# RIME: A physics-based optimization

Hang Su [1], Dong Zhao [1,*], Ali Asghar Heidari [2], Lei Liu [3], Xiaoqin Zhang [4], Majdi Mafarja[5], Huiling Chen [4,*]

[1] College of Computer Science and Technology, Changchun Normal University, Changchun, Jilin 130032, China
(suhang_v@163.com, zd-hy@163.com)

[2] School of Surveying and Geospatial Engineering, College of Engineering, University of Tehran, Tehran, Iran
(as_heidari@ut.ac.ir)

[3] College of Computer Science, Sichuan University, Chengdu, Sichuan 610065, China
(liulei.cx@gmail.com)

[4] Key Laboratory of Intelligent Informatics for Safety & Emergency of Zhejiang Province, Wenzhou University, Wenzhou 325035, China
(zhangxiaoqinnan@gmail.com, chenhuiling.jlu@gmail.com)

[5]Department of Computer Science, Birzeit University, POBox 14, West Bank, Palestine
(mmafarja@birzeit.edu)

*Corresponding Author: Dong Zhao, Huiling Chen
E-mail: zd-hy@163.com (Dong Zhao), chenhuiling.jlu@gmail.com (Huiling Chen)

**Abstract:** This paper proposes an efficient optimization algorithm based on the physical phenomenon of rime-ice, called the RIME or rime optimization algorithm. The RIME algorithm implements the exploration and exploitation behaviors in the optimization methods by simulating the soft-rime and hard-rime growth process of rime-ice and constructing a soft-rime search strategy and a hard-rime puncture mechanism. Meanwhile, the greedy selection mechanism in the algorithm is improved, and the population is updated in the stage of selecting the optimal solution to enhance the exploitation capability of the RIME. In the experimental, this paper conducts qualitative analysis experiments on the RIME to clarify the characteristics of the algorithm in the process of finding the optimal solution. The performance of RIME is then tested on a total of 42 functions in the classic IEEE CEC2017 and the latest IEEE CEC2022 test sets. The proposed algorithm is compared with 10 well-established algorithms and 10 latest improved algorithms to verify its performance advantage. In addition, this paper designs experiments for the parametric analysis of RIME to discuss the potential of the algorithm in running different parameters and handling different problems. Finally, this paper applies RIME to five practical engineering problems to verify its effectiveness and superiority in real-world problems. The statistical and comparison results show that the RIME is a strong and competitive algorithm. The source codes of the RIME algorithm will be publicly available at https://aliasgharheidari.com/RIME.html.

**Keywords:**
Meta-heuristic; Optimization; Rime optimization algorithm; RIME; Metaheuristic; Swarm-intelligence; Nature-inspired computing, Genetic algorithm, Engineering design problems

**Nomenclature**

| | | | |
|---|---|---|---|
| RIME | Rime optimization algorithm | $R_{ij}^{new}$ | The new position of the updated particle |

| | | | |
|---|---|---|---|
| MA | Meta-heuristics | $R_{best,j}$ | The best rime agent in the rime-population |
| ΔABC | Rime's growth plane | $r_1, r_2, r_3$ | A random number |
| D1, D2, D3, D4 | The birth points of rime | $COS\theta$ | The direction of particle movement |
| $R$ | The rime-population | $t$ | The current number of iterations |
| $S_i$ | The rime-agents | $T$ | The maximum number of iterations |
| $x_{ij}$ | The rime-particles | $\beta$ | The environmental factor |
| $d$ | The dimension of the population | $w$ | The number of segments of the step function |
| $i$ | The ordinal number of rime-agent | $h$ | The degree of adhesion |
| $j$ | The ordinal number of rime-particle | $Ub_{ij}, Lb_{ij}$ | The upper and lower bounds of the escape space |
| $F(S_i)$ | The fitness value of the agent | $E$ | The coefficient of being attached |
| $F^{normr}(S_i)$ | The normalized value of the current agent fitness value | $FEs$ | The current number of evaluations |

# 1 Introduction

With the rapid progress of artificial intelligence (AI) technology, academic research and engineering application fields face increasingly complex optimization problems. Optimization algorithms refer to stochastic or deterministic methods that deal with feature space by minimizing or maximizing a single, multiple, or many objective functions considering gradient info, constraints, or priority of the system or decision maker [1-3]. At present, traditional optimization methods include some classical operations, such as linear programming, nonlinear programming, integer programming, etc., and the later developed Newton's method [4], conjugate gradient method [5], gradient descent method [6], etc. Although these methods can find the global optimal solution to some problems, they may require that the feasible domain be convex, the objective function is continuously differentiable, or some more constraints [7]. However, some complex optimization problems are usually non-differentiable, non-convex, multi-modal, or cannot be solved with old frameworks [8, 9]. Thus, they are challenging to be solved with traditional optimization methods [10]. Moreover, when encountering large-scale NP-hard problems [11, 12], traditional methods often find it difficult to obtain satisfactory solutions reasonably. Therefore, an accurate, efficient, and stable optimization algorithm is urgently needed to solve these difficult problems [13, 14].

Evolutionary algorithms (EA) and the particular class of metaheuristic algorithms (MAs) are derivative-free optimization methods that do not rely on the gradient information of the problem being solved [15, 16]. EA and MAs do not guarantee a globally optimal solution but can obtain a near-optimal solutions in a reasonable time, making them very suitable for solving complex optimization problems in practice and for many applications [17, 18]. Most MAs imitate natural or social phenomena, are inspired by other disciplines, and are widely applied to optimization problems in various fields [19, 20].

According to the internal operation process of MAs, metaheuristic optimization algorithms can be divided into four main categories of algorithms. The first is evolution-based algorithms, which mainly

simulate the evolutionary law of superiority and inferiority in nature to achieve the overall progress of the population and finally complete the solution of the optimal solution. The representative algorithms of evolution-based algorithms are the genetic algorithm (GA) [21] and the differential evolution algorithm (DE) [22]. With the rise of evolutionary algorithms, more and more algorithms of the same type are proposed, for example, evolutionary strategy (ES) [23], evolutionary programming (EP) [24], and gene expression programming (GEP) [25], etc. While biological evolutionary algorithms have been intensively studied, several researchers have found that biological populations also contain the behavior of seeking advantages, and therefore, algorithms based on population intelligence have gradually emerged. These algorithms are designed to iteratively update the near-optimal solution by simulating the collaboration or information exchange among populations and then lock the global optimal solution. The representative algorithms with widespread adoption are particle swarm optimization (PSO) [26], Harris hawks optimization (HHO) [27], slime mould algorithm (SMA) [28], hunger games search (HGS) [29], grey wolf optimization (GWO) [30], colony predation algorithm (CPA) [31] and whale-optimization algorithm (WOA) [32]. Since the inception of swarm intelligence-based algorithms, researchers have proposed several homogeneous algorithms for biological populations, such as ant colony optimization (ACO) [33, 34], artificial bee colony (ABC) [35], bat algorithm (BA) [36], and cuckoo search (CS) [37], etc.

With the study of biological populations, human populations have inevitably been chosen as the focus of research, and algorithms based on human systems have been proposed one after another. Behaviors mainly inspire such algorithms in human systems, such as teaching-learning, social, learning, emotional, and managerial behaviors. Among them, teaching-learning-based optimization (TLBO) [38] is the main one. As researchers investigate deeper, algorithms based on human systems have emerged, such as harmony search (HS) [39], group search optimizer (GSO) [40], interior search algorithm (ISA) [41], mine blast algorithm (MBA) [42] and social group optimization (SGO) [43], etc.

The last category is the algorithms based on natural phenomena, which are not related to biological groups but are mainly derived from the reflection of physical and chemical rules in nature. The algorithms based on natural phenomena usually take the motion of the object or particle as the criterion to update the searched agent, take the stability of the object or environment as the judgment of the optimization state, and then regulate the next iteration of the algorithm. Among them, simulated annealing (SA) [44] is the representative with high visibility and wide application. Other common algorithms for natural phenomena, such as biogeography based optimization (BBO) [45], and gravitational search algorithm (GSA) [46], etc.

Although the above algorithms perform a significant role in natural-phenomena optimization, there are still certain limitations, which can be summarized as follows:

1) The natural-phenomena algorithms are usually divided into the exploration and exploitation phases. The exploration phase is characterized by the strong randomness of the algorithm, high update efficiency, and unstable solution quality after each iteration; the strong stability of the algorithm, low update efficiency, and stable solution quality after each iteration characterize the exploitation phase. Therefore, balancing these two optimization-seeking phases of the algorithm is crucial and directly affects the algorithm's performance.

2) The parameters significantly influence the optimization performance of most algorithms, and it is difficult to determine the exact parameters for optimization problems. When proposed algorithms lack qualitative analysis and do not elucidate parameter sensitivity, making it is difficult for them to solve problems in depth.

3) Some algorithms are proposed with wastefully focus on novelty claims with employing new metaphors, instead of computational performance advantages over complex problems, or they tested only for one type of problems, with limited variety of test sets, resulting in high complexity, low compatibility, and poor results when dealing with those new algorithms for other types of problems.

Although, according to the no free lunch theory [47], algorithms cannot be adapted to all types of optimization problems, the actual application scenarios are complex and diverse, and it is necessary to balance the specialization and adaptability of algorithms so that they can be widely disseminated and applied.

To solve the above problems as much as possible, this study proposes a meta-heuristic algorithm based on natural phenomena, the rime optimization algorithm (RIME), inspired by the growth behavior of rime-ice in nature. The RIME consists of three main processes:

1) Simulating the motion of soft-rime particles in rime-ice and proposing a soft-rime search strategy, mainly used to explore the algorithm. This strategy has a unique stepwise exploration and exploitation method, and the algorithm can continuously switch between large-range exploration and small-range exploitation to achieve high efficiency and high accuracy.

2) Simulate the crossover behavior between hard-rime agents, and propose a hard-rime puncture mechanism, mainly used to exploit the algorithm. This mechanism achieves effective information exchange between agents through dimensional cross-swapping between ordinary agents and optimal agents.

3) Enhance the greedy selection mechanism of the algorithm, and propose the positive greedy selection mechanism. This mechanism filters the inferior solutions in the population and actively introduces suboptimal solutions by changing the selection of optimal solutions. On the premise of ensuring the quality of the population, the diversity of the population is increased, and the algorithm is avoided to fall into local optimum as much as possible.

In the experiments, to elucidate the characteristics and adaptability of the RIME algorithm, qualitative analysis experiments and parameter sensitivity experiments are designed in this paper. Further, to verify the comprehensive performance of the algorithm, this paper tests the RIME algorithm with 10 highly-cited original algorithms and 10 recent high-performance improved algorithms on the test sets of CEC2017 [48] and CEC2022 [49], respectively. In addition, this paper designs experiments for the parametric analysis of RIME to discuss the algorithm's potential when running with different parameters and dealing with different problems. Finally, to verify the ability of the algorithm to solve real-world problems, the RIME algorithm is used in this paper for five classical engineering optimization problems, including pressure vessel design (PVD) problem, welded beam design (WBD) problem, speed reducer design (SRD) problem, I-beam design (IBD) problem, and multiple disk clutch brake design (MDCBD) problem.

In summary, the contributions of this paper are as follows:

1) A novel meta-heuristic algorithm based on natural phenomena, called the rime optimization algorithm, is inspired by the growth of rime-ice.

2) A new exploration strategy, exploitation mechanism, and selection mechanism are constructed in the RIME algorithm, and each strategy is portable and can be used to improve peer algorithms.

3) Through qualitative analysis experiments and parameter sensitivity experiments, the algorithmic characteristics of RIME are detailed for more relevant application to various optimization problems.

4) A comparison experiment between RIME and 20 peer algorithms is designed based on the complete data set, and the experimental results confirm that the RIME has a tremendous advantage over peer algorithms in terms of optimal performance in various types of problems.

5) The RIME algorithm is applied to five practical engineering optimization problems, which initially demonstrates the algorithm's potential for application to practical optimization problems and can be subsequently used on other optimization problems.

The rest of this paper is structured as follows. In Section 2, the mechanism of rime-ice formation is described and the mechanism inspired is illustrated. In Section 3, rime formation is modeled, and the RIME algorithm is proposed. Section 4 describes the experiments involved in this work, including qualitative analysis, performance experiments, parametric analysis, and practical applications. Section 5 concludes the whole paper and clarifies future research directions.

# 2 Inspiration from the formation of rime-ice

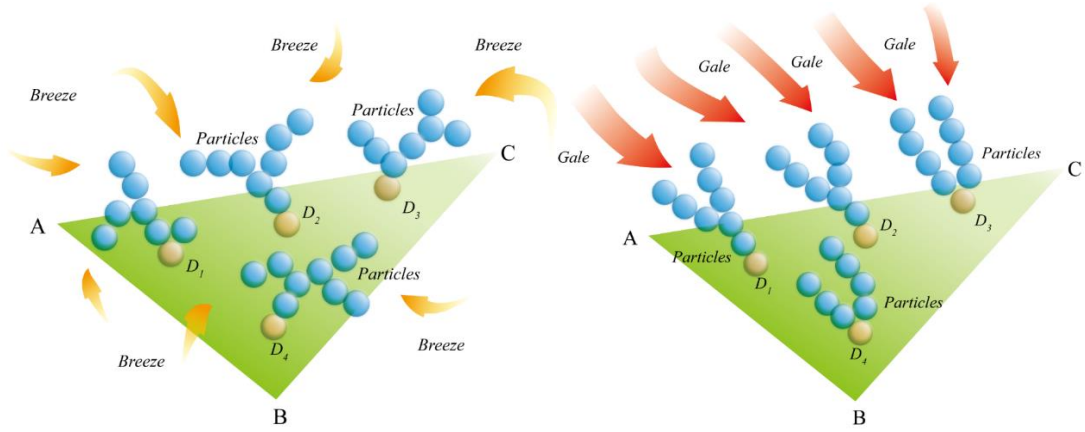
Rime-ice is resulted from accumulated water vapor in the air that has not yet condensed. It freezes and sticks to objects such as tree branches at low temperatures. Due to their unique climatic characteristics and topography, some regions form a unique landscape like rime-ice every year, as shown in Figure 1.

(a) soft rime (b) hard rime

**Figure 1.** Rime-ice real scene[1]

[1] Pictures obtained from https://pixabay.com/ as copy right free images
(a) https://pixabay.com/photos/barbed-wire-frost-frozen-cold-ice-1938842/
(b) https://pixabay.com/photos/thuja-ice-winter-cold-frozen-6015613/

The growth process of rime ice is determined by the temperature, wind speed, humidity, air, and other factors, and rime formation varies under different conditions. At the same time, due to the influence of environmental factors and the growth pattern, rime-ice cannot grow indefinitely, and it will stop growing when it reaches a relatively stable state. The growth pattern of rime is generally divided into two types: soft-rime and hard-rime, which is mainly determined by the wind speed during the formation process, as shown in Figure 2, where ΔABC represents the growth plane of rime, and D1, D2, D3, D4 represent the birth points of rime. Usually, soft rime is generated in a breeze environment, and hard rime is formed in a high-wind environment. The breeze is characterized by small wind speed and variable wind direction, and the wind exists in all directions simultaneously and in the same height plane, as shown in Figure 2(a). Therefore, the soft rime formed by the breeze grows slowly and randomly. On the other hand, a gale is characterized by high wind speed and roughly the same wind direction in the same height plane, as shown in Fig. 2(b). Therefore, the hard rime formed by the gale is fast and grows in approximately the same direction.




(a) soft-rime (b) hard-rime

**Figure 2.** The formation process of soft rime and hard rime under different environments

In summary, this study is inspired by the growth mechanism of rime-ice and proposes a soft-rime search strategy for algorithm search by simulating the motion of soft-rime particles. Also, a hard-rime puncture mechanism is proposed to exploit the algorithm by simulating the crossover behavior between hard rime agents. Finally, the selection mechanism of the metaheuristic algorithm is improved, and the positive greedy selection mechanism is proposed. This paper proposes the RIME algorithm with better performance by combining the above three mechanisms.

# 3 Mathematical model of the RIME

In this section, the growth process of each rime strip is simulated by analyzing the effects of wind speed, freezing coefficient, the cross-sectional area of the attached material, and growth time. On the other hand, inspired by the diffusion-limited aggregation [50] method of simulating metal particle aggregation, the motion process of each rime particle coalescing into a rime agent is simulated by modeling the motion behavior of each rime particle, and the final generated rime-agent is in the form of a strip crystal. The RIME consists of four stages: the initialization of rime clusters, the proposed soft-rime search strategy, the proposed hard-rime

puncture mechanism, and the improvement of the greedy selection mechanism.

## 3.1 Rime cluster initialization

Inspired by reality, this paper treats each agent rime as the searched agent of the algorithm and the rime-population formed by all agents as the population of the algorithm. Firstly, the whole rime-population $R$ is initialized. The rime population consists of $n$ rime agents $S_i$ and each rime- agent consists of $d$ rime-particles $x_{ij}$, as shown in Figure 3 and Eq. (1). Thus, the rime-population $R$ can be directly represented by the rime-particles $x_{ij}$, as shown in Eq. (2).

$$R = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_i \end{bmatrix}; S_i = \begin{bmatrix} x_{i1} & x_{i2} & \cdots & x_{ij} \end{bmatrix} \tag{1}$$

$$R = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1j} \\ x_{21} & x_{22} & \cdots & x_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ x_{i1} & x_{i2} & \cdots & x_{ij} \end{bmatrix} \tag{2}$$

where $i$ is the ordinal number of the rime agent and $j$ is the ordinal number of the rime particle. In addition, $F(S_i)$ is used to denote the growth state of each rime-agent, i.e., the fitness value of the agent in the meta-heuristic algorithm.

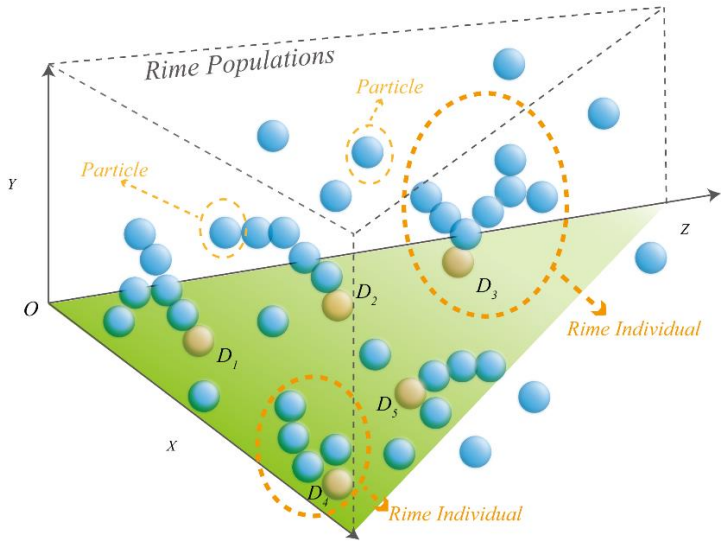



**Figure 3.** Initialization of the rime space

## 3.2 Soft-rime search strategy

In a breezy environment, soft-rime growth is strongly random, and the rime particles can freely cover

most of the surface of the attached object but grow slowly in the same direction. Inspired by the growth of soft-rime, this study proposes a soft-rime search strategy using the strong randomness and coverage of rime particles, which enables the algorithm to cover the entire search space in the early iteration quickly and does not easily fall into the local optimum.

When the rime particles condense into soft-rime agents, there are the following characteristics:

1) Before the particles condense to form a soft rime agent, each particle $x_{ij}$ will wander according to a certain law, and the efficiency of the wandering is affected by environmental factors.

2) If the free-state rime particles move to the vicinity of a soft-rime agent, they will condense with the particles in the agent so that the stability of the soft-rime agent will change.

3) The distance between the centers of the two particles adhering to each other is not fixed, as the degree of condensation varies between each particle.

4) If the particles move directly outside the escape radius, no interparticle condensation occurs.

5) During the formation of a soft rime, the random condensation of each particle increases the area to which the agent is attached, resulting in a greater probability of free particle condensation. However, the agent will not grow indefinitely and will eventually reach a stable state due to environmental factors.

In this paper, corresponding to the five motion characteristics of the rime particles, the process of condensation of each particle is concisely simulated, as shown in Figure 4, and the position of the rime-particles is calculated as shown in Eq. (3).

$$R_{ij}^{new} = R_{best,j} + r_1 \cdot \cos\theta \cdot \beta \cdot \left(h \cdot \left(Ub_{ij} - Lb_{ij}\right) + Lb_{ij}\right), r_2 < E \tag{3}$$

where, $R_{ij}^{new}$ is the new position of the updated particle, and $i$ and $j$ denote the $j$-th particle of the $i$-th rime-agent. $R_{best,j}$ is the $j$-th particle of the best rime-agent in the rime-population $R$. The parameter $r_1$ is a random number in the range (-1,1) and $r_1$ controls the direction of particle movement together with $\cos\theta$ will change following the number of iterations, as shown in Eq. (4). $\beta$ is the environmental factor, which follows the number of iterations to simulate the influence of the external environment and is used to ensure the convergence of the algorithm, as shown in Eq. (5). $h$ is the degree of adhesion, which is a random number in the range (0,1), and is used to control the distance between the centers of two rime-particles.

$$\theta = \pi \cdot \frac{t}{10 \cdot T} \tag{4}$$

where $t$ is the current number of iterations and $T$ is the maximum number of iterations of the algorithm.

$$\beta = 1 - [\frac{w \cdot t}{T}]/w \tag{5}$$

where the mathematical model of $\beta$ is the step function, $[\cdot]$ denotes rounding; the default value of $w$ is 5, which is used to control the number of segments of the step function. Returning to Eq. (3), $Ub_{ij}$ and $Lb_{ij}$ are the upper and lower bounds of the escape space, respectively, which limit the effective region of particle motion. $E$ is the coefficient of being attached, which affects the condensation probability of an agent and increases with the number of iterations, as shown in Eq. (6).

$$E = \sqrt{(t/T)} \tag{6}$$

$r_2$ is a random number in the range ([51]) which, together with $E$, controls whether the particles condense, i.e., whether the particle positions are updated. The pseudo-code for the soft-rime search strategy is shown in Algorithm 1.

**Algorithm 1** Pseudo-code of the soft-rime search strategy

Initialize the rime-population $R$
Get the current optimal agent and optimal fitness
**While** $t \leq T$
    Coefficient of adherence $E = \sqrt{(t/T)}$
    **For** $i = 1:n$
        **For** $j = 1:d$
            **If** $r_2 < E$
                Position update according to the characteristics of the rime particles by Eq. (3)
            **End If**
        **End For**
    **End For**
    Update the current optimal agent and optimal fitness
    $t = t + 1$
**End While**

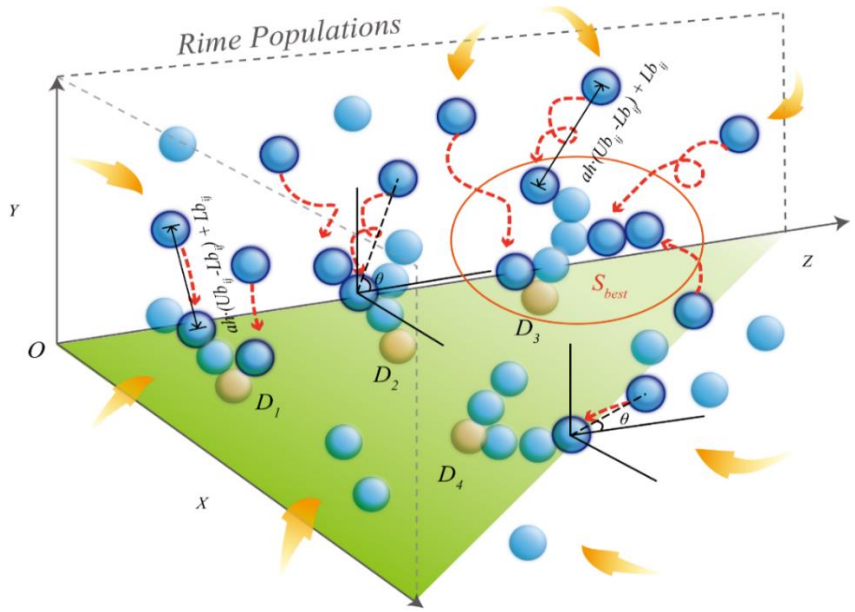



**Figure 4.** Soft-rime particles motion

## 3.3 Hard-rime puncture mechanism

In strong gale conditions, hard-rime growth is simpler and more regular than soft-rime growth. When the rime particle condenses into a hard rime, there are the following characteristics: 1) The gale is so strong that other influences are negligible, resulting in different hard-rime agents snowballing in the same direction. 2) Due to the growth direction being the same, each rime agent can easily cross over, a phenomenon called rime puncture. 3) Like soft-rime agents, hard-rime agents increase in size as they grow, resulting in a greater probability of puncturing between agents in better growing conditions.

