e+01
f_{27}	2.75e+03 ≈	3.67e+01	2.71e+03 ≈	7.59e+01	2.80e+03 ≈	1.25e+02	2.86e+03 +	1.13e+02	**2.68e+03** ≈	**6.94e+01**	3.76e+03 +	3.25e+02	2.73e+03	1.20e+02
f_{28}	5.37e+03 +	2.77e+02	6.01e+03 +	6.14e+02	6.92e+03 +	7.44e+02	5.57e+03 +	5.58e+02	6.35e+03 +	6.90e+02	6.36e+03 +	4.51e+02	**4.86e+03**	**3.03e+02**
+/≈/−	21/5/2		22/5/1		24/2/2		26/2/0		24/2/2		28/0/0			
Ave. rank	3.89		4.67		4.03		3.67		3.53		6.67		**1.5**	

reduced performance in problems that possess similar structural characteristics to f_{24} and f_{25}.

f_6 to f_{28} are multimodal and composition functions, and these benchmark functions contain multiple optima and complex fitness landscape characteristics so that they are allowed to evaluate the overall performance of algorithms. In general, our proposed EMSMO is quite competitive with the other six optimization techniques based on statistical analysis and the average rank. However, some deterioration of EMSMO can be observed such as in f_{24} and f_{25}, and we may use the No Free Lunch Theorem [54] to explain this deterioration: No Free Lunch Theorem states that any pair of algorithms have identical averaging performance in all possible problems, and if an algorithm performs better in a certain class of problems, it must degenerate in the other class of problems since it is the only way to achieve the same averaging performance. Therefore, we infer that our proposed EMSMO may not be good at dealing with the problem which has a similar structure to f_{24} and f_{25}.

5.3 Performance analysis on engineering optimization problems

In this study, we utilize three classical engineering optimization problems to comprehensively evaluate EMSMO: corrugated bulkhead design, tension/compression spring design, and pressure vessel design. The experimental results are summarized in Table 10.

An additional aspect we are concerned about is the stability of the algorithm, which is an important metric when addressing real-world optimization problems. The stability can be reflected through metrics like standard deviation and the worst results, as presented in Table 10. Especially noteworthy are the results of the tension/compression spring design problem. In this scenario, the optimizers of WOA and SLO fail to find a feasible solution at least once. On the contrary, our proposed EMSMO consistently finds a feasible solution in all trial runs. This outcome primarily proves the stability of EMSMO in real-world applications.

5.4 Performance analysis with various HHs

To rigorously evaluate the efficiency of our novel improvement-based probabilistic selection function, we employ three baseline approaches: simple random, random descent, and random permutation. These baseline methods compare with EMSMO in comprehensive experiments. The experimental and statistical results practically prove the efficiency of our proposed selection function.

The results demonstrate that the improvement-based probabilistic selection function generally performs comparably or better than the three compared hyper-heuristics (HHs) frameworks in the majority of cases. In fact, it is significantly superior in nearly half of the instances. These results validate the efficacy of our proposal.

Nevertheless, we observe instances where EMSMO performs noticeably worse compared to the competing HHs on specific problems, such as 10-D f_6, 30-D f_{26}, and 50-D f_4. While the exact inner mechanism of the deterioration is not evident, we

Table 6 Experimental and statistical results between EMSMO and compared EAs on 50-D CEC2013 Suite

Func	DE		PSO		WOA		SLO		SMA		AOA		EMSMO	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
f_1	9.17e+04 +	7.36e+03	6.49e+04 +	1.64e+04	4.29e+04 +	7.50e+03	3.37e+04 +	9.52e+03	6.35e+04 +	7.66e+03	7.52e+04 +	3.34e+03	**-1.35e+03**	**1.05e+01**
f_2	1.30e+09 +	1.79e+08	1.57e+09 +	7.90e+08	5.33e+08 +	1.90e+08	3.83e+08 +	1.76e+08	1.00e+09 +	3.63e+08	3.12e+09 +	6.47e+08	**2.78e+07**	**6.21e+06**
f_3	5.79e+11 +	4.27e+11	7.58e+14 +	1.71e+15	5.76e+13 +	1.57e+14	1.19e+13 +	4.42e+13	8.17e+13 +	1.67e+14	1.48e+15 +	4.10e+15	**2.80e+09**	**5.11e+09**
f_4	2.91e+05 +	2.58e+04	3.02e+05 +	9.27e+04	1.58e+05 +	3.69e+04	1.16e+05 +	3.31e+04	2.20e+05 +	4.87e+04	1.72e+05 +	2.84e+05	**7.54e+04**	**1.10e+04**
f_5	6.59e+04 +	8.66e+03	8.69e+04 +	5.23e+04	6.48e+03 +	1.87e+03	1.58e+04 +	8.76e+03	1.11e+04 +	3.24e+03	6.19e+04 +	2.23e+04	**-8.51e+02**	**3.83e+01**
f_6	1.03e+04 +	1.85e+03	6.01e+03 +	2.54e+03	2.49e+03 +	7.47e+02	1.18e+03 +	6.94e+02	5.17e+03 +	1.36e+03	8.68e+03 +	1.37e+03	**-7.64e+02**	**5.09e+01**
f_7	1.00e+06 +	1.38e+05	2.24e+07 +	4.07e+07	2.65e+04 −	7.83e+04	6.35e+07 +	1.11e+08	**3.45e+03** −	**5.49e+03**	4.39e+07 +	4.44e+07	8.54e+05	7.13e+05
f_8	− 6.79e+02 ≈	4.18e⁻02	**-6.79e+02** ≈	**5.32e-02**	− 6.79e+02 +	3.34e⁻02	− 6.79e+02 ≈	5.03e⁻02	− 6.79e+02 +	5.68e⁻02	− 6.79e+02 +	4.55e⁻02	− 6.79e+02	3.57e⁻02
f_9	− 5.24e+02 +	1.42e+00	− 5.30e+02 ≈	5.24e+00	− 5.25e+02 +	3.13e+00	− 5.27e+02 +	3.01e+00	− 5.26e+02 +	2.89e+00	− 5.18e+02 +	2.42e+00	**-5.34e+02**	**4.02e+00**
f_{10}	1.05e+04 +	1.23e+03	9.22e+03 +	2.58e+03	4.72e+03 +	1.22e+03	3.59e+03 +	1.19e+03	7.95e+03 +	1.21e+03	1.20e+04 +	1.24e+03	**-4.18e+02**	**1.99e+01**
f_{11}	1.19e+03 +	1.04e+02	9.85e+02 +	1.94e+02	7.08e+02 +	1.09e+02	9.16e+02 +	2.17e+02	7.00e+02 +	1.02e+02	8.82e+02 +	7.51e+01	**3.51e+02**	**8.81e+01**
f_{12}	1.30e+03 +	1.13e+02	9.83e+02 +	2.51e+02	8.87e+02 +	1.07e+02	9.74e+02 +	1.63e+02	8.58e+02 +	1.06e+02	9.59e+02 +	7.66e+01	**4.78e+02**	**9.58e+01**
f_{13}	1.36e+03 +	1.18e+02	1.09e+03 +	1.90e+02	1.04e+03 +	1.39e+02	1.06e+03 +	1.84e+02	9.62e+02 +	1.14e+02	9.58e+02 +	6.30e+01	**5.78e+02**	**8.04e+01**
f_{14}	1.20e+04 +	4.13e+02	1.54e+04 +	7.50e+02	1.36e+04 +	9.29e+02	1.16e+04 +	1.14e+03	1.44e+04 +	6.99e+02	1.65e+04 +	4.71e+02	**1.03e+04**	**9.55e+02**
f_{15}	1.54e+04 +	2.88e+02	1.57e+04 +	6.13e+02	1.50e+04 +	7.47e+02	1.28e+04 +	1.26e+03	1.47e+04 +	7.66e+02	1.64e+04 +	7.32e+02	**1.05e+04**	**7.30e+02**