Therefore, this paper is inspired by the puncturing phenomenon and proposes a hard-rime puncture mechanism, which can be used to update the algorithm between agents, so that the particles of the algorithm can be exchanged and the convergence of the algorithm and the ability to jump out of the local optimum can be improved. The puncture phenomenon is shown in Figure 5, and the formula for replacement between particles is shown in Eq. (7).

$$R_{ij}^{new} = R_{best,j} \ , r_3 < F^{normr}(S_i) \tag{7}$$

where $R_{ij}^{new}$ is the new position of the updated particle and $R_{best,j}$ is the $j$-th particle of the best rime-agent in the rime-population $R$. $F^{normr}(S_i)$ denotes the normalized value of the current agent fitness value, indicating the chance of the $i$-th rime-agent being selected. $r_3$ is a random number in the range (-1,1).

The pseudo-code for the hard-rime puncture mechanism is shown in Algorithm 2.

**Algorithm 2** Pseudo-code of the hard-rime puncture mechanism

Initialize the rime-population $R$
Get the current optimal agent and optimal fitness
**While** $t \leq T$

**For** $i = 1 : n$
  **For** $j = 1 : d$
    **If** $r_3 < Normalize\ fitness\ of\ S_i$
      Position update according to the characteristics of the rime-particles by Eq. (7)
    **End If**
  **End For**
**End For**
Update the current optimal agent and optimal fitness
$t = t + 1$
**End While**

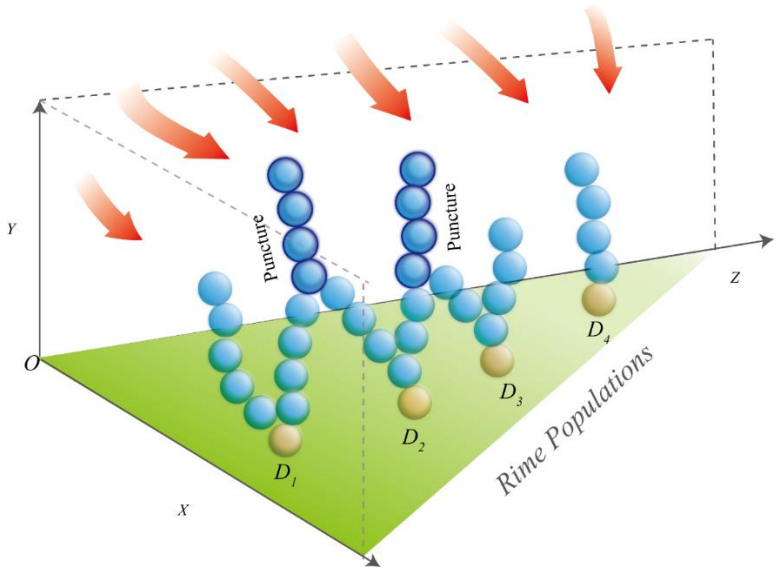



**Figure 5.** Hard-rime puncturing

## 3.4 Positive greedy selection mechanism

Typically, metaheuristic optimization algorithms have a greedy selection mechanism that replaces and records the best fitness value and the best agent after each update. The typical idea is to compare the updated fitness value of an agent with the global optimum, and if the updated value is better than the current global optimum, then the optimum fitness value is replaced, and the agent is recorded as the optimum. The advantage of such an operation is that it is simple and fast, but it does not help in the exploration and exploitation of the population and only serves as a record.

Therefore, the paper proposes an aggressive greedy selection mechanism for participating in population updates to improve global exploration efficiency. The specific idea is to compare the updated fitness value of an agent with the fitness value of an agent before the update, and if the updated fitness value is better than the value before the update, a replacement occurs, and also, the solution of both agents is replaced. On the one hand, this mechanism allows the population to continuously have good agents through active agent replacement, which improves the quality of the global solution. On the other hand, as the position of the agents of the population changes significantly with each iteration, there will inevitably be agents that are worse than the population before the update and are detrimental to the next iteration. Therefore, this operation can be used to ensure that the population evolves in a more optimal direction at each iteration.

In this paper, the pseudo-code of the positive greedy selection mechanism for solving the minimum value problem, as an example, is shown in **Algorithm 3**.

**Algorithm 3** Pseudo-code of the positive greedy selection mechanism

Initialize the rime population $R$
Get the current optimal agent and optimal fitness
**While** $t \leq T$
  **For** $i = 1 : n$
    **If** $F(R_i^{new}) < F(R_i)$ // Compare fitness values
      $F(R_i) = F(R_i^{new})$ // Replace fitness values
      $R_i = R_i^{new}$ // Replace the current agent
      **If** $F(R_i^{new}) < F(R_{best})$ // Compare optimal fitness values
        $F(R_{best}) = F(R_i^{new})$ // Record optimal fitness values
        $R_{best} = R_i^{new}$ // Record the current optimal agent
      **End If**
    **End If**
  **End For**
  $t = t + 1$
**End While**

## 3.5 Proposed RIME algorithm

In summary, firstly, inspired by the motion of soft-rime particles in this section, a unique stepwise search and exploitation approach is designed to propose a soft-rime search strategy as the core optimization-seeking method of the algorithm. Immediately afterward, inspired by the crossover of hard-rime agents, a hard-rime puncture mechanism is proposed to achieve dimensional crossover interchange between ordinary and optimal agents, which is conducive to improving the solution accuracy of the algorithm. Finally, based on the greedy selection mechanism, an improved positive greedy selection mechanism is proposed to increase the diversity of the population and prevent the algorithm from falling into the local optimum as far as possible by changing the selection of optimal solutions. The overall structure of the algorithm in terms of pseudo-code and flow chart is shown in Algorithm 4 and Figure 6.

**Algorithm 4** Pseudo-code of RIME

Initialize the rime population $R$
Get the current optimal agent and optimal fitness
**While** $t \leq T$
  Coefficient of adherence $E = (t/T)\text{^}0.5$
  **If** $r_2 < E$
    Update rime agent location by ***the soft-rime search strategy***
  **End If**
  **If** $r_3 < Normalize\ fitness\ of\ S_i$
    Cross updating between agents by ***the hard-rime puncture mechanism***
  **End If**
  **If** $F(R_i^{new}) < F(R_i)$
    Select the optimal solution and replace the suboptimal solution using ***the positive greedy selection mechanism***
  **End If**
  $t = t + 1$
**End While**

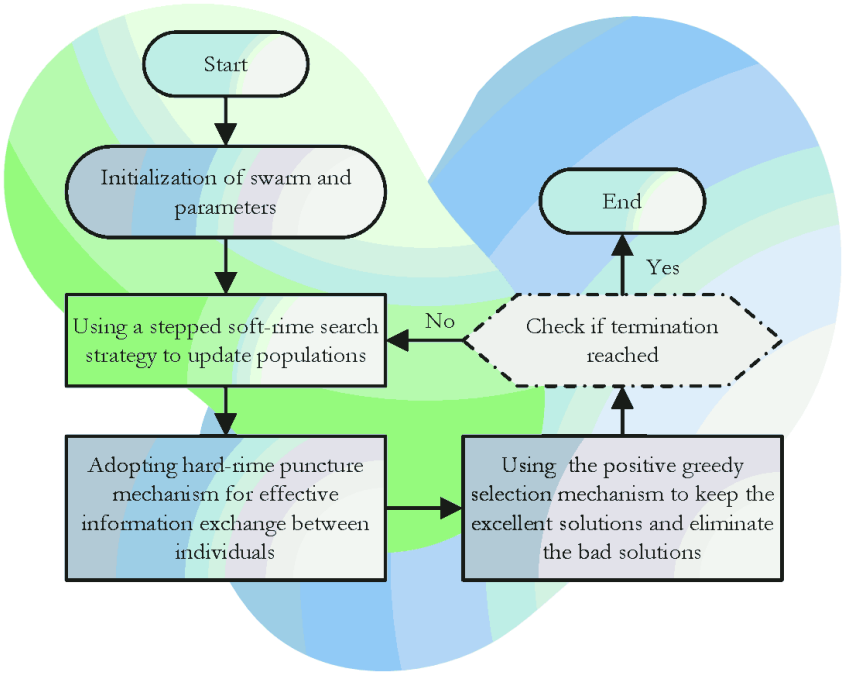



**Figure 6.** Flowchart of RIME

The complexity of RIME mainly includes the soft-rime search strategy, the hard-rime puncture mechanism, the positive greedy selection mechanism, and the calculation of the fitness value. First, the complexity level of the soft-rime search mechanism is $O(n^2)$. Then, the complexity level of the hard-rime puncture mechanism in the two extreme cases is $O(n)$ and $O(n^2)$. The complexity of the positive greedy selection mechanism is $O(n)$. Finally, the complexity level of the fitness value calculation is $O(n * logn)$. Therefore, the overall complexity level of the RIME algorithm is $O(RIME) = O((n + logn) * n)$.

# 4 Experiments and results

This section demonstrates the RIME algorithm's advantages and characteristics through experiments. Firstly, a qualitative analysis of the RIME algorithm demonstrates the algorithm's characteristics in finding the optimal solution. Then, the performance advantages of the algorithm are demonstrated experimentally by comparing the RIME algorithm with peer algorithms. Further, the parameter sensitivity analysis of the RIME algorithm is used to determine the parameters of RIME for different optimization problems to ensure maximum performance. Finally, the RIME algorithm is applied to five engineering optimization problems to demonstrate the algorithm's potential for application to practical optimization problems.

To ensure fairness and reproducibility of the experiments, all experiments in this paper were run in a unified environment where the software used was MATLAB 2017b and the core hardware was an Intel(R) Xeon(R) CPUE5-2660v3 (2.60GHz).

## 4.1 Qualitative analysis of RIME

This subsection uses the classical 23 benchmark functions [52] to design four experiments for

qualitatively analyzing the RIME algorithm concerning agents, dimensional particles, fitness values and iteration curves.

First, to analyze the distribution of optimal agents in the solution space of the RIME algorithm in the optimization problem, the search characteristics of the agents of the algorithm are visualized. In this paper, experiments on the historical position of agents are designed by recording the position of the optimal agent for each iteration. Further, while analyzing the agent updates, the paper analyses the particles in the agents, i.e., the dimensionality of the agent row vectors, to demonstrate the pattern and magnitude of change of the particles. In this paper, particle change experiments are designed by recording the first particle of the optimally solved agent at each iteration. Then, to analyze the changing trend of the fitness value of the algorithm after each iteration, this paper designs the fitness value change experiment by recording the optimal fitness value after each update. Finally, to analyze the algorithm's overall iteration trend, this paper records the fitness value of the optimal solution after each iteration and designs an iteration curve experiment. In the experiments, the population size of RIME is set to 30, the number of iterations is 2000, and 30 iterations are independently parallel.

Figure 7 shows the results of RIME for the four qualitative experiments described above. In particular, the plots in the column of Figure 7(a) represent the 3D location distribution of all solutions for each benchmark function, and it is within this solution space that RIME searches for the optimal solution. Figure 7(b) represents the two-dimensional location distribution of RIME search histories. It can be seen that a small number of historical optimal solutions are scattered within the solution space, and most of the historical optimal solutions are clustered around the global optimal solution. This indicates that the RIME algorithm can find the approximate optimal solution within the solution space quickly and can enter the exploitation stage earlier to improve the accuracy of the solution.

Figure 7(c) records the trend of the first dimension of the RIME's agent throughout the iterations. It can be seen that the RIME algorithm has very long search steps in F1, F3, F10, F11, F15, F18 due to the role of the soft-rime search strategy in the early search phase, which is very beneficial for the algorithm to go out of local optimality and find the global optimal solution. The hard-rime puncture mechanism plays a significant role in the later iterations of the algorithm, resulting in shorter search steps, which helps to improve the accuracy of the optimal solution. On the other hand, the agent dimensions also oscillate less with the number of iterations. This facilitates the algorithm to converge quickly during the exploitation phase.

Figure 7(d) records the optimal fitness values of the RIME after each iteration. It can be seen that, again, due to the search agents of the algorithm actively adjusting their search positions as they are updated, the fitness values for each iteration follow the agents with regular fluctuations. This indicates that the RIME algorithm is effective in searching for the optimal solution with agent updates, which is in line with the design of the algorithm.

Figure 7(e) shows the overall iterative convergence curve of RIME. For most of the tested functions, the algorithm improves the quality of the solution as the number of iterations increases and does not fall into a local optimum. Further, Figure 7(e) also shows that whether on simple single-peaked test functions such as F1 and F3 or complex multi-peaked and composite functions such as F7 and F13, RIME algorithms can search and develop incrementally during iterations through the combined action of the soft-rime search strategy and hard-rime puncture mechanism, avoiding as much as possible the algorithm falling into the local optimum trap.

In summary, the characteristics of RIME include: 1) It can quickly find the global approximate optimal solution, centralize exploitation and improve solution accuracy. 2) While ensuring convergence speed, the search position is actively changed during updates to improve the algorithm's global exploration capability and ability to jump out of local optima. 3) In the process of finding the optimal solution, the RIME algorithm has a unique stepped exploration and exploitation approach, which allows the algorithm to continuously RIME algorithm has a unique stepped exploration and exploitation approach, which allows the algorithm to continuously switch between large-scale exploration and small-scale exploitation directly, allowing for both breadth and depth in the search for optimality.

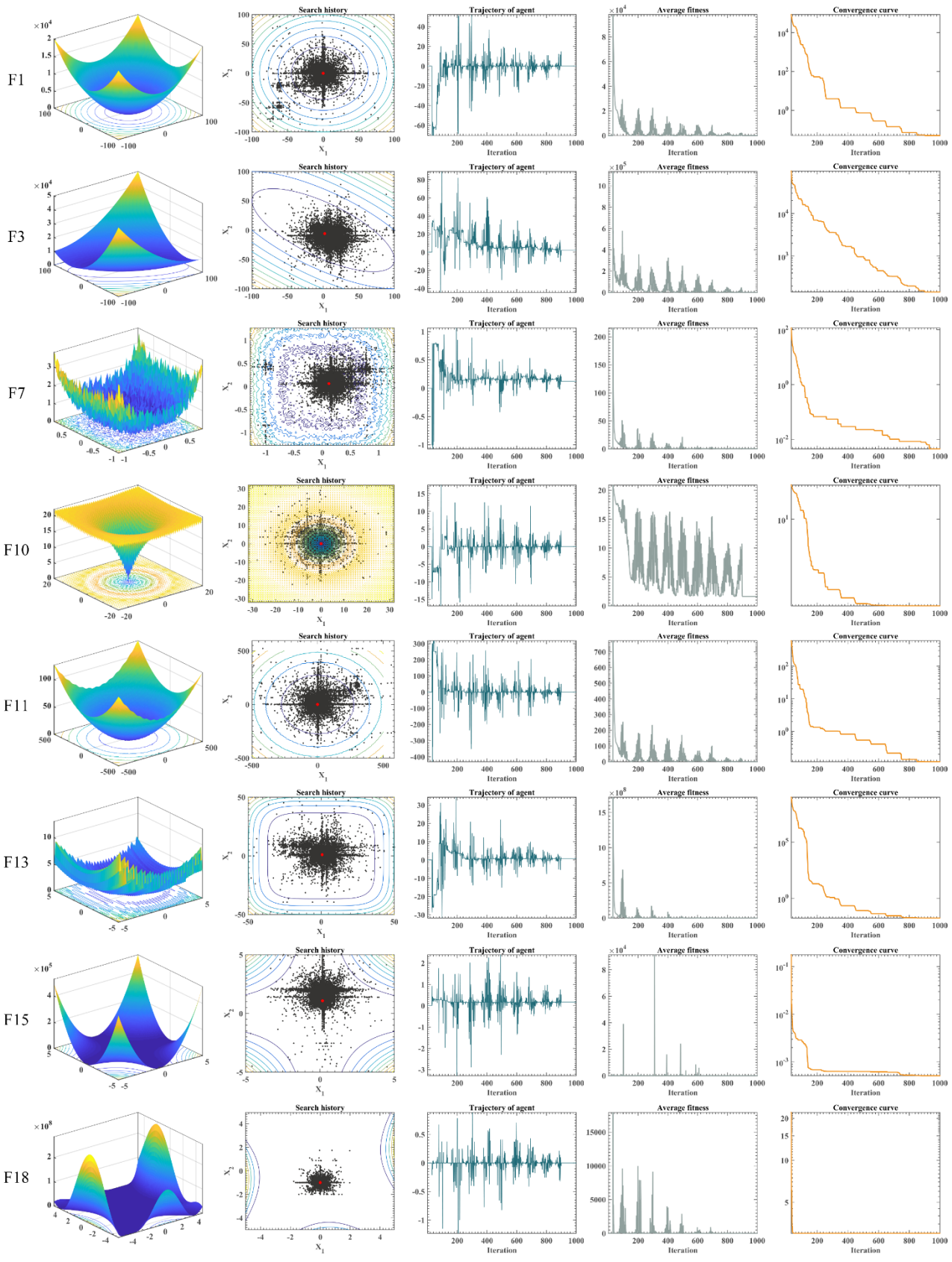



**Figure 7.** Qualitative analysis experiment of RIME

## 4.2 Performance comparison experiment of RIME

In this section, the RIME algorithm is compared with 10 classical algorithms and 10 high-performance improvement algorithms, respectively, to demonstrate the superiority of the proposed algorithm among peer algorithms. It is worth mentioning that the algorithms are compared with a complete test set for the experiments to fully demonstrate the performance advantages and characteristics of the algorithms.

### 4.2.1 Classical algorithms performance comparison and analysis

In this subsection, RIME is compared with 10 original highly-cited algorithms, including PSO [26], WOA [32], HHO [27], moth-flame optimization (MFO) [53], Jaya optimization algorithm (JAYA) [54], firefly algorithm (FA) [55], red fox optimization algorithm (RFO) [56], GWO [30], and BA [36], to demonstrate the superiority and reasonableness of the proposed algorithm. The function test set for the comparison was adopted from IEEE CEC2017 [48], which is described in Table 1. This function set contains unimodal, multimodal, hybrid, and composition functions to assess each algorithm's performance under different optimization problems comprehensively. The starting population size of all swarm intelligence algorithms is set to 30, the maximum number of evaluations is set to 30,0000, and 30 comparison experiments are performed separately on this basis in order to ensure the fairness of the experiments. The key parameters of the algorithms used for the comparison were all adopted as default values, and details of the parameters are given in Table 2.

**Table 1.** Details of the IEEE CEC2017

| | | Functions | $f_{min}$ |
|---|---|---|---|
| Unimodal Functions | F1 | Shifted and Rotated Bent Cigar Function | 100 |
| | F2 | Shifted and Rotated Sum of Different Power Function | 200 |
| | F3 | Shifted and Rotated Zakharov Function | 300 |
| Multimodal Functions | F4 | Shifted and Rotated Rosenbrocks Function | 400 |
| | F5 | Shifted and Rotated Rastrigins Function | 500 |
| | F6 | Shifted and Rotated Expanded Scaffers F6 Function | 600 |
| | F7 | Shifted and Rotated Lunacek Bi_Rastrigin Function | 700 |
| | F8 | Shifted and Rotated Non-Continuous Rastrigins Function | 800 |
| | F9 | Shifted and Rotated Levy Function | 900 |
| | F10 | Shifted and Rotated Schwefels Function | 1000 |
| Hybrid Functions | F11 | Hybrid Function 1 (N = 3) | 1100 |
| | F12 | Hybrid Function 2 (N = 3) | 1200 |
| | F13 | Hybrid Function 3 (N = 3) | 1300 |
| | F14 | Hybrid Function 4 (N = 4) | 1400 |
| | F15 | Hybrid Function 5 (N = 4) | 1500 |
| | F16 | Hybrid Function 6 (N = 4) | 1600 |
| | F17 | Hybrid Function 6 (N = 5) | 1700 |
| | F18 | Hybrid Function 6 (N = 5) | 1800 |
| | F19 | Hybrid Function 6 (N = 5) | 1900 |
| | F20 | Hybrid Function 6 (N = 6) | 2000 |

| | | | |
|---|---|---|---|
| Composition Functions | F21 | Composition Function 1 (N = 3) | 2100 |
| | F22 | Composition Function 2 (N = 3) | 2200 |
| | F23 | Composition Function 3 (N = 4) | 2300 |
| | F24 | Composition Function 4 (N = 4) | 2400 |
| | F25 | Composition Function 5 (N = 5) | 2500 |
| | F26 | Composition Function 6 (N = 5) | 2600 |
| | F27 | Composition Function 7 (N = 6) | 2700 |
| | F28 | Composition Function 8 (N = 6) | 2800 |
| | F29 | Composition Function 9 (N = 3) | 2900 |
| | F30 | Composition Function 10 (N = 3) | 3000 |

**Table 2.** Key parameters of each classical algorithm

| Algorithms | Key parameters |
|---|---|
| PSO | $c_1 = 2;\ c_2 = 2;\ V_{max} = 6$ |
| WOA | $a_1 = [2,0]; a_2 = [-2,-1]; b = 1$ |
| SCA | $a = 2$ |
| HHO | $k = 0$ |
| MFO | $b = 1$ |
| JAYA | ~ |
| FA | $\alpha = 0.5;\ \beta_{min} = 0.2;\ \gamma = 1$ |
| RFO | $c1 = 0.18; c2 = 0.82$ |
| GWO | $a = [2\ 0]$ |
| BA | $A = 0.5; r = 0.5$ |

The experimental comparison results of RIME with other classical algorithms are given in Table 3, where AVG and STD denote the mean and variance of the algorithms after 30 independent runs, respectively, and the optimal values are bolded in each column. By comparing and observing the average values (AVG), it can be initially seen that for most of the benchmark functions, RIME has the smallest average value. This indicates that RIME obtains relatively higher quality solutions when using RIME and similar algorithms to optimize the benchmark functions. It is particularly effective in the two types of tests: hybrid and composite functions. This indicates that the RIME algorithm will have a stronger optimization ability when facing complex problems. Also, the optimal solution's standard deviation (STD) is smaller, indicating the high stability of RIME in optimizing the benchmark functions.