Table 6 (continued)

Func	DE		PSO		WOA		SLO		SMA		AOA		EMSMO	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
f_{16}	2.04e+02 +	3.87e−01	2.04e+02 +	3.18e−01	2.05e+02 +	8.17e−01	2.03e+02 +	4.87e−01	2.05e+02 +	6.30e−01	2.06e+02 +	9.20e−01	**2.03e+02**	**6.76e-01**
f_{17}	4.96e+03 +	3.22e+02	2.93e+03 +	5.86e+02	1.70e+03 +	1.21e+02	2.09e+03 +	3.51e+02	1.92e+03 +	1.47e+02	1.87e+03 +	5.74e+01	**1.22e+03**	**8.70e+01**
f_{18}	4.66e+03 +	2.55e+02	3.97e+03 +	7.00e+02	**1.78e+03** −	**1.11e+02**	3.75e+03 +	4.83e+02	2.02e+03 ≈	1.94e+02	3.94e+03 +	1.48e+02	1.98e+03	1.97e+02
f_{19}	4.63e+06 +	1.45e+06	7.48e+05 +	9.94e+05	1.71e+05 +	8.85e+04	5.52e+04 +	7.64e+04	7.64e+05 +	4.94e+05	5.63e+05 +	2.10e+05	**5.72e+02**	**1.13e+01**
f_{20}	**6.24e+02** ≈	**2.66e-01**	6.25e+02 ≈	3.35e−01	6.25e+02 +	2.81e−01	6.25e+02 ≈	2.76e−01	6.25e+02 +	1.68e−01	6.25e+02 +	4.65e−02	6.25e+02	3.98e−01
f_{21}	9.16e+03 +	7.93e+02	6.89e+03 +	1.23e+03	4.86e+03 +	1.24e+02	3.72e+03 +	3.62e+02	5.76e+03 +	3.45e+02	4.93e+03 +	2.48e+02	**1.17e+03**	**2.45e+00**
f_{22}	1.36e+04 ≈	4.27e+02	1.72e+04 +	6.45e+02	1.56e+04 +	7.06e+02	1.48e+04 +	1.35e+03	1.61e+04 +	7.21e+02	1.84e+04 +	5.62e+02	**1.35e+04**	**9.76e+02**
f_{23}	1.63e+04 +	4.24e+02	1.68e+04 +	4.54e+02	1.64e+04 +	8.43e+02	1.48e+04 +	1.08e+03	1.61e+04 +	6.22e+02	1.83e+04 +	4.56e+02	**1.33e+04**	**1.37e+03**
f_{24}	**1.41e+03** −	**6.88e+00**	1.41e+03 −	1.47e+01	1.51e+03 +	7.42e+01	1.45e+03 ≈	2.32e+01	1.47e+03 ≈	2.81e+01	2.12e+03 +	2.79e+02	1.45e+03	3.98e+01
f_{25}	**1.52e+03** −	**5.78e+00**	1.52e+03 −	1.43e+01	1.63e+03 +	3.70e+01	1.55e+03 −	2.47e+01	1.59e+03 ≈	2.42e+01	1.76e+03 +	5.27e+01	1.58e+03	3.33e+01
f_{26}	1.70e+03 +	4.67e+00	1.68e+03 ≈	1.03e+01	1.70e+03 +	1.11e+01	1.70e+03 +	8.55e+00	1.69e+03 ≈	8.73e+00	1.90e+03 +	2.33e+02	**1.68e+03**	**1.28e+01**
f_{27}	3.68e+03 ≈	5.76e+01	**3.64e+03** ≈	**1.23e+02**	3.93e+03 +	1.95e+02	3.93e+03 +	1.93e+02	3.72e+03 ≈	1.02e+02	5.77e+03 +	4.41e+02	3.71e+03	2.43e+02
f_{28}	1.06e+04 +	7.83e+02	1.25e+04 +	2.06e+03	1.08e+04 +	9.20e+02	1.18e+04 +	1.92e+03	1.02e+04 +	9.67e+02	1.13e+04 +	9.35e+02	**8.36e+03**	**6.11e+02**
+/≈/−	22/4/2		21/5/2		26/0/2		24/3/1		22/5/1		28/0/0			
Ave. rank	4.67		4.82		3.89		3.42		3.92		5.82		**1.42**	

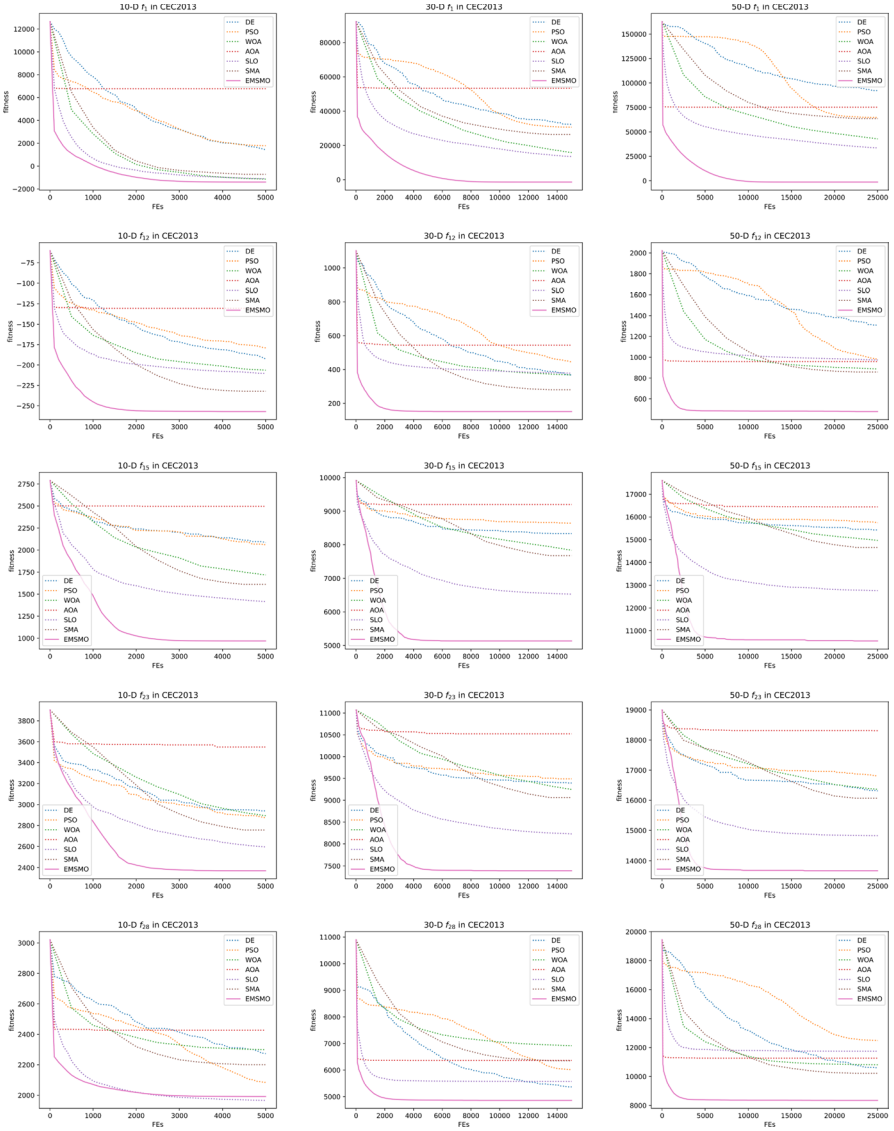

Fig. 6 Convergence curves of representative functions (i.e., f_1: Unimodal; f_{12} and f_{15}: Multimodal; f_{23} and f_{28}: Composite)

Table 7 Experimental and statistical results between EMSMO and compared HHs algorithms on 10-D CEC2013 Suite

Func	SR		RD		RP		EMSMO	
	mean	std	mean	std	mean	std	mean	std
f_1	− 9.08e+02 +	9.12e+02	− 1.17e+03 +	6.76e+02	− 9.31e+02 +	5.99e+02	**-1.40e+03**	**3.12e-01**
f_2	6.54e+06 +	3.69e+06	5.10e+06 +	3.77e+06	7.30e+06 +	3.65e+06	**2.45e+06**	**1.48e+06**
f_3	3.54e+09 +	3.00e+09	2.15e+09 +	3.17e+09	3.74e+09 +	3.31e+09	**7.79e+08**	**1.83e+09**
f_4	9.48e+03 ≈	4.61e+03	**7.41e+03** ≈	**4.80e+03**	1.08e+04 ≈	3.64e+03	9.76e+03	3.74e+03
f_5	− 7.61e+01 +	7.56e+02	− 6.31e+02 ≈	3.99e+02	− 1.80e+02 +	6.04e+02	**-7.52e+02**	**2.53e+02**
f_6	− 8.19e+02 ≈	4.76e+01	**-8.56e+02** −	**2.96e+01**	− 8.01e+02 +	5.69e+01	− 8.31e+02	2.61e+01
f_7	6.51e+03 ≈	2.11e+03	**5.40e+03** ≈	**2.74e+03**	5.82e+03 ≈	1.88e+03	5.84e+03	1.78e+03
f_8	− 6.79e+02 ≈	9.80e−02	− 6.79e+02 ≈	9.64e−02	**-6.79e+02** ≈	**1.08e-01**	− 6.79e+02	9.01e−02
f_9	− 5.93e+02 ≈	1.43e+00	− 5.93e+02 ≈	1.86e+00	− 5.93e+02 ≈	1.31e+00	**-5.93e+02**	**1.51e+00**
f_{10}	− 4.03e+02 +	1.04e+02	− 4.67e+02 ≈	5.89e+01	− 3.95e+02 +	1.05e+02	**-4.86e+02**	**1.37e+01**
f_{11}	− 3.50e+02 ≈	2.28e+01	− 3.48e+02 ≈	2.26e+01	− 3.47e+02 ≈	2.16e+01	**-3.52e+02**	**1.61e+01**
f_{12}	− 2.52e+02 ≈	1.67e+01	− 2.51e+02 ≈	1.88e+01	− 2.50e+02 ≈	2.24e+01	**-2.57e+02**	**1.10e+01**
f_{13}	− 1.37e+02 ≈	2.00e+01	**-1.44e+02** ≈	**2.29e+01**	− 1.39e+02 ≈	2.00e+01	− 1.38e+02	1.59e+01
f_{14}	1.05e+03 ≈	2.37e+02	1.12e+03 ≈	2.75e+02	1.14e+03 ≈	2.78e+02	**1.03e+03**	**2.31e+02**
f_{15}	1.08e+03 ≈	3.00e+02	1.04e+03 ≈	2.96e+02	**9.32e+02** ≈	**3.08e+02**	9.68e+02	2.64e+02
f_{16}	2.01e+02 ≈	5.17e−01	**2.01e+02** ≈	**5.75e-01**	2.02e+02 ≈	3.93e−01	2.02e+02	4.83e−01
f_{17}	3.62e+02 +	2.13e+01	3.64e+02 +	2.98e+01	3.62e+02 +	1.97e+01	**3.45e+02**	**8.48e+00**
f_{18}	5.47e+02 +	6.82e+01	5.29e+02 ≈	7.20e+01	5.51e+02 +	6.22e+01	**5.04e+02**	**4.25e+01**
f_{19}	5.24e+02 +	4.52e+01	5.43e+02 ≈	9.82e+01	5.18e+02 +	1.49e+01	**5.07e+02**	**2.69e+00**
f_{20}	6.04e+02 ≈	3.38e−01	6.04e+02 ≈	3.78e−01	**6.04e+02** ≈	**3.70e-01**	6.04e+02	3.46e−01
f_{21}	1.19e+03 +	9.93e+01	1.12e+03 +	5.87e+01	1.16e+03 +	1.12e+02	**1.10e+03**	**3.52e+01**
f_{22}	2.32e+03 ≈	3.68e+02	2.29e+03 ≈	3.20e+02	2.32e+03 ≈	3.28e+02	**2.13e+03**	**3.92e+02**
f_{23}	2.32e+03 ≈	3.35e+02	2.40e+03 ≈	3.15e+02	**2.30e+03** ≈	**3.38e+02**	2.37e+03	3.07e+02
f_{24}	1.21e+03 ≈	3.11e+01	1.21e+03 ≈	3.16e+01	**1.20e+03** ≈	**3.81e+01**	1.22e+03	2.18e+01