**Table 3.** Comparison results of RIME and classic algorithms

| | F1 | | F2 | | F3 | |
|---|---|---|---|---|---|---|
| | AVG | STD | AVG | STD | AVG | STD |
| RIME | **9.5804E+03** | **7.2134E+03** | 1.1923E+03 | 2.9406E+03 | 3.0175E+02 | 7.8542E-01 |
| PSO | 1.3314E+08 | 1.5266E+07 | 2.8201E+13 | 3.1302E+13 | 6.4339E+02 | 5.1363E+01 |
| WOA | 2.8107E+06 | 2.1110E+06 | 4.9997E+21 | 1.4877E+22 | 1.5929E+05 | 6.5854E+04 |
| SCA | 1.2402E+10 | 2.5992E+09 | 4.9734E+34 | 1.8293E+35 | 3.7533E+04 | 7.0872E+03 |

| | | | | | | |
|---|---|---|---|---|---|---|
| HHO | 1.0773E+07 | 1.9815E+06 | 9.8126E+11 | 1.9924E+12 | 4.5983E+03 | 1.9073E+03 |
| MFO | 1.3056E+10 | 8.8314E+09 | 1.7391E+40 | 9.4287E+40 | 1.0056E+05 | 6.7313E+04 |
| JAYA | 5.5345E+09 | 8.5545E+08 | 1.1693E+31 | 3.6259E+31 | 4.1615E+04 | 7.1914E+03 |
| FA | 1.4385E+10 | 1.6036E+09 | 5.8984E+33 | 9.3635E+33 | 6.1052E+04 | 7.4163E+03 |
| RFO | 3.9392E+06 | 7.3227E+05 | 2.4407E+02 | 2.7604E+01 | 3.1073E+02 | 1.6168E+00 |
| GWO | 1.6828E+09 | 1.8882E+09 | 1.0966E+33 | 5.3786E+33 | 3.5221E+04 | 9.8938E+03 |
| BA | 5.5668E+05 | 3.0421E+05 | **2.0000E+02** | **7.8717E-05** | **3.0011E+02** | **1.1557E-01** |
| | F4 | | F5 | | F6 | |
| | AVG | STD | AVG | STD | AVG | STD |
| RIME | 4.9309E+02 | **1.8031E+01** | **5.7601E+02** | 2.2968E+01 | **6.0033E+02** | **3.7670E-01** |
| PSO | 4.8427E+02 | 2.0761E+01 | 7.4444E+02 | 3.0290E+01 | 6.5256E+02 | 1.3483E+01 |
| WOA | 5.5550E+02 | 4.1366E+01 | 7.7923E+02 | 4.7730E+01 | 6.6949E+02 | 7.9739E+00 |
| SCA | 1.3763E+03 | 2.1709E+02 | 7.7826E+02 | 2.2831E+01 | 6.4905E+02 | 5.2621E+00 |
| HHO | 5.1296E+02 | 3.4899E+01 | 7.3322E+02 | 3.3309E+01 | 6.5935E+02 | 7.6759E+00 |
| MFO | 1.1303E+03 | 5.5911E+02 | 7.1955E+02 | 4.7249E+01 | 6.4510E+02 | 1.4605E+01 |
| JAYA | 7.6059E+02 | 5.8431E+01 | 7.4070E+02 | 1.6189E+01 | 6.2102E+02 | 3.1399E+00 |
| FA | 1.3352E+03 | 1.2754E+02 | 7.5746E+02 | **1.0748E+01** | 6.4376E+02 | 2.6379E+00 |
| RFO | 4.8961E+02 | 1.9568E+01 | 8.1102E+02 | 2.4516E+01 | 6.7407E+02 | 8.1524E+00 |
| GWO | 5.7622E+02 | 6.3779E+01 | 6.0861E+02 | 3.5808E+01 | 6.0910E+02 | 5.1502E+00 |
| BA | **4.7088E+02** | 3.3240E+01 | 8.2140E+02 | 6.7974E+01 | 6.6968E+02 | 1.1175E+01 |
| | F7 | | F8 | | F9 | |
| | AVG | STD | AVG | STD | AVG | STD |
| RIME | **8.2166E+02** | 2.0938E+01 | **8.8070E+02** | 2.0781E+01 | **1.2495E+03** | 5.2498E+02 |
| PSO | 9.2237E+02 | **1.5451E+01** | 9.9408E+02 | 2.2937E+01 | 5.9619E+03 | 2.8103E+03 |
| WOA | 1.2484E+03 | 9.6092E+01 | 1.0087E+03 | 5.2253E+01 | 7.5749E+03 | 2.3131E+03 |
| SCA | 1.1262E+03 | 3.7943E+01 | 1.0465E+03 | 1.6814E+01 | 5.4451E+03 | 1.1343E+03 |
| HHO | 1.2317E+03 | 5.3750E+01 | 9.6715E+02 | 2.2308E+01 | 6.7807E+03 | 8.1372E+02 |
| MFO | 1.1371E+03 | 2.2217E+02 | 1.0210E+03 | 5.9242E+01 | 7.1988E+03 | 2.2853E+03 |
| JAYA | 1.0279E+03 | 2.1959E+01 | 1.0319E+03 | 1.7385E+01 | 2.9897E+03 | 6.7114E+02 |
| FA | 1.3769E+03 | 3.5341E+01 | 1.0555E+03 | **1.0116E+01** | 5.1681E+03 | 5.9236E+02 |
| RFO | 1.3490E+03 | 2.3825E+01 | 1.0030E+03 | 1.2538E+01 | 7.3654E+03 | **3.2341E+02** |
| GWO | 8.6641E+02 | 4.8510E+01 | 8.9915E+02 | 2.9336E+01 | 1.5974E+03 | 4.2114E+02 |
| BA | 1.5731E+03 | 1.7582E+02 | 1.0545E+03 | 5.7951E+01 | 1.4170E+04 | 4.8362E+03 |
| | F10 | | F11 | | F12 | |
| | AVG | STD | AVG | STD | AVG | STD |
| RIME | **3.6038E+03** | 5.2470E+02 | **1.1810E+03** | **3.4567E+01** | 2.8309E+06 | 1.9095E+06 |
| PSO | 6.1450E+03 | 6.1459E+02 | 1.2843E+03 | 4.1459E+01 | 2.5427E+07 | 1.1081E+07 |
| WOA | 6.1178E+03 | 9.0644E+02 | 1.5206E+03 | 1.5114E+02 | 3.2674E+07 | 2.0903E+07 |
| SCA | 8.1471E+03 | 3.2952E+02 | 2.0374E+03 | 2.8993E+02 | 1.2375E+09 | 3.3003E+08 |
| HHO | 5.4451E+03 | 7.4591E+02 | 1.2580E+03 | 4.4123E+01 | 1.0311E+07 | 8.2110E+06 |
| MFO | 5.7599E+03 | 7.8055E+02 | 5.2847E+03 | 4.9290E+03 | 2.2682E+08 | 5.4299E+08 |
| JAYA | 7.9982E+03 | 3.3601E+02 | 1.9400E+03 | 1.2111E+02 | 1.6308E+08 | 4.9215E+07 |
| FA | 8.0624E+03 | **2.6609E+02** | 3.4727E+03 | 5.8443E+02 | 1.2999E+09 | 2.5306E+08 |
| RFO | 5.6638E+03 | 6.8758E+02 | 1.3122E+03 | 6.4424E+01 | 3.7778E+06 | 1.7836E+06 |
| GWO | 3.9186E+03 | 8.0741E+02 | 1.8961E+03 | 7.5896E+02 | 5.1506E+07 | 7.9362E+07 |
| BA | 5.7918E+03 | 7.3003E+02 | 1.2996E+03 | 6.9671E+01 | **2.1712E+06** | **1.7095E+06** |
| | F13 | | F14 | | F15 | |
| | AVG | STD | AVG | STD | AVG | STD |
| RIME | **2.0195E+04** | **1.9740E+04** | 1.4445E+04 | 9.4948E+03 | **1.2729E+04** | **1.2769E+04** |
| PSO | 4.6558E+06 | 1.3021E+06 | 7.9138E+03 | 4.3340E+03 | 4.8684E+05 | 2.0279E+05 |
| WOA | 1.4767E+05 | 8.2769E+04 | 9.3807E+05 | 1.0729E+06 | 7.1529E+04 | 5.4310E+04 |

| | | | | | | |
|---|---|---|---|---|---|---|
| SCA | 4.4614E+08 | 1.7742E+08 | 1.5968E+05 | 6.7378E+04 | 1.3869E+07 | 1.4377E+07 |
| HHO | 4.6237E+05 | 6.7301E+05 | 4.6630E+04 | 4.3951E+04 | 5.5770E+04 | 3.2295E+04 |
| MFO | 4.6197E+07 | 1.9297E+08 | 1.3446E+05 | 3.0942E+05 | 5.9342E+04 | 7.7794E+04 |
| JAYA | 7.1447E+06 | 8.5011E+06 | 7.8174E+04 | 3.7041E+04 | 5.0073E+06 | 3.5533E+06 |
| FA | 6.0412E+08 | 1.8395E+08 | 1.8178E+05 | 7.4326E+04 | 5.4324E+07 | 2.7645E+07 |
| RFO | 2.4666E+05 | 8.2477E+04 | **5.9472E+03** | **3.2379E+03** | 4.7253E+04 | 4.2734E+04 |
| GWO | 7.1581E+05 | 3.4247E+06 | 3.5769E+05 | 4.4808E+05 | 4.4043E+05 | 8.5266E+05 |
| BA | 2.9030E+05 | 1.1951E+05 | 6.8898E+03 | 3.5489E+03 | 1.1333E+05 | 7.4812E+04 |
| | F16 | | F17 | | F18 | |
| | AVG | STD | AVG | STD | AVG | STD |
| RIME | **2.3576E+03** | 2.9134E+02 | 2.1068E+03 | 2.0841E+02 | 2.6204E+05 | 2.2238E+05 |
| PSO | 2.9371E+03 | 2.2493E+02 | 2.3422E+03 | 2.2483E+02 | 2.2434E+05 | 1.2218E+05 |
| WOA | 3.5541E+03 | 4.8106E+02 | 2.5624E+03 | 3.0007E+02 | 1.9549E+06 | 2.4697E+06 |
| SCA | 3.5168E+03 | 1.8627E+02 | 2.4102E+03 | 1.7040E+02 | 3.2410E+06 | 2.0991E+06 |
| HHO | 3.3004E+03 | 3.5712E+02 | 2.5555E+03 | 2.4492E+02 | 1.1565E+06 | 1.2753E+06 |
| MFO | 3.2299E+03 | 4.7370E+02 | 2.5450E+03 | 2.5851E+02 | 2.6155E+06 | 6.2305E+06 |
| JAYA | 3.4839E+03 | 2.2796E+02 | 2.2719E+03 | 1.2107E+02 | 1.5997E+06 | 7.5998E+05 |
| FA | 3.4792E+03 | **1.7379E+02** | 2.5219E+03 | **9.8800E+01** | 3.6239E+06 | 1.5653E+06 |
| RFO | 3.4554E+03 | 3.8052E+02 | 2.9013E+03 | 2.4925E+02 | **1.7542E+05** | **1.1282E+05** |
| GWO | 2.4374E+03 | 2.8542E+02 | **2.0152E+03** | 1.7117E+02 | 6.9407E+05 | 1.0938E+06 |
| BA | 3.5269E+03 | 3.9964E+02 | 2.9396E+03 | 3.1693E+02 | 2.0367E+05 | 1.6370E+05 |
| | F19 | | F20 | | F21 | |
| | AVG | STD | AVG | STD | AVG | STD |
| RIME | **1.5373E+04** | **1.2402E+04** | 2.3669E+03 | 1.7074E+02 | 2.3901E+03 | 2.4583E+01 |
| PSO | 1.2475E+06 | 6.9784E+05 | 2.6641E+03 | 2.1226E+02 | 2.5358E+03 | 3.0860E+01 |
| WOA | 3.3182E+06 | 2.2738E+06 | 2.7634E+03 | 1.8496E+02 | 2.5622E+03 | 6.1058E+01 |
| SCA | 2.5395E+07 | 1.2434E+07 | 2.5986E+03 | 1.0194E+02 | 2.5463E+03 | 2.5507E+01 |
| HHO | 3.2830E+05 | 2.3979E+05 | 2.6562E+03 | 1.6920E+02 | 2.5420E+03 | 3.6672E+01 |
| MFO | 3.8355E+07 | 6.7784E+07 | 2.8191E+03 | 2.3722E+02 | 2.5091E+03 | 5.2703E+01 |
| JAYA | 1.0494E+06 | 9.4399E+05 | 2.5746E+03 | 9.2331E+01 | 2.5209E+03 | 1.7669E+01 |
| FA | 9.6165E+07 | 4.4732E+07 | 2.5791E+03 | **8.7855E+01** | 2.5356E+03 | **1.3861E+01** |
| RFO | 3.5194E+05 | 1.1079E+05 | 3.0401E+03 | 2.1264E+02 | 2.7028E+03 | 5.9751E+01 |
| GWO | 1.5936E+06 | 5.8665E+06 | **2.3647E+03** | 1.3992E+02 | **2.3834E+03** | 1.7447E+01 |
| BA | 5.1032E+05 | 2.2591E+05 | 2.9427E+03 | 2.3784E+02 | 2.6243E+03 | 6.7220E+01 |
| | F22 | | F23 | | F24 | |
| | AVG | STD | AVG | STD | AVG | STD |
| RIME | 4.2239E+03 | 1.5682E+03 | **2.7365E+03** | 1.8629E+01 | 2.9257E+03 | 2.9612E+01 |
| PSO | 5.2287E+03 | 2.7456E+03 | 3.0874E+03 | 1.3678E+02 | 3.2057E+03 | 1.2571E+02 |
| WOA | 6.9475E+03 | 2.2223E+03 | 3.0387E+03 | 9.9496E+01 | 3.1911E+03 | 9.0082E+01 |
| SCA | 8.0561E+03 | 2.5967E+03 | 2.9908E+03 | 2.4687E+01 | 3.1644E+03 | 3.1794E+01 |
| HHO | 6.6730E+03 | 1.7111E+03 | 3.1100E+03 | 9.0558E+01 | 3.4388E+03 | 1.4129E+02 |
| MFO | 6.3740E+03 | 1.4784E+03 | 2.8358E+03 | 3.7546E+01 | 2.9920E+03 | 3.2696E+01 |
| JAYA | **2.8174E+03** | **1.0209E+02** | 2.9738E+03 | 2.7612E+01 | 3.1217E+03 | 2.3897E+01 |
| FA | 3.8185E+03 | 1.4241E+02 | 2.9138E+03 | **1.2340E+01** | 3.0676E+03 | **8.1560E+00** |
| RFO | 7.6364E+03 | 1.0924E+03 | 3.5217E+03 | 1.8542E+02 | 3.6856E+03 | 1.2303E+02 |
| GWO | 4.2857E+03 | 1.5323E+03 | 2.7550E+03 | 3.2269E+01 | **2.9226E+03** | 4.5536E+01 |
| BA | 7.2071E+03 | 1.1670E+03 | 3.3206E+03 | 1.3970E+02 | 3.3316E+03 | 1.1799E+02 |
| | F25 | | F26 | | F27 | |
| | AVG | STD | AVG | STD | AVG | STD |
| RIME | 2.8927E+03 | 1.5766E+01 | **4.4899E+03** | 5.1474E+02 | 3.2181E+03 | **1.0031E+01** |
| PSO | 2.8993E+03 | 2.1279E+01 | 4.9571E+03 | 1.8904E+03 | **3.2074E+03** | 9.8242E+01 |

| | | | | | | |
|---|---|---|---|---|---|---|
| WOA | 2.9404E+03 | 2.9837E+01 | 7.7229E+03 | 1.0704E+03 | 3.3535E+03 | 8.2366E+01 |
| SCA | 3.1891E+03 | 7.2299E+01 | 6.8750E+03 | 2.5913E+02 | 3.3947E+03 | 4.4983E+01 |
| HHO | 2.9094E+03 | 2.2130E+01 | 7.1835E+03 | 1.3243E+03 | 3.3723E+03 | 9.3578E+01 |
| MFO | 3.4128E+03 | 4.9598E+02 | 5.9231E+03 | 5.4183E+02 | 3.2573E+03 | 2.3904E+01 |
| JAYA | 2.9688E+03 | 3.4983E+01 | 6.4129E+03 | 1.1156E+03 | 3.3425E+03 | 2.8685E+01 |
| FA | 3.5915E+03 | 1.0366E+02 | 6.5039E+03 | **1.7301E+02** | 3.3328E+03 | 1.4703E+01 |
| RFO | **2.8851E+03** | **2.5985E+00** | 8.9678E+03 | 2.8832E+03 | 4.0247E+03 | 4.2215E+02 |
| GWO | 2.9860E+03 | 8.3395E+01 | 4.6921E+03 | 3.4722E+02 | 3.2415E+03 | 2.2852E+01 |
| BA | 2.9083E+03 | 2.2836E+01 | 8.7783E+03 | 2.2941E+03 | 3.4475E+03 | 1.3312E+02 |
| | F28 | | F29 | | F30 | |
| | AVG | STD | AVG | STD | AVG | STD |
| RIME | 3.2196E+03 | 3.0021E+01 | **3.6373E+03** | 1.7649E+02 | **1.9725E+04** | **1.6175E+04** |
| PSO | 3.2434E+03 | **2.4562E+01** | 4.2038E+03 | 2.7344E+02 | 3.4727E+06 | 1.1923E+06 |
| WOA | 3.3088E+03 | 3.4525E+01 | 4.6856E+03 | 3.9778E+02 | 1.3140E+07 | 7.7038E+06 |
| SCA | 3.8213E+03 | 1.4647E+02 | 4.6475E+03 | 2.3773E+02 | 7.7593E+07 | 2.5040E+07 |
| HHO | 3.2477E+03 | 2.7953E+01 | 4.3875E+03 | 3.3960E+02 | 1.6623E+06 | 8.2072E+05 |
| MFO | 4.3886E+03 | 8.6819E+02 | 4.1165E+03 | 2.4830E+02 | 1.1625E+06 | 3.3423E+06 |
| JAYA | 3.5723E+03 | 5.1373E+01 | 4.4949E+03 | **1.4332E+02** | 1.4591E+07 | 4.3065E+06 |
| FA | 3.8878E+03 | 8.2484E+01 | 4.6902E+03 | 1.6766E+02 | 8.6619E+07 | 3.5259E+07 |
| RFO | 3.1636E+03 | 5.1017E+01 | 4.8604E+03 | 4.2929E+02 | 1.3091E+06 | 5.9054E+05 |
| GWO | 3.4274E+03 | 1.7747E+02 | 3.7504E+03 | 1.7169E+02 | 7.9304E+06 | 5.5896E+06 |
| BA | **3.1329E+03** | 5.7041E+01 | 5.0284E+03 | 6.2577E+02 | 1.0898E+06 | 7.2594E+05 |

We further investigated the experimental data using Wilcoxon signed-rank test and Friedman test in order to demonstrate the validity of the RIME experimental results and improve the comprehension of the experimental results. The Wilcoxon signed-rank results are shown in Table 4, where '+' denotes that RIME performs better than other algorithms, '-' denotes that RIME performs worse than other algorithms, '=' denotes that RIME performs equally well as other algorithms, Mean denotes the average ranking following 30 iterations of parallelization, and Rank denotes the overall final ranking. According to Table 4's findings, RIME outperforms the competition in at least 23 out of 30 function assessment trials. The average rating demonstrates that RIME outperforms its rival algorithms by a wide margin. Moreover, the RIME algorithm has tremendous advantages over HHO, WOA, JAYA, and other popular algorithms in recent years.

Figure 8 displays the results of the Friedman test, which is also utilized for comparison analysis with the Wilcoxon test in order to further confirm the validity of the Wilcoxon signed-rank test and the accuracy of the experiments. Figure 8 illustrates that both approaches are useful for validating experiment findings, despite a tiny variance in the average ranking of the Friedman and Wilcoxon test and essentially identical rankings for each algorithm. After the two testing approaches, it is clear that RIME is a very effective metaheuristic optimization algorithm and outperforms the new algorithm of its competitors.

**Table 4.** Wilcoxon signed-rank test results of RIME and classic algorithms

| Algorithms | +/−/= | Mean | Rank |
|---|---|---|---|
| RIME | ~ | 1.77 | 1 |
| PSO | 23/2/5 | 4.93 | 3 |
| WOA | 30/0/0 | 7.47 | 9 |
| SCA | 30/0/0 | 8.27 | 11 |
| HHO | 30/0/0 | 5.53 | 4 |

| | | | |
|---|---|---|---|
| MFO | 30/0/0 | 6.67 | 8 |
| JAYA | 29/0/1 | 6.13 | 5 |
| FA | 29/1/0 | 8.10 | 10 |
| RFO | 23/4/3 | 6.23 | 6 |
| GWO | 23/0/7 | 4.33 | 2 |
| BA | 23/5/2 | 6.47 | 7 |

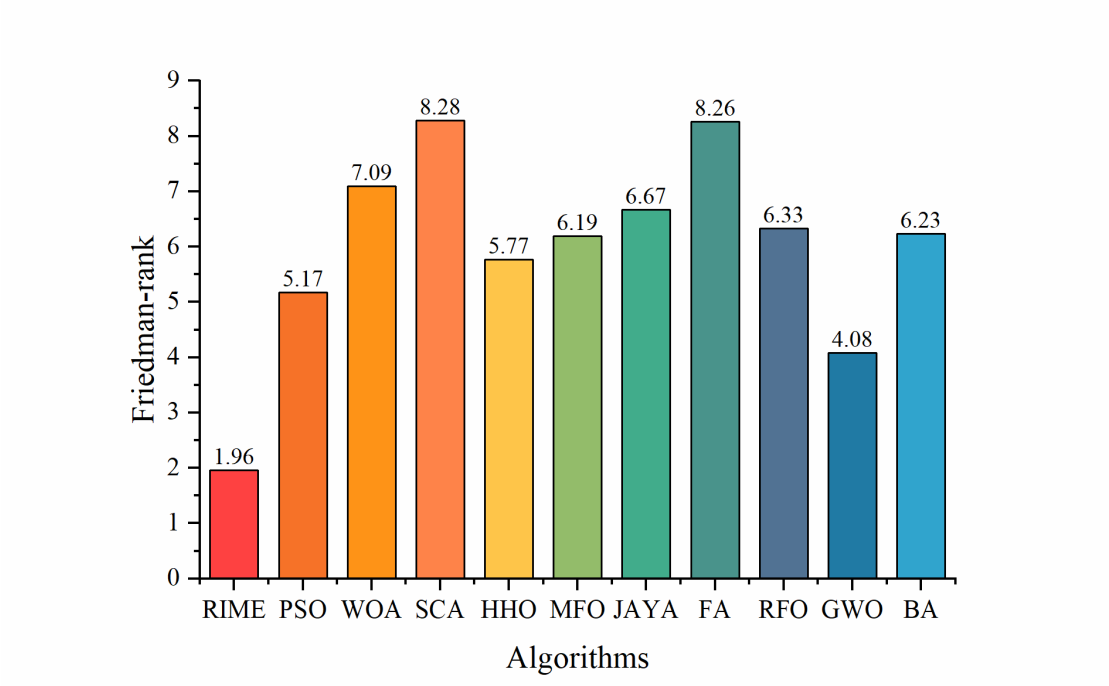



**Figure 8.** Friedman test results of RIME and classic algorithms

The fitness values produced from each evaluation are documented in this work, and the findings are displayed in Figure 9 to highlight the trend of the RIME algorithm's fitness values in comparison to other comparable algorithms. To analyze the iterative process of the algorithm as thoroughly as possible, the 12 functions in the picture are selected from a set of 30 functions in the following order: Unimodal Functions, Multimodal Functions, Hybrid Functions, and Composition Functions. Figure 9 illustrates how RIME outperforms other algorithms in pre-searching for functions F5, F6, F7, F9, and F19. On the functions of F13, F16, and F30, it is clear that RIME updates to a superior solution even in the late stages of the optimization search, demonstrating that it has a strong exploitation potential. Additionally, RIME outperforms competing algorithms in the F1, F13, and F26 functions' exploration and exploitation phases. Thus, by integrating the aforementioned experimental findings, it can be demonstrated that RIME is a metaheuristic optimization algorithm with strong performance and has robust exploration and exploitation capabilities.