Table 7 (continued)

Func	SR		RD		RP		EMSMO	
	mean	std	mean	std	mean	std	mean	std
f_{25}	1.31e+03 ≈	3.38e+01	1.31e+03 ≈	3.40e+01	**1.31e+03** ≈	**3.57e+01**	1.32e+03	2.14e+01
f_{26}	1.39e+03 +	2.35e+01	1.40e+03 +	1.58e+01	1.39e+03 +	1.88e+01	**1.38e+03**	**2.82e+01**
f_{27}	1.84e+03 ≈	4.81e+01	1.83e+03 ≈	5.07e+01	1.84e+03 ≈	6.21e+01	**1.81e+03**	**4.00e+01**
f_{28}	2.02e+03 ≈	1.74e+02	2.01e+03 ≈	1.88e+02	**1.98e+03** ≈	**1.90e+02**	1.99e+03	1.73e+02
+/≈/−	10/18/0		6/21/1		11/17/0			
Ave. rank	3.07		2.35		2.71		**1.85**	

infer that these problems might present deceptive information for the score matrix update, which may mislead the construction of the optimization sequence.

5.5 Potential and future topics

Through the above analyses, we have known that our proposed EMSMO has achieved satisfactory performance and has great potential. Finally, we list a few open topics for further improving the performance of EMSMO.

5.5.1 Hybridizing the benefits of the other search strategies

Many EAs have proposed various efficient operators to realize optimization: the differential-based operator in DE, the crossover and mutation strategy in GA, the velocity-location-based operator in PSO, the local search scheme, surrogate-assisted estimation, and so on. Paper [55] concludes ten search strategies of well-known meta-heuristics in the literature, and we can take advantage of the hybridization of these operators to EMSMO, which is expected to generate solutions with various characteristics and strengthen the population diversity in the optimization process.

5.5.2 The scalability to various optimization

In this paper, we proposed EMSMO for single-objective continuous optimization, and this HHs framework has strong scalability to extend to various optimization problems: The online learning mechanism of high-level component design allows EMSMO can be applied to computationally expensive optimization and complex real-world problems. If we adjust the search strategy and add the typical operators of combinatorial optimization like *k*-opt and ruin-and-recreate methods, it can deal with combinatorial optimization automatically. Meanwhile, the modification of the acceptance principle for solutions endows the ability of EMSMO to solve

Table 8 Experimental and statistical results between EMSMO and compared HHs algorithms on 30-D CEC2013 Suite

Func	SR		RD		RP		EMSMO	
	mean	std	mean	std	mean	std	mean	std
f_1	4.00e+03 +	3.23e+03	6.06e+03 +	9.53e+03	4.83e+03 +	3.76e+03	**-1.38e+03**	**4.34e+00**
f_2	5.75e+07 +	2.45e+07	6.96e+07 +	8.40e+07	6.08e+07 +	3.02e+07	**1.90e+07**	**8.24e+06**
f_3	1.96e+11 +	5.04e+11	4.75e+14 +	2.42e+15	8.34e+10 +	5.97e+10	**1.43e+10**	**1.55e+10**
f_4	**4.81e+04** ≈	**7.39e+03**	5.16e+04 ≈	8.27e+03	4.95e+04 ≈	8.53e+03	4.97e+04	8.14e+03
f_5	2.36e+03 +	1.52e+03	2.23e+03 +	4.45e+03	2.44e+03 +	2.08e+03	**-6.61e+02**	**1.46e+02**
f_6	− 3.39e+02 +	2.85e+02	− 2.03e+01 +	1.50e+03	− 3.10e+02 +	3.82e+02	**-7.93e+02**	**3.40e+01**
f_7	1.74e+06 +	2.09e+06	9.15e+05 ≈	1.34e+06	2.46e+06 +	2.82e+06	**9.06e+05**	**1.93e+06**
f_8	− 6.79e+02 ≈	5.91e−02	− 6.79e+02 ≈	5.69e−02	− 6.79e+02 ≈	5.48e−02	**-6.79e+02**	**5.50e-02**
f_9	− 5.65e+02 ≈	3.25e+00	− 5.66e+02 ≈	3.21e+00	− 5.65e+02 ≈	2.89e+00	**-5.66e+02**	**2.78e+00**
f_{10}	4.67e+02 +	3.62e+02	1.86e+02 +	9.54e+02	5.28e+02 +	5.02e+02	**-4.58e+02**	**1.80e+01**
f_{11}	1.00e+02 ≈	8.85e+01	**8.13e+01** ≈	**1.23e+02**	8.78e+01 ≈	7.04e+01	1.06e+02	6.37e+01
f_{12}	2.00e+02 ≈	8.65e+01	1.88e+02 ≈	9.60e+01	1.95e+02 ≈	9.85e+01	**1.51e+02**	**8.39e+01**
f_{13}	3.35e+02 ≈	9.33e+01	3.13e+02 ≈	9.22e+01	3.32e+02 ≈	8.06e+01	**3.02e+02**	**5.89e+01**
f_{14}	5.74e+03 ≈	5.16e+02	**5.29e+03** ≈	**9.21e+02**	5.58e+03 ≈	5.69e+02	5.44e+03	8.22e+02
f_{15}	5.21e+03 ≈	6.09e+02	5.35e+03 ≈	9.92e+02	5.22e+03 ≈	6.63e+02	**5.13e+03**	**6.74e+02**
f_{16}	2.02e+02 ≈	5.96e−01	2.02e+02 ≈	6.19e−01	2.02e+02 ≈	5.01e−01	**2.02e+02**	**5.42e-01**
f_{17}	8.67e+02 ≈	1.13e+02	8.30e+02 ≈	1.15e+02	8.74e+02 ≈	1.16e+02	**8.01e+02**	**7.34e+01**
f_{18}	1.78e+03 +	3.38e+02	1.51e+03 ≈	2.65e+02	1.86e+03 +	2.94e+02	**1.43e+03**	**1.95e+02**
f_{19}	1.91e+03 +	2.89e+03	1.92e+04 +	4.66e+04	1.90e+03 +	2.13e+03	**5.42e+02**	**8.40e+00**
f_{20}	6.14e+02 ≈	1.28e−01	**6.14e+02** ≈	**1.81e-01**	6.14e+02 ≈	1.72e−01	6.15e+02	3.96e−01
f_{21}	2.15e+03 +	1.12e+02	2.30e+03 +	4.18e+02	2.17e+03 +	1.14e+02	**1.91e+03**	**6.43e+01**
f_{22}	7.32e+03 ≈	7.28e+02	7.19e+03 ≈	1.01e+03	7.51e+03 ≈	1.03e+03	**7.02e+03**	**8.54e+02**
f_{23}	**7.24e+03** ≈	**8.16e+02**	7.36e+03 ≈	9.56e+02	7.48e+03 ≈	9.09e+02	7.39e+03	9.79e+02
f_{24}	1.32e+03 ≈	1.59e+01	1.32e+03 ≈	2.67e+01	1.32e+03 +	1.71e+01	**1.31e+03**	**1.06e+01**

Table 8 (continued)

Func	SR		RD		RP		EMSMO	
	mean	std	mean	std	mean	std	mean	std
f_{25}	1.45e+03 +	1.74e+01	1.46e+03 +	2.67e+01	1.45e+03 +	1.72e+01	**1.44e+03**	**1.61e+01**
f_{26}	1.53e+03 ≈	8.74e+01	1.57e+03 ≈	7.40e+01	**1.51e+03** −	**8.74e+01**	1.58e+03	5.39e+01
f_{27}	**2.71e+03** ≈	**2.83e+02**	2.75e+03 ≈	2.77e+02	2.86e+03 +	1.75e+02	2.73e+03	1.20e+02
f_{28}	4.84e+03 ≈	4.49e+02	**4.77e+03** ≈	**4.41e+02**	4.92e+03 ≈	3.45e+02	4.86e+03	3.03e+02
+/≈/−	11/17/0		9/19/0		13/14/1			
Ave. rank	2.60		2.60		3.17		**1.60**	

Table 9 Experimental and statistical results between EMSMO and compared HHs algorithms on 50-D CEC2013 Suite

Func	SR		RD		RP		EMSMO	
	mean	std	mean	std	mean	std	mean	std
f_1	1.68e+03 +	2.05e+03	1.74e+04 +	2.01e+04	1.02e+03 +	1.83e+03	**-1.35e+03**	**1.05e+01**
f_2	6.84e+07 +	2.38e+07	2.04e+08 +	1.58e+08	7.09e+07 +	3.38e+07	**2.78e+07**	**6.21e+06**
f_3	5.54e+10 +	2.14e+10	9.30e+11 +	3.77e+12	4.94e+10 +	2.18e+10	**2.80e+09**	**5.11e+09**
f_4	**6.10e+04** −	**1.06e+04**	7.35e+04 ≈	1.22e+04	6.14e+04 −	9.44e+03	7.54e+04	1.10e+04
f_5	6.68e+02 +	5.91e+02	3.75e+03 +	5.80e+03	5.92e+02 +	5.35e+02	**-8.51e+02**	**3.83e+01**
f_6	− 5.69e+02 +	5.43e+01	1.59e+02 +	1.18e+03	− 5.80e+02 +	6.93e+01	**-7.64e+02**	**5.09e+01**
f_7	9.95e+05 +	4.03e+05	8.62e+05 ≈	4.32e+05	9.68e+05 ≈	4.39e+05	**8.54e+05**	**7.13e+05**
f_8	− 6.79e+02 ≈	3.82e−02	**-6.79e+02** ≈	**4.61e-02**	− 6.79e+02 ≈	2.51e−02	− 6.79e+02	3.57e−02
f_9	− 5.34e+02 ≈	3.69e+00	**-5.34e+02** ≈	**4.68e+00**	− 5.33e+02 ≈	3.30e+00	− 5.34e+02	4.02e+00
f_{10}	1.69e+02 +	1.98e+02	1.45e+03 +	2.32e+03	1.70e+02 +	2.44e+02	**-4.18e+02**	**1.99e+01**
f_{11}	3.72e+02 ≈	1.16e+02	3.81e+02 ≈	1.27e+02	3.65e+02 ≈	9.22e+01	**3.51e+02**	**8.81e+01**
f_{12}	5.21e+02 ≈	9.35e+01	5.44e+02 ≈	1.17e+02	4.96e+02 ≈	1.11e+02	**4.78e+02**	**9.58e+01**
f_{13}	6.04e+02 ≈	9.30e+01	6.11e+02 ≈	1.11e+02	6.00e+02 ≈	7.70e+01	**5.78e+02**	**8.04e+01**
f_{14}	1.03e+04 ≈	9.97e+02	**1.02e+04** ≈	**1.58e+03**	1.08e+04 ≈	8.47e+02	1.03e+04	9.55e+02
f_{15}	1.10e+04 ≈	8.63e+02	**1.03e+04** ≈	**1.26e+03**	1.14e+04 +	9.65e+02	1.05e+04	7.30e+02
f_{16}	2.03e+02 ≈	4.92e−01	2.03e+02 ≈	5.30e−01	2.03e+02 +	4.98e−01	**2.03e+02**	**6.76e-01**