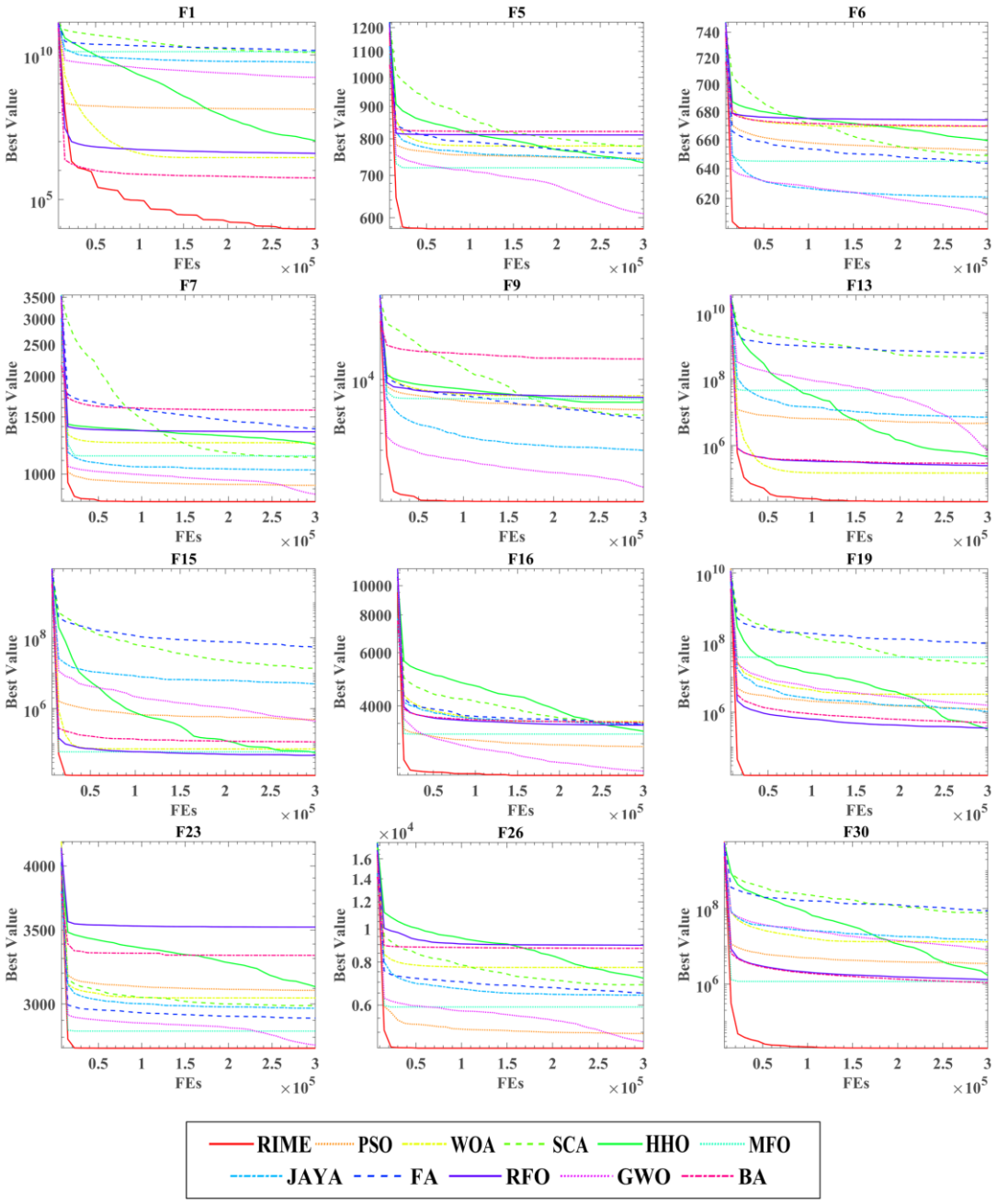



**Figure 9.** Convergence curves of RIME and excellent algorithms

To illustrate that the time complexity of the RIME algorithm is acceptable, in this experiment, we also selected F1, F4, F11, and F12 from the four classes of test functions to fully evaluate the time spent by RIME in terms of CPU runtime records. Figure 10 shows the experiment results, and it can be seen that the time spent by the RIME algorithm is at the same level as most of its peer algorithms, such as PSO and SCA, for both simple and complex functions and is much better than Jaya and FA. It can be said that the time complexity of the RIME algorithm is at a normal level and completely acceptable. Combining the above algorithm comparison experiments and time complexity experiments, the RIME algorithm performs better at the same level of computational complexity and is an excellent algorithm with very high efficiency in finding the best.

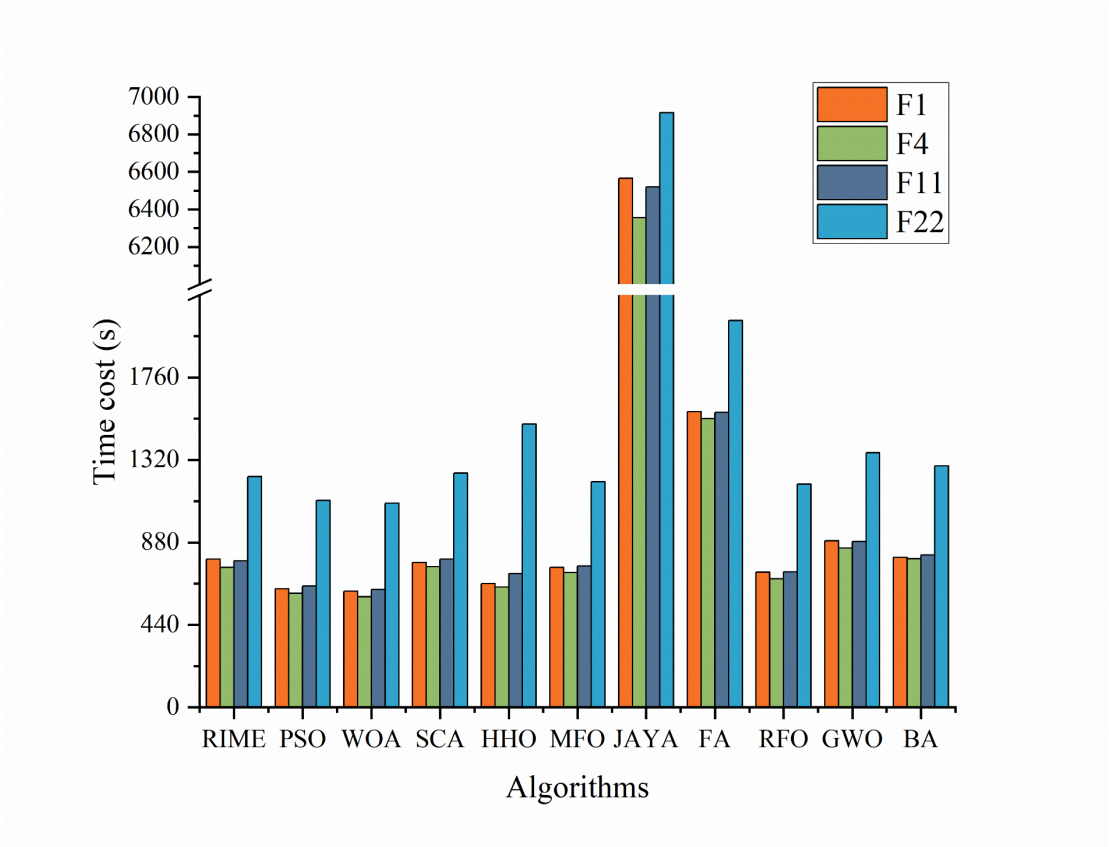



**Figure 10.** Time spent for each algorithm

### 4.2.2 High-performance algorithms comparison and analysis

To further verify the optimization performance of RIME and demonstrate the innovation and superiority of RIME, this section compares RIME with 10 high-performance improvement algorithms. The compared algorithms are the latest optimization algorithms in the last five years, which include: enhanced GWO with a new hierarchical structure (IGWO) [57], opposition-based sine cosine algorithm (OBSCA) [58], double adaptive random spare reinforced whale optimization algorithm (RDWOA) [59], fruit fly optimizer (FOA) with multi-population outpost mechanism (MOFOA) [60], boosted GWO (OBLGWO) [61], hybrid bat algorithm (RCBA) [62], A-C parametric WOA (ACWOA) [63], modified sine cosine algorithm (m_SCA) [64], sine cosine algorithm with differential evolution (SCADE) [65] and balanced whale optimization algorithm (BWOA) [66]. Again, the test set is used IEEE CEC2017, the population size of all algorithms is set to 30, the number of evaluations is 30,000, and they run independently 30 times. Table 5 gives the detailed parameters of the improved algorithms.

**Table 5.** Key parameters of the high-performance algorithms

| Algorithms | Key parameters |
|---|---|
| IGWO | $\beta_{num} = 10;\ \Omega_{num} = 15$ |
| OBSCA | $a = 2$ |
| RDWOA | $S = 0$ |

| | |
|---|---|
| MOFOA | $M = 3$ |
| OBLGWO | $P = 0.5$ |
| RCBA | $Qmin = 0; Qmax = 2; r = 0.5$ |
| ACWOA | $a_1 = [2,0]; a_2 = [-2,-1]; b = 1$ |
| m_SCA | $JR = 0.1;\ a = 2;\ SR =$ random in [0,1] |
| SCADE | $a = 2;\ F =$ random in $[0.2, 0.8];\ P_c = 0.8$ |
| BWOA | $m = 2500$ |

Table 6 displays the precise findings of the comparative trials, with bolded data denoting the best outcomes of the same category. Table 6 shows that RIME does very well on the large majority of the test functions. RIME has a higher AVG in comparison to other high-performance algorithms, which suggests that RIME performs optimizations well. Additionally, RIME performs well in STD, demonstrating that it is more stable than other algorithms introduced in recent years in optimizing the same class of problems.

**Table 6.** Comparison results of RIME and high-performance algorithms

| | F1 | | F2 | | F3 | |
|---|---|---|---|---|---|---|
| | AVG | STD | AVG | STD | AVG | STD |
| RIME | **5.8376E+03** | 6.4529E+03 | 2.1910E+02 | 4.9314E+01 | **3.0013E+02** | **5.5803E-02** |
| IGWO | 1.3989E+06 | 6.9537E+05 | 9.8063E+12 | 1.6862E+13 | 1.3858E+03 | 7.3587E+02 |
| OBSCA | 1.6675E+10 | 3.0109E+09 | 6.1079E+35 | 1.5159E+36 | 6.0471E+04 | 7.5002E+03 |
| RDWOA | 5.9995E+06 | 9.7611E+06 | 2.3378E+16 | 5.0855E+16 | 2.1105E+04 | 5.1995E+03 |
| MOFOA | 5.2108E+10 | 2.4626E+09 | 1.8926E+47 | 9.7268E+47 | 8.0549E+04 | 2.3698E+03 |
| OBLGWO | 1.3839E+07 | 1.0539E+07 | 4.6568E+16 | 1.8456E+17 | 2.0004E+04 | 5.7670E+03 |
| RCBA | 1.7511E+04 | **5.1644E+03** | **2.0000E+02** | **1.3353E-03** | 3.0103E+02 | 2.5320E-01 |
| ACWOA | 5.1816E+09 | 2.5997E+09 | 1.6986E+33 | 6.1741E+33 | 4.8553E+04 | 9.6579E+03 |
| m_SCA | 6.6292E+09 | 2.3633E+09 | 4.0891E+32 | 2.0530E+33 | 2.6172E+04 | 7.3251E+03 |
| SCADE | 2.0052E+10 | 2.4986E+09 | 7.6933E+37 | 2.4719E+38 | 6.0489E+04 | 6.1191E+03 |
| BWOA | 1.5777E+08 | 9.7275E+07 | 1.6784E+25 | 6.4473E+25 | 5.3285E+04 | 8.8340E+03 |
| | F4 | | F5 | | F6 | |
| | AVG | STD | AVG | STD | AVG | STD |
| RIME | 4.8772E+02 | **1.5712E+01** | **5.7639E+02** | 2.2909E+01 | **6.0059E+02** | **5.8799E-01** |
| IGWO | 5.0217E+02 | 3.0400E+01 | 6.2323E+02 | 2.8940E+01 | 6.2325E+02 | 5.1581E+00 |
| OBSCA | 2.6869E+03 | 9.1507E+02 | 8.0199E+02 | 2.0290E+01 | 6.5620E+02 | 4.6450E+00 |
| RDWOA | 5.2527E+02 | 3.1569E+01 | 6.8649E+02 | 4.2708E+01 | 6.1298E+02 | 5.3047E+00 |
| MOFOA | 1.3203E+04 | 1.3283E+03 | 9.5427E+02 | **1.6627E+01** | 7.0279E+02 | 6.7661E+00 |
| OBLGWO | 5.2535E+02 | 4.0151E+01 | 6.6382E+02 | 3.5159E+01 | 6.2125E+02 | 1.6160E+01 |
| RCBA | **4.8674E+02** | 2.8815E+01 | 8.2745E+02 | 6.9732E+01 | 6.7010E+02 | 1.0543E+01 |
| ACWOA | 1.2482E+03 | 7.3025E+02 | 8.0811E+02 | 3.0716E+01 | 6.6640E+02 | 7.6102E+00 |
| m_SCA | 7.0641E+02 | 1.1813E+02 | 6.5511E+02 | 2.7691E+01 | 6.2693E+02 | 6.7888E+00 |
| SCADE | 3.8498E+03 | 1.0234E+03 | 8.2457E+02 | 2.0371E+01 | 6.6019E+02 | 6.3195E+00 |
| BWOA | 6.0688E+02 | 6.6016E+01 | 7.8063E+02 | 3.1820E+01 | 6.6483E+02 | 6.5169E+00 |
| | F7 | | F8 | | F9 | |
| | AVG | STD | AVG | STD | AVG | STD |
| RIME | **7.9963E+02** | **2.1184E+01** | **8.7700E+02** | 2.1810E+01 | **1.0696E+03** | **3.7541E+02** |
| IGWO | 8.9658E+02 | 4.0958E+01 | 8.9369E+02 | 2.4882E+01 | 2.9025E+03 | 1.0144E+03 |
| OBSCA | 1.1758E+03 | 3.0053E+01 | 1.0683E+03 | 1.5228E+01 | 6.4394E+03 | 1.0198E+03 |

| | | | | | | |
|---|---|---|---|---|---|---|
| RDWOA | 9.8240E+02 | 6.0061E+01 | 9.7349E+02 | 3.6018E+01 | 5.0806E+03 | 1.2084E+03 |
| MOFOA | 1.3970E+03 | 2.2965E+01 | 1.1591E+03 | **1.5023E+01** | 1.3368E+04 | 1.2527E+03 |
| OBLGWO | 9.1861E+02 | 6.1457E+01 | 9.5468E+02 | 3.1462E+01 | 2.4861E+03 | 1.2079E+03 |
| RCBA | 1.8733E+03 | 3.6096E+02 | 1.0579E+03 | 6.0709E+01 | 7.9668E+03 | 2.2970E+03 |
| ACWOA | 1.2376E+03 | 6.2383E+01 | 1.0159E+03 | 2.4117E+01 | 6.8550E+03 | 7.0060E+02 |
| m_SCA | 9.9897E+02 | 4.6377E+01 | 9.3181E+02 | 2.5586E+01 | 3.2791E+03 | 8.3761E+02 |
| SCADE | 1.1783E+03 | 3.5463E+01 | 1.0811E+03 | 1.8776E+01 | 8.4005E+03 | 1.4170E+03 |
| BWOA | 1.2496E+03 | 7.8099E+01 | 9.9055E+02 | 2.3064E+01 | 6.3508E+03 | 8.5353E+02 |
| | F10 | | F11 | | F12 | |
| | AVG | STD | AVG | STD | AVG | STD |
| RIME | **3.9839E+03** | 6.6878E+02 | **1.2391E+03** | 4.6442E+01 | 2.2284E+06 | 2.2159E+06 |
| IGWO | 4.8145E+03 | 7.8161E+02 | 1.2606E+03 | **3.5025E+01** | 1.2798E+07 | 8.7528E+06 |
| OBSCA | 7.3083E+03 | 4.0009E+02 | 2.5741E+03 | 4.7116E+02 | 2.0915E+09 | 6.7000E+08 |
| RDWOA | 4.8260E+03 | 6.5146E+02 | 1.2434E+03 | 4.3310E+01 | 3.2776E+06 | 2.0082E+06 |
| MOFOA | 8.7002E+03 | **2.5087E+02** | 8.4740E+03 | 7.6555E+02 | 1.7508E+10 | 1.9279E+09 |
| OBLGWO | 5.3387E+03 | 8.1183E+02 | 1.2995E+03 | 4.9329E+01 | 1.9103E+07 | 1.3096E+07 |
| RCBA | 5.8131E+03 | 6.0098E+02 | 1.3123E+03 | 7.8185E+01 | **1.7800E+06** | **1.3121E+06** |
| ACWOA | 6.6610E+03 | 8.9372E+02 | 3.1592E+03 | 1.0195E+03 | 6.1396E+08 | 4.4073E+08 |
| m_SCA | 4.8807E+03 | 6.3746E+02 | 1.6609E+03 | 4.9275E+02 | 2.5046E+08 | 2.1138E+08 |
| SCADE | 8.1992E+03 | 2.8308E+02 | 3.3444E+03 | 6.0852E+02 | 1.9867E+09 | 5.0948E+08 |
| BWOA | 6.4210E+03 | 8.3835E+02 | 1.7806E+03 | 2.8912E+02 | 9.7624E+07 | 9.9516E+07 |
| | F13 | | F14 | | F15 | |
| | AVG | STD | AVG | STD | AVG | STD |
| RIME | 2.9856E+04 | 2.4760E+04 | 1.3001E+04 | 7.8818E+03 | 1.1817E+04 | 1.1376E+04 |
| IGWO | 1.7023E+05 | 1.8778E+05 | 5.2894E+04 | 3.8472E+04 | 4.8861E+04 | 3.0935E+04 |
| OBSCA | 5.5072E+08 | 2.4407E+08 | 2.5199E+05 | 1.6271E+05 | 1.1552E+07 | 1.1375E+07 |
| RDWOA | **1.9858E+04** | **1.9627E+04** | 9.0743E+04 | 8.0830E+04 | **8.4086E+03** | **8.2873E+03** |
| MOFOA | 1.8822E+10 | 3.8288E+09 | 3.6276E+06 | 2.3694E+06 | 5.7079E+08 | 1.6863E+08 |
| OBLGWO | 1.8567E+05 | 1.1389E+05 | 6.6703E+04 | 5.4086E+04 | 8.1110E+04 | 4.8503E+04 |
| RCBA | 1.2279E+05 | 9.7040E+04 | **6.9868E+03** | **3.4648E+03** | 3.7671E+04 | 2.5330E+04 |
| ACWOA | 9.5806E+07 | 1.3412E+08 | 1.0400E+06 | 7.9027E+05 | 1.0343E+07 | 1.2890E+07 |
| m_SCA | 7.5740E+07 | 1.1679E+08 | 6.5479E+04 | 7.6512E+04 | 3.5697E+05 | 8.2313E+05 |
| SCADE | 6.9404E+08 | 2.5238E+08 | 3.1116E+05 | 1.6635E+05 | 7.9280E+06 | 8.5614E+06 |
| BWOA | 2.3145E+05 | 1.0493E+05 | 1.0928E+06 | 1.2286E+06 | 1.2977E+05 | 1.8734E+05 |
| | F16 | | F17 | | F18 | |
| | AVG | STD | AVG | STD | AVG | STD |
| RIME | **2.2711E+03** | **2.2170E+02** | 2.1075E+03 | 1.9261E+02 | **1.6290E+05** | 1.2837E+05 |
| IGWO | 2.5519E+03 | 2.9638E+02 | **1.9955E+03** | 1.5673E+02 | 6.2444E+05 | 4.8305E+05 |
| OBSCA | 3.8369E+03 | 2.2585E+02 | 2.6360E+03 | 1.5899E+02 | 2.9471E+06 | 1.7147E+06 |
| RDWOA | 2.7996E+03 | 2.5788E+02 | 2.2662E+03 | 2.4641E+02 | 7.2215E+05 | 1.0537E+06 |
| MOFOA | 6.3529E+03 | 5.8327E+02 | 5.4192E+03 | 1.2279E+03 | 3.8450E+07 | 1.2148E+07 |
| OBLGWO | 2.8484E+03 | 3.3815E+02 | 2.2020E+03 | 1.9019E+02 | 1.0999E+06 | 7.1909E+05 |
| RCBA | 3.3484E+03 | 3.4262E+02 | 2.8846E+03 | 4.2768E+02 | 1.9019E+05 | **1.1156E+05** |
| ACWOA | 3.9149E+03 | 3.2707E+02 | 2.4773E+03 | 2.2359E+02 | 3.0769E+06 | 3.0777E+06 |
| m_SCA | 2.5218E+03 | 3.0428E+02 | 2.0132E+03 | 1.3314E+02 | 9.5746E+05 | 9.9834E+05 |
| SCADE | 3.8650E+03 | 2.3939E+02 | 2.5317E+03 | **1.1903E+02** | 4.0667E+06 | 2.3328E+06 |
| BWOA | 3.7037E+03 | 4.5284E+02 | 2.6189E+03 | 2.6606E+02 | 5.6346E+06 | 5.1778E+06 |
| | F19 | | F20 | | F21 | |
| | AVG | STD | AVG | STD | AVG | STD |
| RIME | 1.6614E+04 | 1.5110E+04 | **2.3203E+03** | 1.5896E+02 | **2.3771E+03** | **1.9593E+01** |
| IGWO | 3.7603E+05 | 3.2615E+05 | 2.3551E+03 | 1.4154E+02 | 2.3948E+03 | 2.4129E+01 |