Table 9 (continued)

Func	SR		RD		RP		EMSMO	
	mean	std	mean	std	mean	std	mean	std
f_{17}	1.32e+03 +	1.36e+02	1.35e+03 +	1.90e+02	1.30e+03 +	1.31e+02	**1.22e+03**	**8.70e+01**
f_{18}	2.60e+03 +	3.58e+02	2.58e+03 +	3.46e+02	2.59e+03 +	3.45e+02	**1.98e+03**	**1.97e+02**
f_{19}	6.20e+02 +	5.41e+01	7.68e+03 +	2.83e+04	6.14e+02 +	3.82e+01	**5.72e+02**	**1.13e+01**
f_{20}	6.24e+02 ≈	2.08e−01	6.24e+02 ≈	3.09e−01	**6.24e+02** ≈	**3.84e-01**	6.24e+02	3.98e−01
f_{21}	1.70e+03 +	2.33e+02	2.38e+03 +	1.05e+03	1.79e+03 +	3.51e+02	**1.17e+03**	**2.45e+00**
f_{22}	1.39e+04 ≈	1.14e+03	**1.32e+04** ≈	**1.47e+03**	1.35e+04 ≈	1.15e+03	1.35e+04	9.76e+02
f_{23}	1.36e+04 ≈	1.26e+03	**1.33e+04** ≈	**1.26e+03**	1.38e+04 ≈	1.14e+03	1.33e+04	1.37e+03
f_{24}	1.47e+03 ≈	5.20e+01	1.48e+03 ≈	9.66e+01	1.46e+03 ≈	3.50e+01	**1.45e+03**	**3.98e+01**
f_{25}	1.61e+03 ≈	3.15e+01	**1.57e+03** ≈	**5.56e+01**	1.61e+03 +	3.35e+01	1.58e+03	3.33e+01
f_{26}	1.68e+03 ≈	1.43e+01	**1.67e+03** ≈	**5.16e+01**	1.68e+03 ≈	5.24e+01	1.68e+03	1.28e+01
f_{27}	3.83e+03 ≈	1.86e+02	3.75e+03 ≈	3.12e+02	3.85e+03 ≈	1.80e+02	**3.71e+03**	**2.43e+02**
f_{28}	8.37e+03 ≈	5.39e+02	8.33e+03 ≈	6.26e+02	**8.23e+03** ≈	**6.58e+02**	8.36e+03	6.11e+02
+/≈/−	11/16/1		10/18/0		13/14/1			
Ave. rank	2.92		2.67		2.67		**1.71**	

Table 10 Experimental results on engineering optimization problems

Problem		DE	PSO	WOA	SLO	SMA	AOA	EMSMO
CBD	Mean	**6.8492e+00**	7.8896e+00	7.5919e+00	7.4157e+00	8.2067e+00	8.0645e+00	7.0027e+00
	Std	**1.6473e-03**	5.6763e−01	6.7792e−01	5.2653e−01	1.4708e+00	1.3656e+00	1.9991e−01
	Best	**6.8449e+00**	7.1050e+00	6.8884e+00	6.8511e+00	6.8462e+00	6.8923e+00	6.8529e+00
	Worst	**6.8523e+00**	9.2709e+00	9.5588e+00	9.0316e+00	1.1335e+01	1.0411e+01	7.7317e+00
TCSD	Mean	2.7368e−02	1.6177e−02	*NaN*	*NaN*	1.3449e−02	2.2017e−02	**1.2790e-02**
	Std	2.3459e−02	3.4204e−03	*NaN*	*NaN*	9.8589e−04	8.5871e−03	**9.1033e-05**
	Best	1.2681e−02	1.2863e−02	**1.2666e-02**	1.2671e−02	1.2710e−02	1.3062e−02	1.2677e−02
	Worst	9.3420e−02	2.5602e−02	*NaN*	*NaN*	1.6954e−02	3.9296e−02	**1.3121e-02**
PVD	Mean	7.5978e+03	4.1833e+04	7.3184e+04	3.5210e+04	6.3324e+03	7.8339e+03	**5.9900e+03**
	Std	6.5189e+03	2.2271e+04	8.5217e+04	4.8743e+04	4.1312e+02	1.7404e+03	**3.2718e+01**
	Best	**5.8862e+03**	1.0233e+04	6.2511e+03	6.3547e+03	5.9091e+03	6.1582e+03	5.9102e+03
	Worst	4.2620e+04	9.5789e+04	3.4844e+05	2.1892e+05	7.3116e+03	1.5397e+04	**6.0640e+03**

multi-objective and constraint problems. In summary, we will focus on extending EMSMO for various optimization problems in future research.

6 Conclusion

This paper proposes a novel hyper-heuristic algorithm named evolutionary multi-mode slime mold optimization (EMSMO). Inspired by the slime mold foraging behaviors, we design four search schemes as the low-level heuristics. An improvement-based probabilistic selection function is designed as the high-level component of EMSMO. To evaluate the performance of EMSMO, we implement comprehensive comparative experiments with six classic or advanced EAs and three HHs algorithms. Experimental and statistical results prove the competitiveness of EMSMO in practice.

At the end of this paper, we list some open topics. In future research, we will focus on improving the performance of EMSMO and extending EMSMO to various optimization problems.

Acknowledgements This work was supported by JSPS KAKENHI Grant Numbers JP20K11967 and 21A402 and JST SPRING, Grant Number JPMJSP2119.

Author contributions RZ was involved in conceptualization, methodology, investigation, writing—original draft, writing—review and editing, and funding acquisition. EZ helped in investigation, methodology, formal analysis, and writing—review and editing. MM contributed to writing—review and editing and project administration.

Data availibility The research code can be downloaded from https://github.com/RuiZhong961230/EMSMO.

Declarations

Conflict of 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.