| | | | | | | |
|---|---|---|---|---|---|---|
| OBSCA | 4.4403E+07 | 2.5480E+07 | 2.6244E+03 | 1.3004E+02 | 2.4236E+03 | 8.3912E+01 |
| RDWOA | **1.0005E+04** | 1.3451E+04 | 2.4795E+03 | 1.7736E+02 | 2.5007E+03 | 4.8569E+01 |
| MOFOA | 1.0868E+09 | 3.0378E+08 | 3.0519E+03 | 1.1522E+02 | 2.7775E+03 | 2.2862E+01 |
| OBLGWO | 5.0681E+05 | 6.4460E+05 | 2.4566E+03 | 1.7609E+02 | 2.4565E+03 | 3.8460E+01 |
| RCBA | 1.2721E+04 | **8.0851E+03** | 2.9417E+03 | 2.6111E+02 | 2.6205E+03 | 7.1041E+01 |
| ACWOA | 8.6972E+06 | 5.7102E+06 | 2.6706E+03 | 1.8153E+02 | 2.5818E+03 | 3.8522E+01 |
| m_SCA | 1.5388E+06 | 2.3867E+06 | 2.3756E+03 | 1.3278E+02 | 2.4347E+03 | 3.2932E+01 |
| SCADE | 2.7261E+07 | 1.0016E+07 | 2.7079E+03 | **1.0528E+02** | 2.5714E+03 | 3.9982E+01 |
| BWOA | 3.6274E+06 | 3.4117E+06 | 2.6563E+03 | 2.0011E+02 | 2.5887E+03 | 5.0138E+01 |
| | F22 | | F23 | | F24 | |
| | AVG | STD | AVG | STD | AVG | STD |
| RIME | 4.3050E+03 | 1.5896E+03 | **2.7244E+03** | **2.1135E+01** | **2.9052E+03** | **2.2829E+01** |
| IGWO | **2.3108E+03** | **1.6884E+00** | 2.7806E+03 | 2.6002E+01 | 2.9449E+03 | 2.8086E+01 |
| OBSCA | 4.1421E+03 | 3.5241E+02 | 3.0191E+03 | 3.4235E+01 | 3.1814E+03 | 3.9838E+01 |
| RDWOA | 6.3287E+03 | 1.3231E+03 | 2.8714E+03 | 5.8665E+01 | 3.1316E+03 | 7.9750E+01 |
| MOFOA | 9.0719E+03 | 4.2978E+02 | 3.7244E+03 | 1.6932E+02 | 3.8952E+03 | 1.8405E+02 |
| OBLGWO | 2.9586E+03 | 1.6334E+03 | 2.8178E+03 | 5.2717E+01 | 2.9719E+03 | 4.1930E+01 |
| RCBA | 7.4649E+03 | 8.9248E+02 | 3.3384E+03 | 1.9761E+02 | 3.3938E+03 | 2.1785E+02 |
| ACWOA | 5.0141E+03 | 2.3420E+03 | 3.0647E+03 | 6.4532E+01 | 3.1875E+03 | 8.7515E+01 |
| m_SCA | 5.2543E+03 | 1.5849E+03 | 2.7954E+03 | 2.9614E+01 | 2.9681E+03 | 4.2059E+01 |
| SCADE | 4.5540E+03 | 4.1106E+02 | 3.0043E+03 | 2.7426E+01 | 3.1688E+03 | 3.3825E+01 |
| BWOA | 6.3444E+03 | 2.4505E+03 | 3.0898E+03 | 1.1085E+02 | 3.1913E+03 | 9.8830E+01 |
| | F25 | | F26 | | F27 | |
| | AVG | STD | AVG | STD | AVG | STD |
| RIME | **2.8910E+03** | **8.2019E+00** | **4.4704E+03** | **2.1631E+02** | **3.2203E+03** | **1.3727E+01** |
| IGWO | 2.9043E+03 | 1.5166E+01 | 4.7592E+03 | 2.4323E+02 | 3.2355E+03 | 1.3804E+01 |
| OBSCA | 3.3441E+03 | 1.1336E+02 | 6.9962E+03 | 6.6377E+02 | 3.4521E+03 | 4.4524E+01 |
| RDWOA | 2.9196E+03 | 2.7728E+01 | 5.9250E+03 | 1.0540E+03 | 3.2533E+03 | 1.9719E+01 |
| MOFOA | 4.9416E+03 | 1.4000E+02 | 1.0966E+04 | 3.0380E+02 | 4.7214E+03 | 2.8587E+02 |
| OBLGWO | 2.9182E+03 | 2.9059E+01 | 5.3068E+03 | 6.3271E+02 | 3.2428E+03 | 1.9888E+01 |
| RCBA | 2.9014E+03 | 2.4477E+01 | 9.5235E+03 | 1.8368E+03 | 3.4434E+03 | 1.5086E+02 |
| ACWOA | 3.1289E+03 | 3.5826E+01 | 7.5683E+03 | 9.4723E+02 | 3.4330E+03 | 1.1230E+02 |
| m_SCA | 3.0538E+03 | 7.8352E+01 | 5.2734E+03 | 3.4912E+02 | 3.2589E+03 | 2.7386E+01 |
| SCADE | 3.4265E+03 | 8.6793E+01 | 7.4286E+03 | 3.1302E+02 | 3.4541E+03 | 5.2577E+01 |
| BWOA | 3.0109E+03 | 3.7316E+01 | 8.0170E+03 | 1.0923E+03 | 3.3854E+03 | 7.3306E+01 |
| | F28 | | F29 | | F30 | |
| | AVG | STD | AVG | STD | AVG | STD |
| RIME | 3.2151E+03 | 3.9526E+01 | **3.6612E+03** | **1.7953E+02** | **1.8176E+04** | **8.3621E+03** |
| IGWO | 3.2542E+03 | **2.6751E+01** | 3.8293E+03 | 1.9432E+02 | 4.3596E+06 | 2.7233E+06 |
| OBSCA | 4.1871E+03 | 2.2509E+02 | 4.9922E+03 | 2.4776E+02 | 1.2769E+08 | 4.7840E+07 |
| RDWOA | 3.2680E+03 | 2.8965E+01 | 3.9419E+03 | 2.3247E+02 | 2.1042E+04 | 1.0220E+04 |
| MOFOA | 6.8872E+03 | 2.5442E+02 | 8.0642E+03 | 8.6987E+02 | 3.4099E+09 | 1.1149E+09 |
| OBLGWO | 3.2772E+03 | 4.2033E+01 | 4.0081E+03 | 2.5554E+02 | 2.6581E+06 | 1.6303E+06 |
| RCBA | **3.1740E+03** | 6.5055E+01 | 4.9038E+03 | 4.1341E+02 | 1.7922E+05 | 1.0487E+05 |
| ACWOA | 3.7542E+03 | 2.4233E+02 | 4.8017E+03 | 3.2791E+02 | 6.5433E+07 | 4.4222E+07 |
| m_SCA | 3.4832E+03 | 9.2865E+01 | 3.8830E+03 | 1.7973E+02 | 8.1453E+06 | 7.8817E+06 |
| SCADE | 4.3014E+03 | 2.2969E+02 | 5.1153E+03 | 2.4847E+02 | 9.6796E+07 | 3.5657E+07 |
| BWOA | 3.3961E+03 | 5.0166E+01 | 5.1008E+03 | 5.0841E+02 | 2.1122E+07 | 1.4786E+07 |

Similarly, in Table 7, the benchmark function optimization performance between RIME and other classes of algorithms is further analyzed in this paper using the Wilcoxon signed rank test. Further, one can

see that the RIME algorithm has the same large advantage over emerging high-performance optimization algorithms such as SCADE and IGWO, with RIME outperforming the other algorithms in at least 22 of the 30 tested functions.

Figure 11 uses the Friedman test to further assess the experimental outcomes for the benchmark functions. As can be seen, the RIME algorithm continues to receive the highest overall ranking. The examination of the Friedman test and the Wilcoxon signed-rank test demonstrates that RIME still has a significant edge over the most recent algorithms.

**Table 7.** Wilcoxon signed-rank test results of RIME and high-performance algorithms

| Algorithms | +/−/= | Mean | Rank |
|---|---|---|---|
| RIME | ~ | 1.47 | 1 |
| IGWO | 27/2/1 | 2.73 | 2 |
| OBSCA | 29/0/1 | 7.83 | 8 |
| RDWOA | 25/2/3 | 4.03 | 3 |
| MOFOA | 30/0/0 | 10.97 | 11 |
| OBLGWO | 29/1/0 | 4.27 | 4 |
| RCBA | 22/3/5 | 5.80 | 6 |
| ACWOA | 29/0/1 | 7.87 | 9 |
| m_SCA | 28/0/2 | 4.97 | 5 |
| SCADE | 29/0/1 | 8.70 | 10 |
| BWOA | 30/0/0 | 7.37 | 7 |

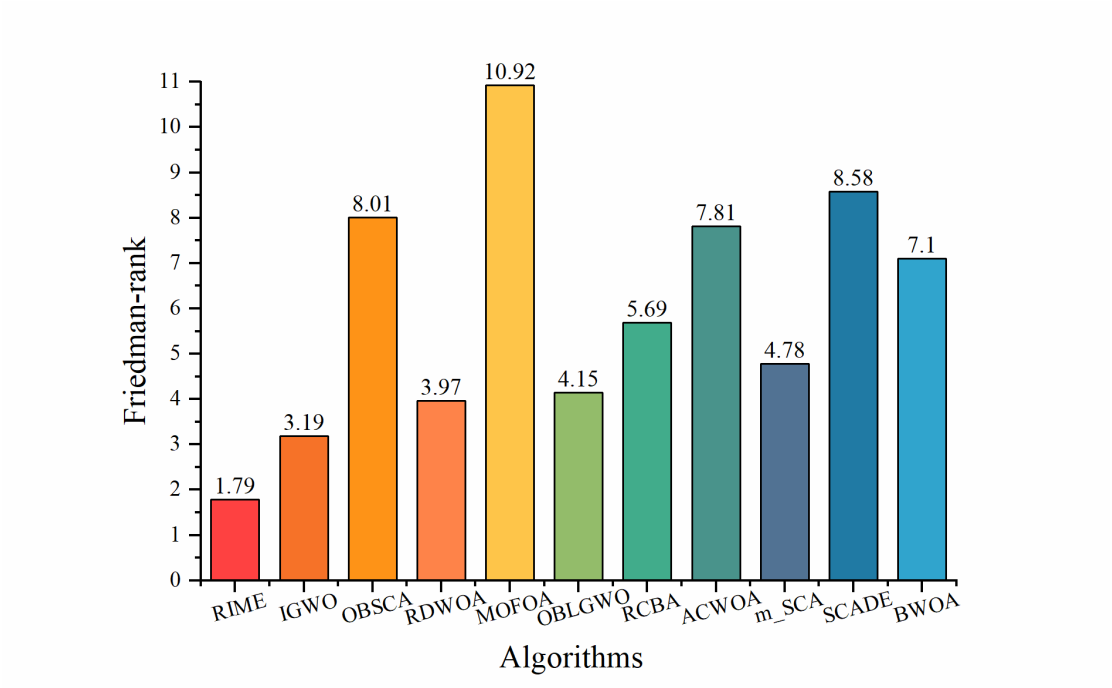



**Figure 11.** Friedman test results of RIME and high-performance algorithms

The convergence curves for RIME and other comparable methods for various functions are shown in Figure 12. As can be seen from the convergence curves, although while RIME converges a little more slowly than the other algorithms in the first stage, it produces solutions of higher quality than the other algorithms

when optimizing the benchmark functions F1, F5, F7, F9, F18, and F30. High-quality solutions are achieved while optimizing F1 and F18, and the algorithm clearly jumps out of the locally optimum inflection point. It can be observed that RIME not only achieves high-quality solutions but also converges more quickly than other algorithms while optimizing F6, F16, F20, F21, F23, and F26. RIME has a strong capacity to leap out of the local optimal solution, a strong ability to find high-quality solutions, and a faster convergence speed, according to the study of the benchmark function results done above.

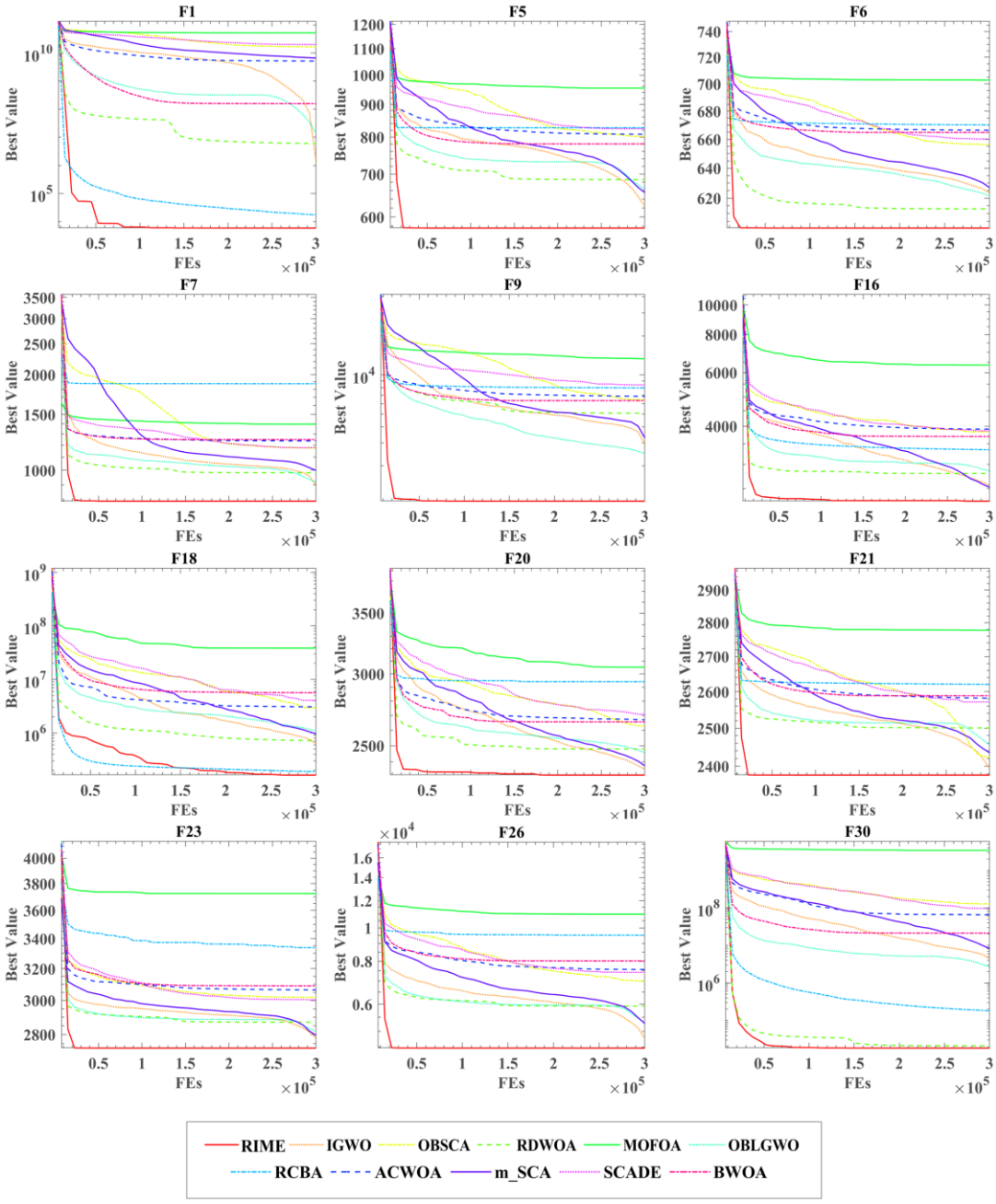



**Figure 12.** Convergence curves of RIME and high-performance algorithms

### 4.2.3 Comparative analysis based on the IEEE CEC 2022 test set

To further demonstrate the novelty and superiority of the RIME algorithm, this subsection uses the latest CEC2022 test set [49] to experiment with the algorithm, as shown in Table 8. This experiment compares the RIME algorithm with the above 10 algorithms, which include five classical algorithms: PSO, GWO, HHO, MFO, WOA, and five improved algorithms: OBSCA, MOFOA, ACWOA, SCADE, and BWOA. In the experiment, the population size is set to 30, the dimension is set to 20 (default value), and the same is run 30 times independently with 200,000 evaluations each time. The key parameters of each algorithm are set as in the above experiments.

**Table 8.** Details of the IEEE CEC2022

| | | Functions | $f_i$ |
|---|---|---|---|
| Unimodal Functions | F1 | Shifted and full Rotated Zakharov Function | 300 |
| Multimodal Functions | F2 | Shifted and full Rotated Rosenbrock's Function | 400 |
| | F3 | Shifted and full Rotated Expanded Schaffer's f6 Function | 600 |
| | F4 | Shifted and full Rotated Non-Continuous Rastrigin's Function | 800 |
| | F5 | Shifted and full Rotated Levy Function | 900 |
| Hybrid Functions | F6 | Hybrid Function 1 (N = 3) | 1800 |
| | F7 | Hybrid Function 2 (N = 6) | 2000 |
| | F8 | Hybrid Function 3 (N = 5) | 2200 |
| Composition Functions | F9 | Composition Function 1 (N = 5) | 2300 |
| | F10 | Composition Function 2 (N = 4) | 2400 |
| | F11 | Composition Function 3 (N = 5) | 2600 |
| | F12 | Composition Function 4 (N = 6) | 2700 |

The average and standard deviation of the experimental results for 30 independent runs are shown in Table 9. It can be seen that the overall performance of the RIME algorithm remains excellent in the state-of-the-art test set. The AVG of the RIME algorithm is better than other algorithms, such as PSO and GWO, which indicates that the algorithm can still maintain a high level of optimization performance for new optimization problems. In addition, the STD of RIME after running is also minimal under most problems, which indicates that the algorithm is also very stable in the face of new problems.

**Table 9.** Comparative results of RIME and peer algorithms in IEEE CEC2022

| | F1 | | F2 | | F3 | |
|---|---|---|---|---|---|---|
| | AVG | STD | AVG | STD | AVG | STD |
| RIME | **3.0000E+02** | **2.7715E-04** | 4.4880E+02 | **1.7606E+01** | **6.0006E+02** | **1.1498E-01** |
| PSO | 3.7700E+02 | 1.2809E+01 | **4.3169E+02** | 2.4364E+01 | 6.3999E+02 | 1.4807E+01 |
| GWO | 7.9510E+03 | 3.6565E+03 | 4.9367E+02 | 3.8953E+01 | 6.0456E+02 | 4.0695E+00 |
| HHO | 3.2773E+02 | 1.4335E+01 | 4.6550E+02 | 2.8768E+01 | 6.5624E+02 | 8.3755E+00 |
| MFO | 2.7600E+04 | 2.2958E+04 | 5.2878E+02 | 1.2760E+02 | 6.2065E+02 | 9.0276E+00 |
| WOA | 2.6265E+03 | 2.2704E+03 | 4.8287E+02 | 3.7936E+01 | 6.6610E+02 | 1.2163E+01 |
| OBSCA | 1.3314E+04 | 3.6982E+03 | 7.4406E+02 | 1.1681E+02 | 6.4179E+02 | 5.3000E+00 |
| MOFOA | 2.3974E+04 | 1.6027E+03 | 2.4477E+03 | 2.9760E+02 | 6.9674E+02 | 7.1910E+00 |

| | | | | | | |
|---|---|---|---|---|---|---|
| ACWOA | 1.4739E+04 | 4.0293E+03 | 6.5030E+02 | 8.2703E+01 | 6.5993E+02 | 7.6142E+00 |
| SCADE | 2.3741E+04 | 5.0809E+03 | 7.8805E+02 | 7.7304E+01 | 6.4445E+02 | 5.6699E+00 |
| BWOA | 8.6304E+03 | 2.1872E+03 | 5.4626E+02 | 5.1570E+01 | 6.6004E+02 | 1.0464E+01 |
| | F4 | | F5 | | F6 | |
| | AVG | STD | AVG | STD | AVG | STD |
| RIME | **8.5127E+02** | 2.0046E+01 | **9.1350E+02** | **4.5060E+01** | 9.7669E+03 | 7.2068E+03 |
| PSO | 8.8342E+02 | 1.3673E+01 | 1.3765E+03 | 6.3557E+02 | 1.5013E+06 | 4.8479E+05 |
| GWO | 8.5186E+02 | 2.1133E+01 | 1.1571E+03 | 2.0110E+02 | 1.3579E+06 | 3.9496E+06 |
| HHO | 8.8755E+02 | 1.4853E+01 | 2.6827E+03 | 2.5926E+02 | 7.3111E+04 | 3.6572E+04 |
| MFO | 8.9933E+02 | 2.6147E+01 | 3.0454E+03 | 1.1593E+03 | 2.1059E+07 | 6.4873E+07 |
| WOA | 9.1585E+02 | 3.7539E+01 | 3.3978E+03 | 1.0776E+03 | **8.9521E+03** | **6.7427E+03** |
| OBSCA | 9.4101E+02 | 9.4386E+00 | 2.3867E+03 | 4.1922E+02 | 4.5139E+07 | 2.9076E+07 |
| MOFOA | 9.7451E+02 | **6.8930E+00** | 3.4482E+03 | 2.3321E+02 | 9.1882E+08 | 9.5393E+07 |
| ACWOA | 9.0576E+02 | 1.3788E+01 | 2.6759E+03 | 3.1354E+02 | 3.4293E+07 | 3.1536E+07 |
| SCADE | 9.4845E+02 | 1.0140E+01 | 2.5913E+03 | 4.3276E+02 | 5.1544E+07 | 3.8983E+07 |
| BWOA | 8.8734E+02 | 1.4280E+01 | 2.5354E+03 | 2.8778E+02 | 1.7800E+05 | 4.3477E+05 |
| | F7 | | F8 | | F9 | |
| | AVG | STD | AVG | STD | AVG | STD |
| RIME | **2.0547E+03** | 3.6427E+01 | **2.2342E+03** | 3.6633E+01 | 2.4808E+03 | **2.5834E-03** |
| PSO | 2.1225E+03 | 3.8797E+01 | 2.2777E+03 | 6.4717E+01 | **2.4658E+03** | 1.3002E-01 |
| GWO | 2.0704E+03 | 4.3252E+01 | 2.2485E+03 | 4.5440E+01 | 2.5128E+03 | 2.9008E+01 |
| HHO | 2.1780E+03 | 6.2871E+01 | 2.2483E+03 | 2.8873E+01 | 2.4868E+03 | 2.9062E+00 |
| MFO | 2.1455E+03 | 5.0221E+01 | 2.2686E+03 | 4.9206E+01 | 2.5161E+03 | 4.5421E+01 |
| WOA | 2.1704E+03 | 5.2707E+01 | 2.2566E+03 | 4.0643E+01 | 2.4933E+03 | 1.2047E+01 |
| OBSCA | 2.1399E+03 | 1.7361E+01 | 2.2619E+03 | 1.4017E+01 | 2.5993E+03 | 3.5360E+01 |
| MOFOA | 2.2760E+03 | 2.9909E+01 | 2.8714E+03 | 3.1774E+02 | 3.2038E+03 | 9.1957E+01 |
| ACWOA | 2.1651E+03 | 1.9616E+01 | 2.2441E+03 | 2.3840E+01 | 2.5935E+03 | 4.1206E+01 |
| SCADE | 2.1554E+03 | **1.6545E+01** | 2.2455E+03 | **3.7302E+00** | 2.5753E+03 | 2.3996E+01 |
| BWOA | 2.1676E+03 | 4.1869E+01 | 2.2632E+03 | 4.9816E+01 | 2.5244E+03 | 2.7121E+01 |
| | F10 | | F11 | | F12 | |
| | AVG | STD | AVG | STD | AVG | STD |
| RIME | 2.6054E+03 | 1.5971E+02 | **2.7001E+03** | 1.4206E+02 | 2.8626E+03 | 2.4832E+00 |
| PSO | 4.4218E+03 | 1.0223E+03 | 2.7234E+03 | 1.7281E+02 | **2.8598E+03** | 3.9629E+01 |
| GWO | 3.1601E+03 | 6.6270E+02 | 2.7765E+03 | 1.5134E+02 | 2.8653E+03 | 4.6883E+00 |
| HHO | 3.2932E+03 | 5.7009E+02 | 2.8177E+03 | 1.4081E+02 | 2.9036E+03 | 5.6494E+01 |
| MFO | 3.8266E+03 | 1.0675E+03 | 2.7688E+03 | 1.4016E+02 | 2.8644E+03 | 1.6924E+00 |
| WOA | 4.1437E+03 | 1.1801E+03 | 2.7836E+03 | 1.6643E+02 | 2.8847E+03 | 2.8977E+01 |
| OBSCA | **2.5307E+03** | **7.8055E+00** | 2.7671E+03 | **9.8306E+00** | 2.8704E+03 | **1.6653E+00** |
| MOFOA | 6.1321E+03 | 1.2652E+03 | 3.3497E+03 | 1.3646E+02 | 2.9390E+03 | 1.5841E+01 |
| ACWOA | 3.3768E+03 | 1.2027E+03 | 2.7995E+03 | 1.6029E+02 | 2.8918E+03 | 3.0487E+01 |
| SCADE | 2.5463E+03 | 9.6190E+00 | 2.7716E+03 | 1.0245E+01 | 2.8694E+03 | 1.6822E+00 |
| BWOA | 4.6024E+03 | 1.1425E+03 | 2.7462E+03 | 1.2524E+02 | 2.8802E+03 | 3.0286E+01 |

Similarly, in Table 10, this paper further analyzes the optimization performance of RIME versus peer algorithms in the latest test set using the Wilcoxon signed-rank test. In the new optimization problem, the RIME algorithm has the same large advantage over peer optimization algorithms such as PSO and GWO, etc. Of the 12 functions tested, RIME outperforms the other algorithms on at least 8 functions.

The Friedman test is used in Figure 13 to further assess the experimental outcomes of the benchmark functions. As can be shown, the RIME algorithm consistently has the highest average ranking. RIME continues to outperform its peers on new problems, as shown by the study of the Friedman test and the Wilcoxon signed-rank test.

**Table 10.** Wilcoxon signed-rank test results of RIME and peer algorithms in CEC2022

| Algorithms | +/−/= | Mean | Rank |
|---|---|---|---|
| RIME | ~ | 1.50 | 1 |
| PSO | 8/3/1 | 3.83 | 2 |
| GWO | 9/0/3 | 4.00 | 3 |
| HHO | 12/0/0 | 5.83 | 4 |
| MFO | 11/0/1 | 6.42 | 6 |
| WOA | 11/0/1 | 6.67 | 8 |
| OBSCA | 11/0/1 | 6.25 | 5 |
| MOFOA | 12/0/0 | 10.92 | 11 |
| ACWOA | 12/0/0 | 7.33 | 10 |
| SCADE | 11/0/1 | 6.75 | 9 |
| BWOA | 11/0/1 | 6.50 | 7 |

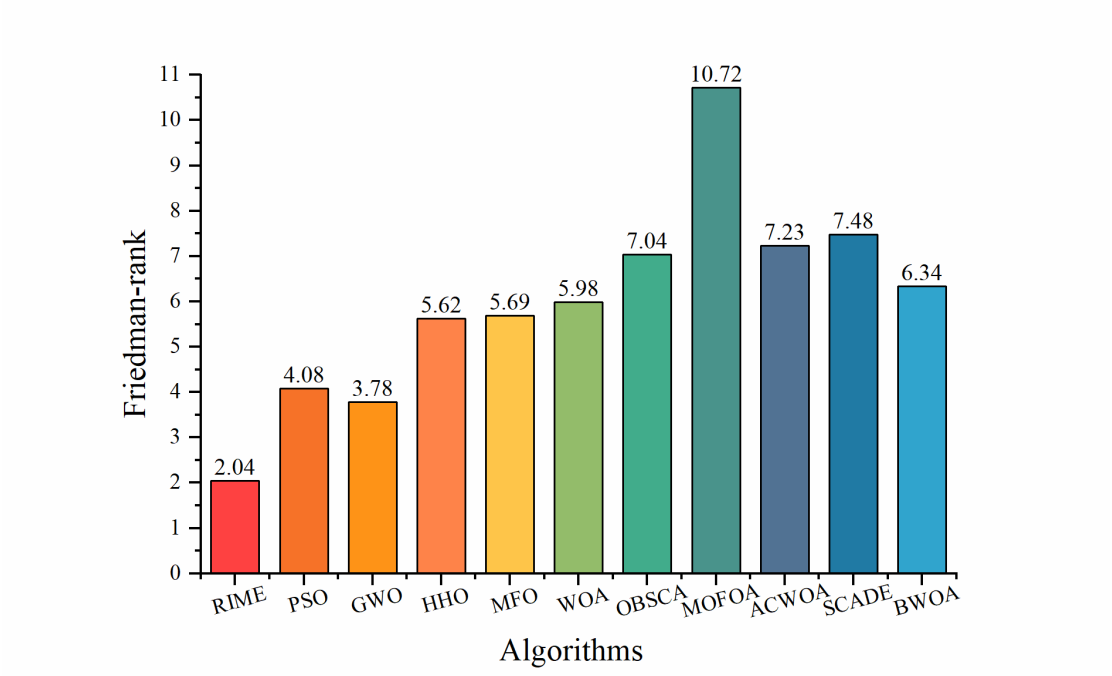



**Figure 13.** Friedman test results of RIME and peer algorithms in CEC2022

Figure 14 shows the convergence curves of the RIME algorithm and the peer algorithm on the latest test set. In the graph of F1, the RIME algorithm has a clear tendency to jump out of the local optimum after 50,000 iterations, and the final convergence accuracy is the highest. In F3, F5, and F7, RIME finds an excellent solution quickly in the initial search stage of the algorithm and keeps developing on this basis, leading to the whole process of finding the best in both precision and speed. In F8 and F11, the optimization problems of a composite type are very complex, but RIME still has the best convergence accuracy. In

summary, the RIME algorithm is a very strong metaheuristic that can maintain high accuracy and efficiency when dealing with the latest optimization problems.

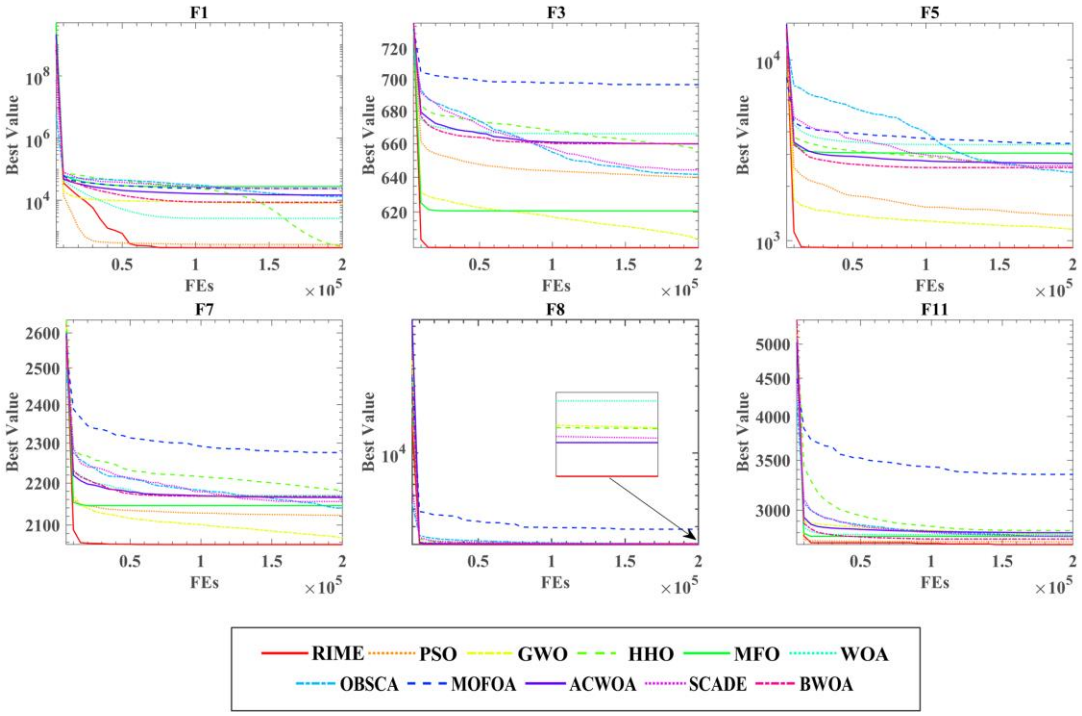



**Figure 14.** Convergence curves of RIME and peer algorithms in CEC2022

## 4.3 Parameter sensitivity analysis of RIME

In this section, experiments and analyses are conducted in terms of both the internal parameters of the algorithm and parameters common to the metaheuristic algorithm. On the one hand, the key internal parameters of the RIME algorithm are discussed and analyzed to determine the optimality and rationality of the key parameters for the proposed algorithm. On the other hand, the parameters common to the swarm intelligence optimization algorithm (including population size, dimension size and evaluation times, etc.) are discussed to set appropriate initial values for RIME under different optimization problems.

### 4.3.1 Analysis of critical parameters

In this section, we change the parameter $w$ in Section 3.2 when the soft-rime mechanism acts. This parameter is used to control the frequency of alternating exploration and exploitation and is an important parameter affecting the algorithm's overall accuracy and stability. Experiments are conducted with parameters $w$ set to 1, 3, 7, and 10 and compared to the original parameter $w$=5 to show the effect of parameter changes in the mechanism on RIME performance. The comparison experiments are performed in a unified evaluation framework with the same number of populations of 30, with 300,000 evaluations and 30 independent parallelization of the algorithm.

Specifically, the value of $w$ directly affects the variation of the rime environment factor $\beta$, as shown in

Figure 15. As $\beta$ is a step-down function, the value of $w$ controls the number of segments of the $\beta$ function. It can be seen that in Figure 15, as the number of evaluations (FEs), the value of $\beta$ decreases in a stepwise fashion, while different values of $w$ have different numbers of segments. When $\beta$ is constant, the algorithm performs a deep exploitation of the current optimal solution; when $\beta$ is decreasing, the algorithm performs a wide range of searches. Therefore, $w$ directly controls the balance of search and exploitation of the algorithm, and it is important to select an appropriate value.

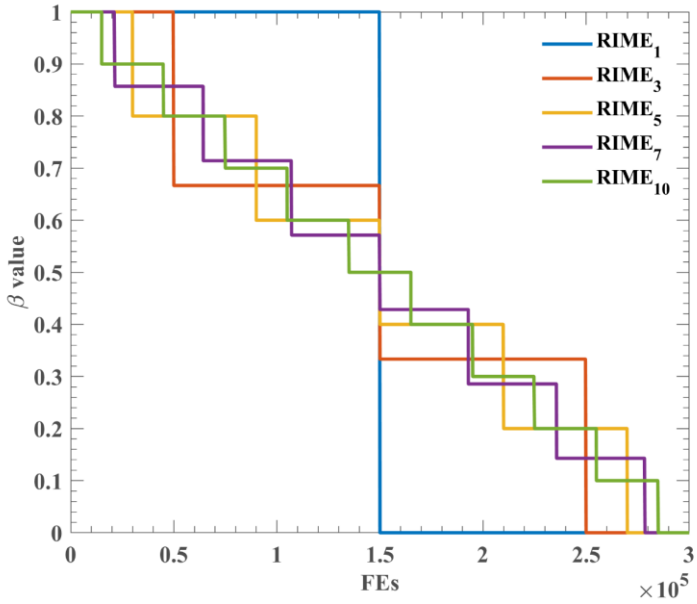



**Figure 15.** Effect of $w$-value on environmental factor $\beta$

Table 11 shows the results of the settings and rankings of the RIME parameters, where $RIME_5$ is ranked first, $RIME_3$ is ranked last, and the average ranking of $RIME_5$ is also larger than the ranking of the algorithms with other values. Table 12 shows the detailed results of RIME under the action of different parameters, and it can be seen that each parameter change has a certain impact on RIME. Although $RIME_5$ is not always the best result, it is stable, so $RIME_5$ does not show poor results in the optimization problem. Therefore, considering the accuracy and stability of the algorithm, the parameter $w$ is set to 5 in this paper.

**Table 11.** RIME's $w$ parameter settings and ranking

| **Algorithms** | $RIME_1$ | $RIME_3$ | $RIME_5$ | $RIME_7$ | $RIME_{10}$ |
|---|---|---|---|---|---|
| $w$-value | 1 | 3 | 5 | 7 | 10 |
| Mean | 3.47 | 3.93 | **2.87** | 3.33 | 3.67 |
| rank | 3 | 6 | **1** | 2 | 4 |

**Table 12.** Detailed results for different $w$ parameter settings

| Fun | Item | $RIME_1$ | $RIME_3$ | $RIME_5$ | $RIME_7$ | $RIME_{10}$ |
|---|---|---|---|---|---|---|
| F1 | AVG | 1.3771E+07 | 2.0826E+07 | 1.7098E+07 | 1.7600E+07 | **1.3348E+07** |
| | STD | **5.2194E+06** | 6.0881E+06 | 7.3208E+06 | 7.8137E+06 | 6.5814E+06 |
| F2 | AVG | 1.0339E+08 | 5.4441E+04 | 8.7692E+04 | 5.8603E+04 | **5.2551E+04** |

| | | | | | | |
|---|---|---|---|---|---|---|
| | STD | 2.3975E+07 | 2.5224E+04 | 3.2832E+04 | **2.5217E+04** | 2.6271E+04 |
| F3 | AVG | 4.7052E+04 | 4.2741E+03 | **4.1681E+03** | 4.6252E+03 | 4.6443E+03 |
| | STD | 1.5622E+04 | 8.5503E+03 | 4.2476E+03 | 3.2594E+03 | **2.3114E+03** |
| F4 | AVG | 4.2549E+03 | 4.1821E+03 | 3.1954E+03 | **5.1532E+02** | 5.4304E+02 |
| | STD | 3.6358E+03 | 3.2760E+03 | 2.4985E+03 | 4.3251E+01 | **3.7873E+01** |
| F5 | AVG | 5.3330E+02 | 5.2508E+02 | 5.2917E+02 | 5.4214E+02 | **5.2011E+02** |
| | STD | 4.9384E+01 | 4.1924E+01 | 3.5648E+01 | 1.8328E+01 | **7.1611E-02** |
| F6 | AVG | 5.2012E+02 | 5.2012E+02 | **5.2011E+02** | 5.2011E+02 | 5.2080E+02 |
| | STD | **5.0007E-02** | 7.4315E-02 | 6.3149E-02 | 8.0070E-02 | 6.1293E-02 |
| F7 | AVG | 6.1814E+02 | 6.1887E+02 | **6.1705E+02** | 6.1816E+02 | 6.1963E+02 |
| | STD | **3.0100E+00** | 3.0141E+00 | 3.0517E+00 | 3.1168E+00 | 3.7510E+00 |
| F8 | AVG | **6.2904E+02** | 7.0018E+02 | 7.0025E+02 | 7.0017E+02 | 7.0020E+02 |
| | STD | 1.8861E+00 | 5.3439E-02 | 6.6353E-02 | **4.5921E-02** | 6.5168E-02 |
| F9 | AVG | 7.0016E+02 | **7.0002E+02** | 8.5942E+02 | 8.5589E+02 | 8.6520E+02 |
| | STD | 4.4644E-02 | **2.4624E-02** | 1.7029E+01 | 1.3228E+01 | 1.6586E+01 |
| F10 | AVG | 8.5776E+02 | 8.5733E+02 | **8.5147E+02** | 9.9953E+02 | 1.0007E+03 |
| | STD | 1.3138E+01 | 1.6080E+01 | **6.5711E+00** | 2.8525E+01 | 3.1202E+01 |
| F11 | AVG | 9.9992E+02 | **9.9645E+02** | 1.0067E+03 | 1.0735E+03 | 2.3473E+03 |
| | STD | 3.2529E+01 | 2.9152E+01 | 2.4468E+01 | **1.1506E+01** | 4.8739E+02 |
| F12 | AVG | 2.2896E+03 | 2.2189E+03 | 2.3268E+03 | 2.3417E+03 | **1.9476E+03** |
| | STD | 4.7646E+02 | 4.1756E+02 | 4.2746E+02 | 3.8951E+02 | **2.0814E+02** |
| F13 | AVG | 4.6173E+03 | **4.1347E+03** | 4.4599E+03 | 4.3186E+03 | 4.5138E+03 |
| | STD | 5.9624E+02 | **5.5896E+02** | 5.9060E+02 | 6.5907E+02 | 5.6286E+02 |
| F14 | AVG | 7.2755E+03 | 1.2005E+03 | 1.2004E+03 | **1.2004E+03** | 1.2005E+03 |
| | STD | 3.1133E+02 | 2.1286E-01 | **1.3729E-01** | 1.5533E-01 | 1.7803E-01 |
| F15 | AVG | **1.2004E+03** | 1.2017E+03 | 1.3005E+03 | 1.3005E+03 | 1.3004E+03 |
| | STD | 1.8819E-01 | 1.5662E-01 | 1.2073E-01 | 1.1687E-01 | **1.0107E-01** |
| F16 | AVG | 1.3005E+03 | **1.3005E+03** | 1.3005E+03 | 1.4005E+03 | 1.4004E+03 |
| | STD | 1.2300E-01 | 1.1515E-01 | **5.9578E-02** | 1.9733E-01 | 1.3165E-01 |
| F17 | AVG | 1.4004E+03 | 1.4004E+03 | 1.4005E+03 | **1.4004E+03** | 1.5124E+03 |
| | STD | 2.0094E-01 | 2.1965E-01 | 2.5315E-01 | **5.1639E-02** | 2.9713E+00 |
| F18 | AVG | 1.5147E+03 | **1.5129E+03** | 1.5135E+03 | 1.5138E+03 | 1.5171E+03 |
| | STD | 5.1488E+00 | 4.5082E+00 | 5.1424E+00 | 5.0680E+00 | **1.0558E+00** |
| F19 | AVG | 1.6121E+03 | 1.6120E+03 | **1.6117E+03** | 1.6119E+03 | 1.6119E+03 |
| | STD | 4.0181E-01 | **3.6179E-01** | 6.4710E-01 | 4.9953E-01 | 5.9776E-01 |
| F20 | AVG | **1.6126E+03** | 1.3591E+06 | 1.1503E+06 | 1.0784E+06 | 1.0203E+06 |
| | STD | **2.3445E-01** | 7.4840E+05 | 5.5064E+05 | 6.0511E+05 | 5.3173E+05 |
| F21 | AVG | 1.3349E+06 | 6.0764E+06 | 1.8485E+04 | 2.5358E+04 | **6.6623E+03** |
| | STD | 6.1959E+05 | 2.5453E+06 | 5.3372E+04 | 5.0251E+04 | **4.8333E+03** |
| F22 | AVG | 2.9400E+04 | 5.5454E+03 | 2.5776E+05 | **1.9220E+03** | 1.9365E+03 |
| | STD | 6.1783E+04 | 3.3065E+03 | 2.0412E+05 | **2.3815E+01** | 3.3992E+01 |
| F23 | AVG | 1.9129E+03 | 1.9179E+03 | 1.9222E+03 | **1.9122E+03** | 8.2066E+03 |
| | STD | 2.5705E+00 | 2.1981E+01 | 2.2562E+01 | **2.3222E+00** | 5.1645E+03 |
| F24 | AVG | 7.6620E+03 | **6.4512E+03** | 7.8151E+03 | 8.0548E+03 | 1.2328E+04 |
| | STD | 4.5952E+03 | **4.2135E+03** | 4.4068E+03 | 7.7943E+03 | 6.2195E+03 |
| F25 | AVG | 4.0710E+05 | 5.8009E+05 | 5.0839E+05 | 3.1048E+05 | **2.9042E+05** |
| | STD | 2.5787E+05 | 4.0823E+05 | 3.5050E+05 | **2.1456E+05** | 3.0108E+05 |
| F26 | AVG | 1.1025E+06 | 2.6830E+03 | 2.7222E+03 | **2.6830E+03** | 2.7150E+03 |
| | STD | 5.1931E+05 | **1.4653E+02** | 1.5612E+02 | 2.1974E+02 | 2.5500E+02 |
| F27 | AVG | 2.7575E+03 | **2.6113E+03** | 2.6181E+03 | 2.6181E+03 | 2.6175E+03 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | STD | 2.2386E+02 | 1.2467E+02 | **1.8639E+00** | 3.1796E+00 | 1.8784E+00 |
| F28 | AVG | 2.6176E+03 | 2.6180E+03 | **2.6152E+03** | 2.6446E+03 | 2.6444E+03 |
| | STD | 1.8444E+00 | 1.5049E+00 | **1.4847E-03** | 4.9979E+00 | 5.3300E+00 |
| F29 | AVG | 2.6448E+03 | 2.6450E+03 | 2.6475E+03 | **2.6268E+03** | 2.7169E+03 |
| | STD | 5.0204E+00 | 4.7893E+00 | 2.9708E+00 | **1.1404E+00** | 6.0273E+00 |
| F30 | AVG | 2.7166E+03 | **2.7151E+03** | 2.7158E+03 | 2.7169E+03 | 2.7212E+03 |
| | STD | 4.7353E+00 | 5.8717E+00 | 4.9039E+00 | 5.6972E+00 | **3.0894E+00** |

## 4.3.2 Analysis of population size and number of evaluations

In metaheuristic algorithms, population size and number of evaluations, etc., as crucial initial setting values, also affect the optimization problem's optimization-seeking accuracy and optimization-seeking efficiency. Therefore, in this paper, we experiment and analyze the parameter sensitivity of the RIME algorithm by varying the population size and the number of evaluations utilizing gradient ascent. In this section, F13 of the CEC2017 test set above is used as the test example. The population sizes are set to 5, 10, 30, 50, 100, and 200, and the number of evaluations is 5000, 10000, 15000, 20000, 25000, and 30000.

The test results are shown in Figure 16. It can be seen that the increase in population size will directly improve the convergence speed of RIME and increase the search efficiency of the algorithm. This phenomenon is most apparent when the number of evaluations is less than 10,000, and the effect is not apparent when the number of evaluations exceeds 10,000. In addition, it can be seen from Figure 16 that the increase in the number of evaluations can help the algorithm find higher-quality solutions, regardless of the value of the population size. Again, this effect becomes minor after a certain level of evaluation times.

In summary, the population size and the number of iterations can significantly impact the RIME algorithm. The main impact is on the algorithm's finding accuracy and finding efficiency, which can be adjusted according to the actual problem in the application problem.

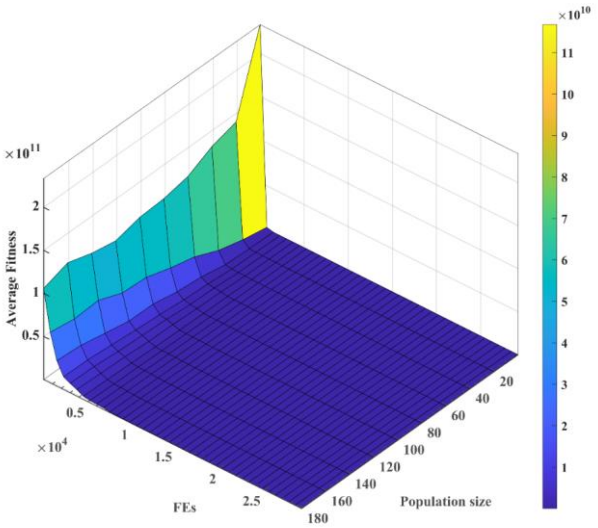



**Figure 16.** Effect of population size and number of evaluations on RIME

### 4.3.3 Analysis of dimension

The size of the dimension represents the different scales of the problem and is directly related to the difficulty of seeking the optimal. Usually, the larger the dimension of the problem, the larger the computation required by the algorithm, and the unreasonable initial value setting will impact the algorithm's accuracy and the ability to jump out of the local optimum. Therefore, in this paper, RIME dimensionality experiments are designed to demonstrate the effect of dimensionality on RIME. The dimensions are set to 10, 30, 50, and 100, respectively. The experimental results are shown in Table 13, where RIME10 means the RIME dimension size is 10, RIME30 means the dimension size is 30, RIME50 means the dimension size is 50, and RIME100 means the dimension size is 100. the population size is fixed at 30, and the evaluation count is 300,000. It can be seen from Table 13 that the dimension size affects the optimization performance of RIME. On the one hand, this indicates that the advantage of the RIME algorithm lies in solving optimization problems of lower dimensionality, which can find high precision solutions and higher efficiency of search in low latitude problems. On the other hand, as the problem dimension increases, the RIME algorithm requires a more significant computational effort, and the application should pay attention to this characteristic of the algorithm for appropriate parameter settings.