References

1. Gao K, Cao Z, Zhang L, Chen Z, Han Y, Pan Q (2019) A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems. IEEE/CAA J Autom Sin 6(4):904–916. https://doi.org/10.1109/JAS.2019.1911540
2. Mouchlis VD, Afantitis A, Serra Fratello M, Papadiamantis AG, Aidinis V, Lynch I, Greco D, Melagraki G (2021) Advances in de novo drug design: from conventional to machine learning methods. Int J Mol Sci. https://doi.org/10.3390/ijms22041676
3. Zhong R, Zhang E, Munetomo M (2023) Cooperative coevolutionary surrogate ensemble-assisted differential evolution with efficient dual differential grouping for large-scale expensive optimization problems. Complex Intell Syst. https://doi.org/10.1007/s40747-023-01262-6
4. Zhong R, Zhang E, Munetomo M (2023) Cooperative coevolutionary differential evolution with linkage measurement minimization for large-scale optimization problems in noisy environments. Complex Intell Syst 9:4439–4456. https://doi.org/10.1007/s40747-022-00957-6
5. Zhong R, Peng F, Yu J, Munetomo M (2024) Q-learning based vegetation evolution for numerical optimization and wireless sensor network coverage optimization. Alex Eng J 87:148–163. https://doi.org/10.1016/j.aej.2023.12.028

6. Li S, Chen H, Wang M, Heidari AA, Mirjalili S (2020) Slime mould algorithm: a new method for stochastic optimization. Future Gener Comput Syst 111:300–323. https://doi.org/10.1016/j.future.2020.03.055
7. Chen Z, Francis A, Li S, Liao B, Xiao D, Ha TT, Li J, Ding L, Cao X (2022) Egret swarm optimization algorithm: an evolutionary computation approach for model free optimization. Biomimetics. https://doi.org/10.3390/biomimetics7040144
8. Yu J (2022) Vegetation evolution: an optimization algorithm inspired by the life cycle of plants. Int J Comput Intell Appl. https://doi.org/10.1142/S1469026822500109
9. Sörensen K, Arnold F, Palhazi Cuervo D (2017) A critical analysis of the "improved clarke and wright savings algorithm''. Int Trans Oper Res 2:6. https://doi.org/10.1111/itor.12443
10. Camacho C, Dorigo M, Stützle T (2019) The intelligent water drops algorithm: why it cannot be considered a novel algorithm: a brief discussion on the use of metaphors in optimization. Swarm Intell. https://doi.org/10.1007/s11721-019-00165-y
11. Tzanetos A, Dounias GD (2020) Nature inspired optimization algorithms or simply variations of metaheuristics? Artif Intell Rev 54:1841–1862. https://doi.org/10.1007/s10462-020-09893-8
12. Aranha C, Villalón C, Campelo F, Dorigo M, Ruiz R, Sevaux M, Sörensen K, Stützle T (2021) Metaphor-based metaheuristics, a call for action: the elephant in the room. Swarm Intell 16:1–6. https://doi.org/10.1007/s11721-021-00202-9
13. Weyland D (2010) A rigorous analysis of the harmony search algorithm: how the research community can be misled by a "novel'' methodology. Int J Appl Metaheuristic Comput 1(2):50–60. https://doi.org/10.4018/jamc.2010040104
14. Sörensen K (2013) Metaheuristics—the metaphor exposed. Int Trans Oper Res. https://doi.org/10.1111/itor.12001
15. Fisher H (1963) Probabilistic learning combinations of local job-shop scheduling rules. Ind Sched 225–251
16. Cowling PI, Kendall G, Soubeiga E (2000) A hyperheuristic approach to scheduling a sales summit. In: International Conference on the Practice and Theory of Automated Timetabling
17. Dowsland KA (1988) Off-the-peg or made-to-measure? timetabling and scheduling with SA and TS. In: Burke E, Carter M (eds) Practice and theory of automated timetabling II. Springer, Berlin, pp 37–52. https://doi.org/10.1007/BFb0055880
18. Cowling P, Kendall G, Soubeiga E (2001) A hyperheuristic approach to scheduling a sales summit. In: Burke E, Erben W (eds) Practice and theory of automated timetabling III. Springer, Berlin, pp 176–190. https://doi.org/10.1007/3-540-44629-X_11
19. Burke EK, Gendreau M, Hyde M, Kendall G, Ochoa G, Özcan E, Qu R (2013) Hyper-heuristics: a survey of the state of the art. J Oper Res Soc 64(12):1695–1724. https://doi.org/10.1057/jors.2013.71
20. Choong SS, Wong L-P, Lim CP (2018) Automatic design of hyper-heuristic based on reinforcement learning. Inf Sci 436–437:89–107. https://doi.org/10.1016/j.ins.2018.01.005
21. Burke EK, Kendall G, Soubeiga E (2003) A tabu-search hyperheuristic for timetabling and rostering. J Heurist 9(6):451–470. https://doi.org/10.1023/B:HEUR.0000012446.94732.b6
22. Terashima-Marín H, Ortiz-Bayliss JC, Ross P, Valenzuela-Rendón M (2008) Hyper-heuristics for the dynamic variable ordering in constraint satisfaction problems. In: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation. GECCO '08. Association for Computing Machinery, New York, pp 571–578. https://doi.org/10.1145/1389095.1389206
23. Burke E, Kendall G, Mısır M, Özcan E (2012) Monte Carlo hyper-heuristics for examination timetabling. Ann Oper Res 196:73–90. https://doi.org/10.1007/s10479-010-0782-2
24. Lin J, Wang Z-J, Li X (2017) A backtracking search hyper-heuristic for the distributed assembly flow-shop scheduling problem. Swarm Evol Comput 36:124–135. https://doi.org/10.1016/j.swevo.2017.04.007
25. Zhao F, Di S, Cao J, Tang J, Jonrinaldi R (2021) A novel cooperative multi-stage hyper-heuristic for combination optimization problems. Complex Syst Model Simul 1(2):91–108. https://doi.org/10.23919/CSMS.2021.0010
26. Lin J, Li Y-Y, Song H-B (2022) Semiconductor final testing scheduling using q-learning based hyper-heuristic. Expert Syst Appl 187:115978. https://doi.org/10.1016/j.eswa.2021.115978
27. Zhong R, Yu J, Chao Z, Munetomo M (2023) Surrogate ensemble-assisted hyper-heuristic algorithm for expensive optimization problems. Int J Comput Intell Syst. https://doi.org/10.1007/s44196-023-00346-y