**Table 13.** Experimental results of RIME in different dimensions

| Fun | Item | RIME10 | RIME30 | RIME50 | RIME100 |
|---|---|---|---|---|---|
| F1 | AVG | **4.3889E+03** | 5.6126E+03 | 7.2589E+03 | 2.1006E+04 |
| | STD | **3.3859E+03** | 6.3421E+03 | 7.2484E+03 | 1.5888E+04 |
| F2 | AVG | **2.0000E+02** | 2.0588E+02 | 3.4422E+09 | 7.1291E+52 |
| | STD | **7.2473E-05** | 1.2964E+01 | 1.5522E+10 | 3.8414E+53 |
| F3 | AVG | **3.0000E+02** | 3.0006E+02 | 4.0194E+02 | 1.0441E+05 |
| | STD | **6.7059E-08** | 3.2031E-02 | 3.7468E+01 | 1.9908E+04 |
| F4 | AVG | **4.0050E+02** | 4.9471E+02 | 5.2085E+02 | 6.8857E+02 |
| | STD | **3.6292E-01** | 2.1713E+01 | 5.5933E+01 | 3.5269E+01 |
| F5 | AVG | **5.0799E+02** | 5.7243E+02 | 6.6986E+02 | 9.7173E+02 |
| | STD | **2.8561E+00** | 2.0630E+01 | 3.9201E+01 | 9.4815E+01 |
| F6 | AVG | **6.0000E+02** | 6.0047E+02 | 6.0528E+02 | 6.2510E+02 |
| | STD | **3.3529E-04** | 4.4106E-01 | 3.4586E+00 | 5.7129E+00 |
| F7 | AVG | **7.1644E+02** | 8.0626E+02 | 8.9718E+02 | 1.2973E+03 |
| | STD | **3.1242E+00** | 2.0803E+01 | 2.7021E+01 | 9.5586E+01 |
| F8 | AVG | **8.0829E+02** | 8.6937E+02 | 9.7037E+02 | 1.2589E+03 |
| | STD | **3.1715E+00** | 2.2368E+01 | 4.5329E+01 | 8.0746E+01 |
| F9 | AVG | **9.0000E+02** | 1.0811E+03 | 2.7054E+03 | 1.3052E+04 |
| | STD | **5.9359E-08** | 3.3265E+02 | 2.1849E+03 | 5.1119E+03 |
| F10 | AVG | **1.3117E+03** | 4.0348E+03 | 6.8410E+03 | 1.5287E+04 |
| | STD | **1.4722E+02** | 5.9114E+02 | 8.6270E+02 | 1.5117E+03 |
| F11 | AVG | **1.1053E+03** | 1.2404E+03 | 1.4325E+03 | 3.1316E+03 |
| | STD | **3.6011E+00** | 5.9501E+01 | 7.9977E+01 | 3.6888E+02 |
| F12 | AVG | **1.7283E+04** | 2.0817E+06 | 1.9033E+07 | 1.4559E+08 |
| | STD | **1.5067E+04** | 1.4244E+06 | 1.0092E+07 | 4.7147E+07 |
| F13 | AVG | **6.8981E+03** | 2.9927E+04 | 2.4923E+04 | 9.5310E+04 |
| | STD | **6.9853E+03** | 2.3096E+04 | 1.5549E+04 | 4.6217E+04 |

| | | | | | |
|---|---|---|---|---|---|
| F14 | AVG | **1.4113E+03** | 1.0366E+04 | 1.1188E+05 | 1.0045E+06 |
| | STD | **9.2057E+00** | 5.2389E+03 | 6.7473E+04 | 4.6865E+05 |
| F15 | AVG | **1.5031E+03** | 1.5187E+04 | 1.2545E+04 | 4.2239E+04 |
| | STD | **2.1856E+00** | 1.3866E+04 | 8.5431E+03 | 1.7314E+04 |
| F16 | AVG | **1.7079E+03** | 2.3438E+03 | 3.1111E+03 | 6.0594E+03 |
| | STD | **1.1863E+02** | 2.1512E+02 | 4.0906E+02 | 6.3284E+02 |
| F17 | AVG | **1.7381E+03** | 2.0134E+03 | 3.0304E+03 | 5.3656E+03 |
| | STD | **3.8669E+01** | 1.5394E+02 | 3.6546E+02 | 5.3649E+02 |
| F18 | AVG | **2.4966E+03** | 2.0528E+05 | 6.7584E+05 | 1.6901E+06 |
| | STD | **1.6424E+03** | 1.6358E+05 | 3.8042E+05 | 8.2535E+05 |
| F19 | AVG | **1.9016E+03** | 1.9008E+04 | 1.7540E+04 | 5.3393E+05 |
| | STD | **1.1207E+00** | 1.8116E+04 | 1.2772E+04 | 4.2437E+05 |
| F20 | AVG | **2.0158E+03** | 2.2571E+03 | 2.9803E+03 | 5.2621E+03 |
| | STD | **2.5257E+01** | 1.4836E+02 | 3.0265E+02 | 4.7647E+02 |
| F21 | AVG | **2.2987E+03** | 2.3784E+03 | 2.4667E+03 | 2.8215E+03 |
| | STD | 3.9348E+01 | **2.1657E+01** | 3.8805E+01 | 9.4717E+01 |
| F22 | AVG | **2.3151E+03** | 4.8311E+03 | 8.1830E+03 | 1.7192E+04 |
| | STD | **1.0327E+02** | 1.2420E+03 | 8.8264E+02 | 1.4613E+03 |
| F23 | AVG | **2.6031E+03** | 2.7337E+03 | 2.9129E+03 | 3.2768E+03 |
| | STD | 5.7412E+01 | **2.0839E+01** | 4.2746E+01 | 7.3189E+01 |
| F24 | AVG | **2.7204E+03** | 2.9116E+03 | 3.0770E+03 | 3.7497E+03 |
| | STD | 7.4921E+01 | **2.3570E+01** | 4.4140E+01 | 7.6257E+01 |
| F25 | AVG | 2.9283E+03 | **2.8962E+03** | 3.0518E+03 | 3.3613E+03 |
| | STD | 2.2859E+01 | **1.6190E+01** | 3.7850E+01 | 7.2865E+01 |
| F26 | AVG | **2.9620E+03** | 4.4781E+03 | 5.6933E+03 | 1.1034E+04 |
| | STD | 2.5909E+02 | **2.2332E+02** | 4.4382E+02 | 1.0357E+03 |
| F27 | AVG | **3.0952E+03** | 3.2213E+03 | 3.4330E+03 | 3.6913E+03 |
| | STD | 1.4660E+01 | **9.3095E+00** | 8.3903E+01 | 1.0106E+02 |
| F28 | AVG | 3.2720E+03 | **3.2106E+03** | 3.3085E+03 | 3.4545E+03 |
| | STD | 1.5331E+02 | 3.3163E+01 | **2.7570E+01** | 3.7577E+01 |
| F29 | AVG | **3.1699E+03** | 3.7025E+03 | 4.3598E+03 | 7.7233E+03 |
| | STD | **3.5597E+01** | 1.7956E+02 | 2.6887E+02 | 5.3929E+02 |
| F30 | AVG | 1.4553E+05 | **1.9904E+04** | 3.4576E+06 | 1.2374E+07 |
| | STD | 3.0712E+05 | **1.2636E+04** | 1.1742E+06 | 3.5045E+06 |

To further investigate the effect of dimensionality on RIME and to demonstrate that RIME still maintains the advantage of optimization performance among peer algorithms under the same dimensionality. This section also designs a comparison experiment of peer algorithms in different dimensions, and the experimental results are also statistically verified using the Wilcoxon signed-rank test and the Friedman test. The Wilcoxon signed rank and Friedman rank are shown in Table 14 and Figure 17, respectively. Table 14 shows that RIME ranks first in the dimensions of 10, 30, 50, and 100 for the optimization problem, indicating that the RIME algorithm has strong and stable optimization performance in different dimensions. In addition, it can be seen that as the dimensionality increases from 10 to 100, the average ranking of RIME becomes more advanced and reaches the highest average ranking in the experiment at dimension 100, which indicates that the RIME algorithm has a significant advantage over other algorithms in dealing with high-dimensional problems. The Friedman rank in Figure 17 also supports the advantages and stability of RIME. In summary, RIME still has powerful performance in different dimensions and can be used to solve optimization problems in various dimensions.

**Table 14.** Wilcoxon signed-rank of RIME and peer algorithm in different dimensions

| Dim=10 | | | | Dim=30 | | | |
|---|---|---|---|---|---|---|---|
| | +/−/= | Mean | Rank | | +/−/= | Mean | Rank |
| RIME | ~ | **1.83** | **1** | RIME | ~ | **1.63** | **1** |
| PSO | 24/0/6 | 4.53 | 3 | PSO | 24/2/4 | 4.80 | 3 |
| WOA | 30/0/0 | 7.67 | 9 | WOA | 30/0/0 | 7.47 | 9 |
| SCA | 30/0/0 | 8.47 | 11 | SCA | 30/0/0 | 8.27 | 11 |
| HHO | 29/1/0 | 5.90 | 4 | HHO | 29/0/1 | 5.50 | 4 |
| MFO | 30/0/0 | 6.23 | 6 | MFO | 30/0/0 | 6.60 | 8 |
| JAYA | 29/1/0 | 6.40 | 7 | JAYA | 29/1/0 | 6.23 | 5 |
| FA | 29/1/0 | 8.13 | 10 | FA | 29/0/1 | 8.13 | 10 |
| RFO | 23/4/3 | 6.20 | 5 | RFO | 25/3/2 | 6.37 | 6 |
| GWO | 23/0/7 | 4.23 | 2 | GWO | 22/0/8 | 4.53 | 2 |
| BA | 23/4/3 | 6.40 | 7 | BA | 23/2/5 | 6.47 | 7 |
| Dim=50 | | | | Dim=100 | | | |
| | +/−/= | Mean | Rank | | +/−/= | Mean | Rank |
| RIME | ~ | **1.63** | **1** | RIME | **~** | **1.40** | **1** |
| PSO | 23/1/6 | 4.73 | 3 | PSO | 24/1/5 | 5.00 | 3 |
| WOA | 30/0/0 | 7.67 | 9 | WOA | 30/0/0 | 7.03 | 8 |
| SCA | 30/0/0 | 8.27 | 11 | SCA | 30/0/0 | 8.17 | 10 |
| HHO | 29/0/1 | 5.17 | 4 | HHO | 30/0/0 | 5.20 | 4 |
| MFO | 30/0/0 | 6.63 | 7 | MFO | 30/0/0 | 7.07 | 9 |
| JAYA | 29/1/0 | 6.07 | 5 | JAYA | 29/1/0 | 6.00 | 5 |
| FA | 29/1/0 | 8.17 | 10 | FA | 29/1/0 | 8.30 | 11 |
| RFO | 25/4/1 | 6.53 | 6 | RFO | 25/2/3 | 6.53 | 6 |
| GWO | 23/1/6 | 4.47 | 2 | GWO | 24/1/5 | 4.50 | 2 |
| BA | 24/3/3 | 6.67 | 8 | BA | 24/3/3 | 6.80 | 7 |

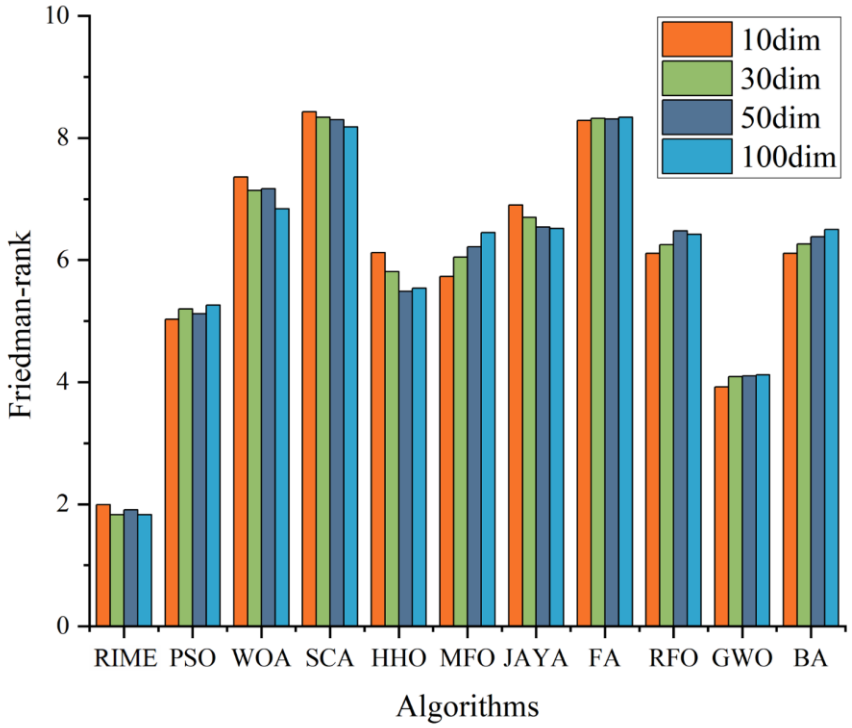



**Figure 17.** Friedman-rank of RIME and peer algorithm in different dimensions

## 4.4 Experiments on engineering design problems

Real-world engineering problems, such as bin packing problems [67], and neural networks [68, 69], deep learning [70], often have multiple feasible solutions, and they include constrained variables that decision-makers need to decide the best possible (approximated) solutions within the required accuracy [71-73]. This section demonstrates the advantages of the RIME algorithm for practical problems by PVD problem, WBD problem, SRD problem, IBD problem, and MDCBD problem. The population size is 50, and the number of iterations of the method is always set to 2000.

### 4.4.1 PVD problem

In order to achieve the goal of minimizing the manufacturing cost of this engineering optimization model, the RIME method is used for the PVD problem in this engineering problem, as illustrated in Figure 18. The PVD problem's inner radius ($R$), head thickness ($T_h$), shell thickness ($T_s$), and section range less head all have unknown values ($l$). Constrictions that must be met in order to implement the pressurizer model must also be met concurrently with the PV model's optimization. The PVD model's linear programming equation is as follows.

$$\text{Assume: } \vec{x} = [x_1, x_2, x_3, x_4] = [T_s, T_h, R, L]$$

$$min \quad f(\vec{x}) = 0.6224 * x_1 * x_3 * x_4 + 1.7781 * x_3 * x_1^2 + 3.1661 * x_4 * x_1^2 + 19.84 * x_3 * x_1^2$$

$$s.t. \begin{cases} g_1(\vec{x}) = -x_1 + 0.0193 * x_3 \leq 0 \\ g_2(\vec{x}) = -x_3 + 0.00954 * x_3 \leq 0 \\ g_3(\vec{x}) = -\pi * x_4 * x_3^2 - \frac{4}{3} * \pi * x_3^3 + 1296000 \leq 0 \\ g_4(\vec{x}) = x_4 - 240 \leq 0 \\ 0 \leq x_1 \leq 99 \\ 0 \leq x_2 \leq 99 \\ 10 \leq x_3 \leq 200 \\ 10 \leq x_4 \leq 200 \end{cases}$$

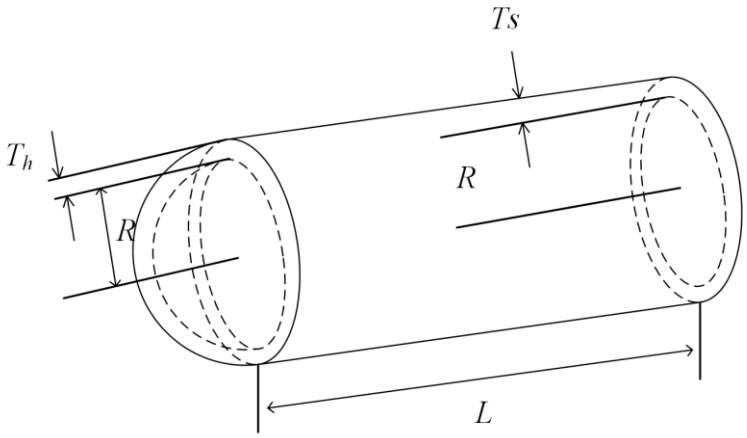



**Figure 18.** PVD problem

Table 15 shows the RIME algorithm's best result for the PVD model, with the pressurizer being the least. In this experiment, the RIME algorithm is compared against ten different algorithms in an identical experimental setting. Last but not least, the RIME algorithm outperforms the other algorithms with the best outcomes and can better resolve the PVD practice.

**Table 15.** Comparison results of the PVD problem

| Algorithm | Optimal values for variables | | | | Optimum cost |
|---|---|---|---|---|---|
| | $T_s$ | $T_h$ | $R$ | $L$ | |
| **RIME** | **0.8750** | **0.4375** | **45.9482** | **135.3594** | **6055.5868** |
| MFO[53] | 0.8125 | 0.4375 | 42.0984 | 176.6366 | 6059.7143 |
| BA[74] | 0.8125 | 0.4375 | 42.0984 | 176.6366 | 6059.7143 |
| HPSO[75] | 0.8125 | 0.4375 | 42.0984 | 176.6366 | 6059.7143 |
| CSS[76] | 0.8125 | 0.4375 | 42.1036 | 176.5727 | 6059.0888 |
| CPSO[77] | 0.8125 | 0.4375 | 42.0912 | 176.7465 | 6061.0777 |
| ACO[78] | 0.8125 | 0.4375 | 42.1036 | 176.5727 | 6059.0888 |
| WOA[32] | 0.8125 | 0.4375 | 42.0983 | 176.6390 | 6059.7410 |
| MDDE[79] | 0.8125 | 0.4375 | 42.0984 | 176.6360 | 6059.7017 |
| Branch-bound[80] | 1.1250 | 0.6250 | 47.7000 | 117.7010 | 8129.1036 |
| NDE[81] | 0.8125 | 0.4375 | 42.0984 | 176.6365 | 6059.7143 |

### 4.4.2 WBD problem

The objective of this case is to determine the welded beam with the lowest cost given four limitations and the key characteristics of shear stress ($\tau$), bending stress ($\theta$), buckling load ($P_c$) and deflection ($\delta$). As indicated in Figure 19, this task includes the four variables: welding seam thickness ($h$); welding joint length ($l$); beam width ($t$); beam thickness ($b$). The following is the problem's mathematical model.

Consider $\vec{x} = [x_1, x_2, x_3, x_4] = [\, h\ l\ t\ b]$

Minimize $f(\vec{x}) = 1.10471x_1^2 + 0.04811x_3x_4(14.0+x_4)$

Subject to $g_1(\vec{x}) = \tau(\vec{x}) - \tau_{max} \leq 0$

$$g_2(\vec{x}) = \sigma(\vec{x}) - \sigma_{max} \leq 0$$

$$g_3(\vec{x}) = \delta(\vec{x}) - \delta_{max} \leq 0$$

$$g_4(\vec{x}) = x_1 - x_4 \leq 0$$

$$g_5(\vec{x}) = P - P_C(\vec{x}) \leq 0$$

$$g_6(\vec{x}) = 0.125 - x_1 \leq 0$$

$$g_7(\vec{x}) = 1.10471x_1^2 + 0.04811x_3x_4(14.0+x_2) - 5.0 \leq 0$$

Variable range $0.1 \leq x_1 \leq 2$, $0.1 \leq x_2 \leq 10$, $0.1 \leq x_3 \leq 10$, $0.1 \leq x_4 \leq 2$

where $\tau(\vec{x}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}$, $\tau' = \frac{P}{\sqrt{2}x_1x_2}$ $\tau'' = \frac{MR}{J}$ $M = P\left(L + \frac{x_2}{2}\right)$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1+x_3}{2}\right)^2}$$

$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{4} + \left(\frac{x_1+x_3}{2}\right)^2\right]\right\}$$

$$\sigma(\vec{x}) = \frac{6PL}{x_4x_3{}^2}, \delta(\vec{x}) = \frac{6PL^3}{Ex_3^2x_4}$$

$$P_C(\vec{x}) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)$$

$$P = 60001b, L = 14 \in .. \delta_{max} = 0.25 \in ..$$

$$E = 30 \times 1^6 psi, G = 12 \times 10^6 psi$$

$$\tau_{max} = 13600psi, \sigma_{max} = 30000psi$$

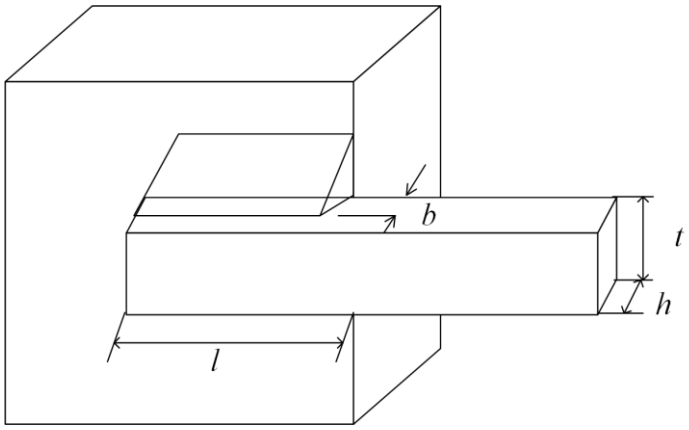



**Figure 19.** WBD problem

Table 16 displays the simulation outcomes for the WBD challenge. The table findings indicate that 1.722821 is the ideal cost for RIME. Of all the algorithms, RIME has the lowest optimized value. This demonstrates that RIME delivers positive outcomes for this technical challenge.

**Table 16.** Comparison results of WBD problem between RIME and other approaches

| Algorithm | Optimal values for variables | | | | Optimum cost |
|---|---|---|---|---|---|
| | *h* | *l* | *t* | *b* | |
| **RIME** | **0.208000** | **3.250000** | **9.053702** | **0.208620** | **1.722821** |
| RO[82] | 0.203687 | 3.528467 | 9.004233 | 0.207241 | 1.735344 |
| SSA[83] | 0.205700 | 3.471400 | 9.036600 | 0.205700 | 1.724910 |
| CDE[84] | 0.203137 | 3.542998 | 9.033498 | 0.206179 | 1.733462 |
| GWO[30] | 0.205700 | 3.478400 | 9.036800 | 0.205800 | 1.726240 |
| GSA[46] | 0.182129 | 3.856979 | 10.00000 | 0.202376 | 1.879950 |
| NDE[81] | 0.205729 | 3.470488 | 9.903662 | 0.205729 | 1.724852 |

## 4.4.3 SRD problem

The reduction of the gearbox's weight is the aim of this challenge. As indicated in Figure 20, the reducer's weight must be kept as low as possible while still respecting the gear teeth's bending stress and the shaft's surface stress. The variable $x_1$ to $x_7$ stand for the face width $(b)$, the module of teeth $(m)$, the number of teeth in the pinion $(z)$, the length of the first shaft between bearings $(l_1)$, the length of the second shaft between bearings $(l_2)$, and the diameter of the first $(d_1)$ and the second shaft $(d_2)$, respectively. The following is the problem's mathematical model.

Consider $\vec{z} = [z_1\ z_2\ z_3\ z_4\ z_5\ z_6\ z_7\ ] = [b\ m\ p\ l_1\ l_2\ d_1\ d_2]$,

Minimize $f(\vec{z}) = 0.7854 z_1 z_2^2 (3.3333 z_3^2 + 14.9334 z_3 - 43.0934) - 1.508 z_1 (z_6^2 + z_7^2) + 7.4777 (z_6^2 + z_7^2) + 0.7854 (z_4 z_6^2 + z_5 z_7^2)$

Subject to:

$g_1(\vec{z}) = \frac{27}{z_1 z_3 z_2^2} - 1 \leq 0,$

$$g_2(\vec{z}) = \frac{397.5}{z_1 z_2^2 z_3^2} - 1 \leq 0,$$

$$g_3(\vec{z}) = \frac{1.93 z_4^3}{z_2 z_6^4 z_3} - 1 \leq 0,$$

$$g_4(\vec{z}) = \frac{1.93 z_5^3}{z_2 z_7^4 z_3} - 1 \leq 0,$$

$$g_5(\vec{z}) = \frac{\left[\left(745(z_4/z_2 z_3)\right)^2 + 16.9\times10^6\right]^{1/2}}{110 z_6^3} - 1 \leq 0,$$

$$g_6(\vec{z}) = \frac{\left[\left(745(z_5/z_2 z_3)\right)^2 + 157.5\times10^6\right]^{1/2}}{85 z_7^3} - 1 \leq 0,$$

$$g_7(\vec{z}) = \frac{z_2 z_3}{40} - 1 \leq 0,$$

$$g_8(\vec{z}) = \frac{5 z_2}{z_1} - 1 \leq 0,$$

$$g_9(\vec{z}) = \frac{z_1}{12 z_2} - 1 \leq 0,$$

$$g_{10}(\vec{z}) = \frac{1.5 z_6 + 1.9}{z_4} - 1 \leq 0,$$

$$g_{11}(\vec{z}) = \frac{1.1 z_7 + 1.9}{z_5} - 1 \leq 0,$$

where $2.6 \leq z_1 \leq 3.6, 0.7 \leq z_2 \leq 0.8, 17 \leq z_3 \leq 28, 7.3 \leq z_4 \leq 28, 7.3 \leq z_5 \leq 8.3, 2.9 \leq z_6 \leq 3.9, 5.0 \leq z_7 \leq 5.5$

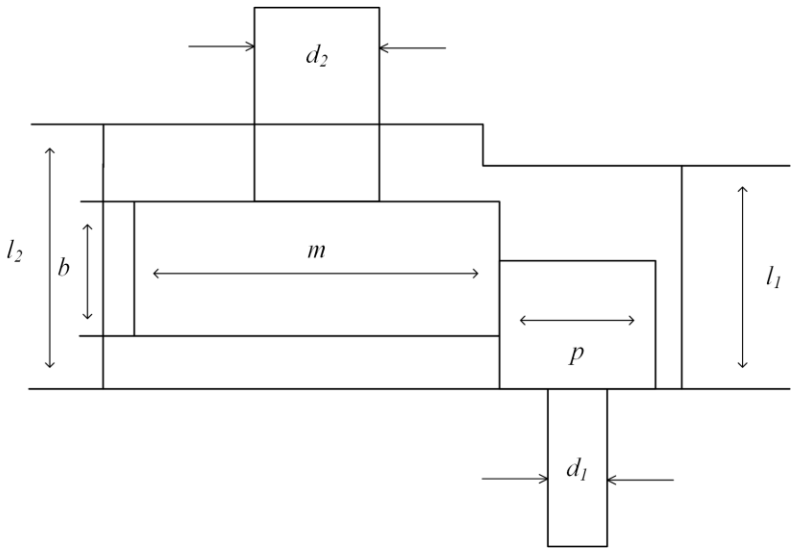


**Figure 20.** SRD problem

The comparative outcomes of the SRD problem are displayed in Table 17. We can see from the table that RIME is rated top and has the lowest optimization cost. RIME significantly improves this engineering challenge as compared to GWO, PSO, GSA, and SCA.