28. Liang J, Qu B, Suganthan P, Hernández-Díaz A (2013) Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization. Technical report 201212, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China
29. Nakagaki T, Yamada H, Toth A (2000) Maze-solving by an amoeboid organism. Nature 407:470. https://doi.org/10.1038/35035159
30. Tero A, Takagi S, Saigusa T, Ito K, Bebber DP, Fricker MD, Yumiki K, Kobayashi R, Nakagaki T (2010) Rules for biologically inspired adaptive network design. Science 327(5964):439–442. https://doi.org/10.1126/science.1177894
31. Adamatzky A, Akl S, Alonso-Sanz R, van Dessel W, Ibrahim Z, Ilachinski A, Jones J, Kayem AVDM, Martínez GJ, de Oliveira P, Prokopenko M, Schubert T, Sloot P, Strano E, Yang X-S (2013) Are motorways rational from slime mould's point of view? Int J Parallel Emerg Distrib Syst 28(3):230–248. https://doi.org/10.1080/17445760.2012.685884
32. Boussard A, Fessel A, Oettmeier C, Briard L, Döbereiner H-G, Dussutour A (2021) Adaptive behaviour and learning in slime moulds: the role of oscillations. Philos Trans R Soc B Biol Sci 376(1820):20190757. https://doi.org/10.1098/rstb.2019.0757
33. Ternois M, Mougon M, Flahaut E, Dussutour A (2021) Slime molds response to carbon nanotubes exposure: from internalization to behavior. Nanotoxicology 15(4):511–526. https://doi.org/10.1080/17435390.2021.1894615
34. Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H (2019) Harris hawks optimization: algorithm and applications. Future Gener Comput Syst 97:849–872. https://doi.org/10.1016/j.future.2019.02.028
35. Jia H, Peng X, Lang C (2021) Remora optimization algorithm. Expert Syst Appl 185:115665. https://doi.org/10.1016/j.eswa.2021.115665
36. Seyyedabbasi A, Kiani F (2022) Sand cat swarm optimization: a nature-inspired algorithm to solve global optimization problems. Eng Comput. https://doi.org/10.1007/s00366-022-01604-x
37. Chopra N, Mohsin Ansari M (2022) Golden jackal optimization: a novel nature-inspired optimizer for engineering applications. Expert Syst Appl 198:116924. https://doi.org/10.1016/j.eswa.2022.116924
38. Nakagaki T, Yamada H, Ueda T (2000) Interaction between cell shape and contraction pattern in the physarum plasmodium. Biophys Chem 84:195–204. https://doi.org/10.1016/S0301-4622(00)00108-3
39. Lipowski A, Lipowska D (2012) Roulette-wheel selection via stochastic acceptance. Physica A 391(6):2193–2196. https://doi.org/10.1016/j.physa.2011.12.004
40. In: Arora JS (ed) Introduction to optimum design, 4th edn. Academic Press, Boston (2017). https://doi.org/10.1016/B978-0-12-800806-5.00024-X
41. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007
42. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008
43. Bayzidi H, Talatahari S, Saraee M, Lamarche C-P (2021) Social network search for solving engineering optimization problems. Comput Intell Neurosci 2021:1–32. https://doi.org/10.1155/2021/8548639
44. Storn R (1996) On the usage of differential evolution for function optimization. In: Proceedings of North American Fuzzy Information Processing, pp 519–523. https://doi.org/10.1109/NAFIPS.1996.534789
45. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of ICNN'95 - International Conference on Neural Networks, vol 4, pp 1942–19484. https://doi.org/10.1109/ICNN.1995.488968
46. Abualigah L, Diabat A, Mirjalili S, Abd Elaziz M, Gandomi AH (2021) The arithmetic optimization algorithm. Comput Methods Appl Mech Eng 376:113609. https://doi.org/10.1016/j.cma.2020.113609
47. Jackson WG, Özcan E, Drake JH (2013) Late acceptance-based selection hyper-heuristics for cross-domain heuristic search. In: 2013 13th UK Workshop on Computational Intelligence (UKCI), pp 228–235. https://doi.org/10.1109/UKCI.2013.6651310
48. Özcan E, Kheiri A (2012) A hyper-heuristic based on random gradient, greedy and dominance. In: Computer and information sciences II. Springer, London, pp 557–563. https://doi.org/10.1007/978-1-4471-2155-8_71

49. Cowling P, Kendall G, Soubeiga E (2002) Hyperheuristics: a tool for rapid prototyping in scheduling and optimisation. In: Applications of evolutionary computing. Springer, Berlin, pp 1–10. https://doi.org/10.1007/3-540-46004-7_1
50. Van Thieu N, Mirjalili S (2023) MEALPY: an open-source library for latest meta-heuristic algorithms in python. J Syst Archit 139:102871. https://doi.org/10.1016/j.sysarc.2023.102871
51. Nguyen T (2020) A framework of Optimization Functions using Numpy (OpFuNu) for optimization problems. Zenodo. https://doi.org/10.5281/zenodo.3620960
52. Holm S (1979) A simple sequentially rejective multiple test procedure. Scand J Stat 6(2):65–70
53. Coello Coello CA (2002) Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Comput Methods Appl Mech Eng 191(11):1245–1287. https://doi.org/10.1016/S0045-7825(01)00323-1
54. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82. https://doi.org/10.1109/4235.585893
55. Cruz-Duarte JM, Amaya I, Ortiz-Bayliss JC, Conant-Pablos SE, Terashima-Marín H (2020) A primary study on hyper-heuristics to customise metaheuristics for continuous optimisation. In: 2020 IEEE Congress on Evolutionary Computation (CEC), pp 1–8 . https://doi.org/10.1109/CEC48606.2020.9185591

Authors and Affiliations

Rui Zhong[1] · Enzhi Zhang[1] · Masaharu Munetomo[2]

✉ Rui Zhong
rui.zhong.u5@elms.hokudai.ac.jp

Enzhi Zhang
enzhi.zhang.n6@elms.hokudai.ac.jp

Masaharu Munetomo
munetomo@iic.hokudai.ac.jp

[1] Graduate School of Information Science and Technology, Hokkaido University, Sapporo, Japan

[2] Information Initiative Center, Hokkaido University, Sapporo, Japan

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH ("Springer Nature").

Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users ("Users"), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use ("Terms"). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.

These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.

We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.

While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.

These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.

To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.

Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.

If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com