**Table 17.** Comparison results of SRD problem between RIME and other approaches

| Algorithm | Optimal values for variables | | | | | | | Optimum cost |
|---|---|---|---|---|---|---|---|---|
| | $z_1$ | $z_2$ | $z_3$ | $z_4$ | $z_5$ | $z_6$ | $z_7$ | |
| **RIME** | **3.50357** | **0.7** | **17** | **7.3** | **7.8** | **3.350230** | **5.287497** | **2997.6982** |
| SCA[85] | 3.50875 | 0.7 | 17 | 7.3 | 7.8 | 3.461020 | 5.289213 | 3030.5630 |
| GSA[46] | 3.60000 | 0.7 | 17 | 8.3 | 7.8 | 3.369658 | 5.289224 | 3051.1200 |
| GWO[30] | 3.50669 | 0.7 | 17 | 7.380933 | 7.815726 | 3.357847 | 5.286768 | 3001.2880 |
| PSO[86] | 3.50001 | 0.7 | 17 | 8.3 | 7.8 | 3.352412 | 5.286715 | 3005.7630 |

## 4.4.4 IBD problem

As seen in Figure 21, the structural design of I-beams is the subject of the second optimization practice. This practice aims to get the least amount of vertical deflection possible during the design phase. The question's structural components are length, two thicknesses, and one height. Following are the formulae and restrictions for this practice:

Consider:

$$\vec{x} = [x_1\ x_2\ x_3\ x_4] = [b\ h\ t_w\ t_f]$$

The values of the above four variables are:

$$10 \le x_1 \le 50$$

$$10 \le x_2 \le 80$$

$$0.9 \le x_3 \le 5$$

$$0.9 \le x_4 \le 5$$

Minimize:

$$f(\vec{x}) = \frac{5000}{\frac{t_w(h-2t_f)^3}{12} + \frac{bt_f^3}{6} + 2bt_f\left(\frac{h-t_f}{2}\right)^2}$$

Subject to:

$$g(\vec{x}) = 2bt_w + t_w(h-2t_f) \le 0$$

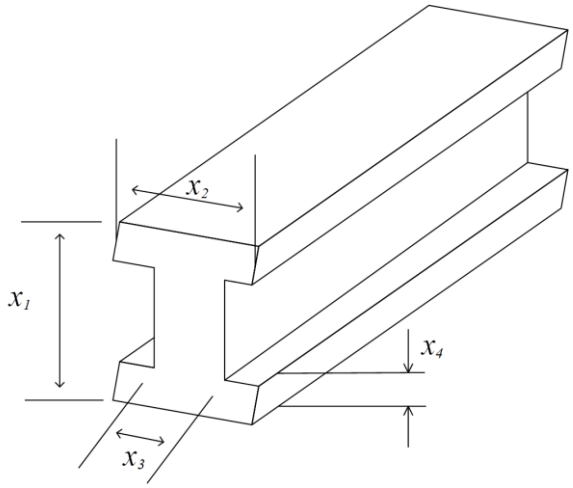



**Figure 21.** IBD problem

This problem compares RIME with ARSM, SOS, CS, and IARSM. Table 18 contains the comparative findings for the IBD problem. The restrictions of this engineering problem are straightforward. As can be observed, all algorithms yield comparable outcomes. This suggests that RIME and other effective algorithms have the same impact on this practice.

**Table 18.** Comparison results of IBD problem between RIME and other approaches

| Algorithm | Optimum variables | | | | Optimum cost |
|---|---|---|---|---|---|
| | $b$ | $h$ | $t_w$ | $t_f$ | |
| **RIME** | **50.0000** | **80.0000** | **0.9000** | **2.32167** | **0.01308** |
| ARSM[87] | 37.0500 | 80.0000 | 1.7100 | 2.3100 | 0.0157 |
| SOS[88] | 50.0000 | 80.0000 | 0.9000 | 2.3218 | 0.0131 |
| CS[89] | 50.0000 | 80.0000 | 0.9000 | 2.3217 | 0.0131 |
| IARSM[87] | 48.4200 | 79.9900 | 0.9000 | 2.4000 | 0.1310 |

## 4.4.5 MDCBD problem

As illustrated in Figure 22, this design challenge calls for the creation of a multi-disc clutch brake while accounting for two factors: the minimal mass of the braking system ($f_1$) and the minimum stop time ($T$). This practice involves five variables: $\vec{x} = (r_i, r_0, t, F, Z)$, where $r_i$ is the internal radius, $r_0$ is the external radius, $t$ is the thickness of the disc, $F$ is the driving force, and $Z$ represents the number of friction surfaces. The five variables mentioned above have the following exact value ranges and are discrete variables.

$$r_i = (60\sim80)\ mm$$

$$r_0 = (90\sim110)\ mm$$

$$t = (1, 1.5, 2, 2.5, 3)$$

$$F = (600, 610, 620, \ldots, 980, 990, 1000)$$

$$Z = (2\sim20)$$

The formula for this design problem is as follows:

Minimize:

$$f_1(\vec{x}) = \pi(x_2^2 - x_1^2)x_3(X_5 + 1)\rho$$

$$f_2(\vec{x}) = T = \frac{I_z\omega}{M_h + M_f}$$

Subject to:

$$g_1(\vec{x}) = x_2 - x_1 - \Delta R > 0$$

$$g_2(\vec{x}) = L_{max} - (x_5 + 1)(x_3 + \delta) \geq 0$$

$$g_3(\vec{x}) = p_{max} - p_{rz} \geq 0$$

$$g_4(\vec{x}) = p_{max}V_{sr,max} - p_{rz}V_{sr} \geq 0$$

$$g_5(\vec{x}) = V_{sr,max} - V_{sr} \geq 0$$

$$g_6(\vec{x}) = M_h - sM_s \geq 0$$

$$g_7(\vec{x}) = T \geq 0$$

$$g_8(\vec{x}) = T_{max} - T \geq 0$$

$$r_{i,min} \leq x_1 \leq r_{i,max}$$

$$r_{o,min} \leq x_2 \leq r_{o,max}$$

$$t_{min} \leq x_3 \leq t_{max}$$

$$0 \leq x_4 \leq F_{max}$$

$$2 \leq x_5 \leq Z_{max}$$

The above parameters are as follows:

$M_h = \frac{2}{3}\mu x_4 x_5 \frac{x_2^3 - x_1^3}{x_2^2 - x_1^2} N \cdot mm$ $\omega = \frac{\pi n}{30}\ rad/s$ $A = \pi(x_2^2 - x_1^2)\ mm^2$ $p_{rz} = \frac{x_a}{A}\ N/mm^2$

$V_{sh} = \frac{\pi R_{sr} n}{30} mm/s$ $R_{sr} = \frac{2}{3}\frac{x_2^3 - x_1^3}{x_2^2 - x_1^2}\ mm$ $\Delta R = 20\ mm$ $L_{max} = 30\ mm$

$\mu = 0.5$ $p_{max} = 1\ MPa$ $\rho = 0.0000078\ kg/mm^3$ $V_{srmax} = 10\ m/s$

| | | | |
|---|---|---|---|
| $s = 1.5$ | $T_{max} = 15\ s$ | $n = 250\ rpm$ | $M_s = 40\ Nm$ |
| $M_f = 3\ Nm$ | $I_z = 55\ kg \cdot m^2$ | $\delta = 0.5\ mm$ | $r_{i,min} = 60\ mm$ |
| $r_{i,max} = 80\ mm$ | $r_{0,min} = 90\ mm$ | $r_{0,min} = 110\ mm$ | $t_{min} = 1.5\ mm$ |
| $t_{max} = 3\ mm$ | $F_{max} = 1000\ N$ | $Z_{max} = 9$ | |

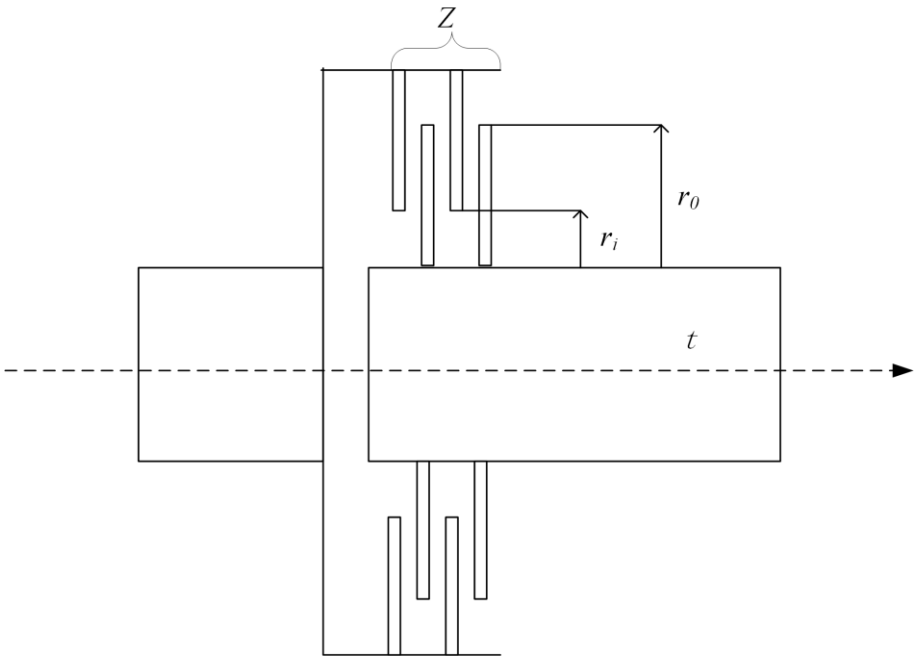



**Figure 22.** MDCBD problem

In this challenge, RIME is contrasted with a number of algorithms, including CBA, WCA, PVS, and TLBO. Table 19 demonstrates that RIME maintains a sizable lead in the MDCBD problem. The effective improvement in engineering material use and considerable reduction in engineering material consumption demonstrates the great performance and precise search of RIME.

**Table 19.** Comparison results of MDCBD problem

| Algorithm | Optimum variables | | | | | Optimum cost |
|---|---|---|---|---|---|---|
| | $r_i$ | $r_0$ | $t$ | $F$ | $Z$ | |
| **RIME** | **75.0000** | **95.0000** | **1.0000** | **1000.0000** | **2.0000** | **0.249945** |
| CBA[90] | 80.0000 | 90.0000 | 3.0000 | 1000.0000 | 2.0000 | 0.263684 |
| WCA[91] | 70.0000 | 90.0000 | 1.0000 | 910.0000 | 3.0000 | 0.313656 |
| PVS[92] | 70.0000 | 90.0000 | 1.0000 | 980.0000 | 3.0000 | 0.313660 |
| TLBO[93] | 70.0000 | 90.0000 | 1.0000 | 810.0000 | 3.0000 | 0.313656 |

# 5 Conclusions and future works

In this study, a new high-performance optimization algorithm based on the rime formation process is proposed to solve the complex optimization problems of today. The RIME algorithm proposes a soft-rime search strategy for the search of the algorithm, mainly by simulating the motion of soft-rime particles. By simulating the crossover behavior between hard-rime agents, the hard-rime puncture mechanism is also proposed for the exploitation step of the algorithm. Finally, the selection mechanism of the metaheuristic algorithm is improved, and the positive greedy selection mechanism is proposed to avoid the local optimum trap. In the experiments, this paper is first designed to qualitatively analyze the algorithm using agent historical position experiments, particle change experiments, fitness value change experiments, and iterative curve experiments to demonstrate the algorithm's characteristics in finding the optimal solution. Then, the RIME algorithm is compared with 10 classical algorithms and 10 high-performance algorithms to demonstrate the performance advantages of the RIME algorithm. The set of functions tested is very comprehensive, including unimodal, multimodal, hybrid, and composition functions. The results of the comparisons also illustrate the validity of the experiments by the Wilcoxon signed-rank test and Friedman test. The experimental results show that the RIME algorithm can balance exploration and exploitation better and has some advantages over peer algorithms. This paper further determines the parameters of RIME for different optimization problems by analyzing the parameter sensitivity of the RIME algorithm to ensure its performance is maximized. Finally, applying the RIME algorithm to five engineering optimization problems, including the PVD problem, WBD problem, SRD problem, IBD problem, and MDCBD problem, shows that the algorithm also has a strong potential for practical optimization problems. In summary, the excellent performance of the RIME algorithm can be theoretically attributed to the following points.

1) The soft-rime search strategy makes the RIME algorithm have a unique ladder exploration and exploitation method, which makes the algorithm constantly switch between large-scale exploration and small-scale exploitation and can simultaneously take into account the breadth and depth when seeking the optimal.

2) The hard-rime puncture mechanism enables the RIME algorithm to quickly lock the global approximate optimal solution and achieve centralized exploitation by the crossover between the optimal solution and the current solution, improving the solution's accuracy and efficiency.

3) The positive greedy selection mechanism enables the RIME algorithm to actively change the position of agents while avoiding poor quality solutions from appearing in the search population, ensuring the quality of the entire population after each iteration, improving population diversity, and significantly reducing the performance loss of the algorithm.

In future work, we will develop binary and multi-objective RIME versions for different optimization problems. In addition, we will also consider further improving the optimization performance of RIME itself. This algorithm will also solve more engineering problems or problems in other fields.

# Acknowledgement

This work was supported in part by the Natural Science Foundation of Zhejiang Province (LZ22F020005), National Natural Science Foundation of China (62076185, U1809209).

# References


1. Li, S. and Z. Geng, *Bicriteria scheduling on an unbounded parallel-batch machine for minimizing makespan and maximum cost.* Information Processing Letters, 2023. **180**: p. 106343.
2. Li, R., et al., *Hybrid Memetic Pretrained Factor Analysis-Based Deep Belief Networks for Transient Electromagnetic Inversion.* IEEE Transactions on Geoscience and Remote Sensing, 2022. **60**: p. 1-20.
3. Wang, H., et al., *A Structural Evolution-Based Anomaly Detection Method for Generalized Evolving Social Networks.* The Computer Journal, 2022. **65**(5): p. 1189-1199.
4. van den Berg, E. and M.P. Friedlander, *PROBING THE PARETO FRONTIER FOR BASIS PURSUIT SOLUTIONS.* Siam Journal on Scientific Computing, 2008. **31**(2): p. 890-912.
5. Rodi, W. and R.L. Mackie, *Nonlinear conjugate gradients algorithm for 2-D magnetotelluric inversion.* Geophysics, 2001. **66**(1): p. 174-187.
6. Hinton, G.E. and R.R. Salakhutdinov, *Reducing the dimensionality of data with neural networks.* Science, 2006. **313**(5786): p. 504-507.
7. Cao, B., et al., *Large-scale many-objective deployment optimization of edge servers.* IEEE Transactions on Intelligent Transportation Systems, 2021. **22**(6): p. 3841-3849.
8. Cao, B., et al., *Diversified personalized recommendation optimization based on mobile data.* IEEE Transactions on Intelligent Transportation Systems, 2020. **22**(4): p. 2133-2139.
9. Dong, R.Y., et al., *Boosted kernel search: Framework, analysis and case studies on the economic emission dispatch problem.* Knowledge-Based Systems, 2021. **233**.
10. Cao, B., et al., *Many-objective deployment optimization for a drone-assisted camera network.* IEEE transactions on network science and engineering, 2021. **8**(4): p. 2756-2764.
11. Zeng, N., et al., *A dynamic neighborhood-based switching particle swarm optimization algorithm.* IEEE Transactions on Cybernetics, 2020.
12. Zeng, N., et al., *A new switching-delayed-PSO-based optimized SVM algorithm for diagnosis of Alzheimer's disease.* Neurocomputing, 2018. **320**: p. 195-202.
13. Ahmadianfar, I., et al., *INFO: An Efficient Optimization Algorithm based on Weighted Mean of Vectors.* Expert Systems with Applications, 2022: p. 116516.
14. Ahmadianfar, I., et al., *RUN Beyond the Metaphor: An Efficient Optimization Algorithm Based on Runge Kutta Method.* Expert Systems with Applications, 2021: p. 115079.
15. Cao, B., et al., *A memetic algorithm based on two_Arch2 for multi-depot heterogeneous-vehicle capacitated Arc routing problem.* Swarm and Evolutionary Computation, 2021. **63**: p. 100864.
16. Liu, Y., et al., *Simulated annealing-based dynamic step shuffled frog leaping algorithm: Optimal performance design and feature selection.* Neurocomputing, 2022. **503**: p. 325-362.
17. Cao, B., et al., *RFID reader anticollision based on distributed parallel particle swarm optimization.* IEEE Internet of Things Journal, 2020. **8**(5): p. 3099-3107.
18. Xu, X., C. Wang, and P. Zhou, *GVRP considered oil-gas recovery in refined oil distribution: from an environmental perspective.* International Journal of Production Economics, 2021. **235**: p. 108078.
19. Li, H., et al., *A ranking-system-based switching particle swarm optimizer with dynamic*

*learning strategies.* Neurocomputing, 2022. **494**: p. 356-367.
20. Chen, H., et al., *Slime mould algorithm: a comprehensive review of recent variants and applications*. International Journal of Systems Science, 2022.
21. Sastry, K., D.E. Goldberg, and G. Kendall, *Genetic Algorithms*, in *Search Methodologies*. 2014. p. 93-117.
22. Storn, R. and K.J.J.o.G.O. Price, *Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces.* 1997. **11**(4): p. 341-359.
23. Robbiano, A., et al., *Evolutionary optimization strategies for Liquid-liquid interaction parameters.* Fluid Phase Equilibria, 2023. **564**.
24. Yao, X., Y. Liu, and G.M. Lin, *Evolutionary programming made faster.* Ieee Transactions on Evolutionary Computation, 1999. **3**(2): p. 82-102.
25. Ren, L., et al., *Shale gas load recovery modeling and analysis after hydraulic fracturing based on genetic expression programming: A case study of southern Sichuan Basin shale.* Journal of Natural Gas Science and Engineering, 2022. **107**.
26. Kennedy, J. and R. Eberhart. *Particle swarm optimization*. in *Proceedings of ICNN'95 - International Conference on Neural Networks*. 1995.
27. Heidari, A.A., et al., *Harris hawks optimization: Algorithm and applications.* Future Generation Computer Systems, 2019. **97**: p. 849-872.
28. Li, S., et al., *Slime mould algorithm: A new method for stochastic optimization.* Future Generation Computer Systems, 2020. **111**: p. 300-323.
29. Yang, Y., et al., *Hunger games search: Visions, conception, implementation, deep analysis, perspectives, and towards performance shifts.* Expert Systems with Applications, 2021. **177**: p. 114864.
30. Mirjalili, S., S.M. Mirjalili, and A. Lewis, *Grey Wolf Optimizer.* Advances in Engineering Software, 2014. **69**: p. 46-61.
31. Tu, J., et al., *The Colony Predation Algorithm.* Journal of Bionic Engineering, 2021. **18**(3): p. 674-710.
32. Mirjalili, S. and A. Lewis, *The Whale Optimization Algorithm.* Advances in Engineering Software, 2016. **95**: p. 51-67.
33. Dorigo, M. *Optimization, Learning and Natural Algorithms*. 1992.
34. Dorigo, M. and G.D. Caro, *The ant colony optimization meta-heuristic*, in *New ideas in optimization*. 1999, McGraw-Hill Ltd., UK. p. 11–32.
35. Karaboga, D., *An idea based on honey bee swarm for numerical optimization.* 2005.
36. Yang, X.-S., *A new metaheuristic bat-inspired algorithm*, in *Nature inspired cooperative strategies for optimization (NICSO 2010)*. 2010, Springer. p. 65-74.
37. Yang, X. and D. Suash. *Cuckoo Search via Lévy flights*. in *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*. 2009.
38. Rao, R.V., V.J. Savsani, and D.P. Vakharia, *Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems.* Computer-Aided Design, 2011. **43**(3): p. 303-315.
39. Geem, Z.W., J.H. Kim, and G.V. Loganathan, *A new heuristic optimization algorithm: harmony search.* simulation, 2001. **76**(2): p. 60-68.
40. He, S., Q. Wu, and J.R. Saunders. *A novel group search optimizer inspired by animal behavioural ecology*. in *2006 IEEE international conference on evolutionary computation*.

2006. IEEE.
41. Gandomi, A.H., *Interior search algorithm (ISA): a novel approach for global optimization.* ISA transactions, 2014. **53**(4): p. 1168-1183.
42. Sadollah, A., et al., *Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems.* Applied Soft Computing, 2013. **13**(5): p. 2592-2612.
43. Satapathy, S. and A. Naik, *Social group optimization (SGO): a new population evolutionary optimization technique.* Complex & Intelligent Systems, 2016. **2**(3): p. 173-203.
44. Kirkpatrick, S., C.D. Gelatt Jr, and M.P. Vecchi, *Optimization by simulated annealing.* science, 1983. **220**(4598): p. 671-680.
45. Zhang, Z.Y. and Y.L. Gao, *Solving large-scale global optimization problems and engineering design problems using a novel biogeography-based optimization with Levy and Brownian movements.* International Journal of Machine Learning and Cybernetics.
46. Rashedi, E., H. Nezamabadi-Pour, and S. Saryazdi, *GSA: a gravitational search algorithm.* Information sciences, 2009. **179**(13): p. 2232-2248.
47. Sterkenburg, T.F. and P.D. Grunwald, *The no-free-lunch theorems of supervised learning.* Synthese, 2021. **199**(3-4): p. 9979-10015.
48. Wu, G., R. Mallipeddi, and P. Suganthan, *Problem Definitions and Evaluation Criteria for the CEC 2017 Competition and Special Session on Constrained Single Objective Real-Parameter Optimization*. 2016.
49. Abhishek Kumar, et al., *Problem definitions and evaluation criteria for the CEC 2022 special session and competition on single objective bound constrained numerical optimization*. 2022.
50. Witten, T.A. and L.M. Sander, *Diffusion-limited aggregation.* Physical review B, 1983. **27**(9): p. 5686.
51. Du, P., et al., *A novel hybrid model based on multi-objective Harris hawks optimization algorithm for daily PM2. 5 and PM10 forecasting.* 2019.
52. Yao, X., Y. Liu, and G. Lin, *Evolutionary programming made faster.* IEEE Transactions on Evolutionary computation, 1999. **3**(2): p. 82-102.
53. Mirjalili, S., *Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm.* Knowledge-based systems, 2015. **89**: p. 228-249.
54. Rao, R., *Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems.* International Journal of Industrial Engineering Computations, 2016. **7**(1): p. 19-34.
55. Yang, X.-S. *Firefly Algorithms for Multimodal Optimization*. 2009. Berlin, Heidelberg: Springer Berlin Heidelberg.
56. Połap, D. and M. Woźniak, *Red fox optimization algorithm.* Expert Systems with Applications, 2021. **166**: p. 114107.
57. Cai, Z., et al., *Evolving an optimal kernel extreme learning machine by using an enhanced grey wolf optimization strategy.* Expert Systems with Applications, 2019. **138**: p. 112814.
58. Abd Elaziz, M., D. Oliva, and S. Xiong, *An improved Opposition-Based Sine Cosine Algorithm for global optimization.* Expert Systems with Applications, 2017. **90**: p. 484-500.
59. Chen, H., et al., *An efficient double adaptive random spare reinforced whale optimization algorithm.* Expert Systems with Applications, 2019.

60. Chen, H., et al., *Efficient multi-population outpost fruit fly-driven optimizers: Framework and advances in support vector machines.* Expert Systems with Applications, 2020. **142**(10.1016/j.eswa.2019.112999).
61. Heidari, A.A., R. Ali Abbaspour, and H. Chen, *Efficient boosted grey wolf optimizers for global search and kernel extreme learning machine training.* Applied Soft Computing, 2019. **81**: p. 105521.
62. Liang, H., et al., *A Hybrid Bat Algorithm for Economic Dispatch With Random Wind Power.* IEEE Transactions on Power Systems, 2018. **33**(5): p. 5052-5061.
63. Elhosseini, M.A., et al., *Biped robot stability based on an A–C parametric Whale Optimization Algorithm.* Journal of Computational Science, 2019. **31**: p. 17-32.
64. Qu, C., et al., *A Modified Sine-Cosine Algorithm Based on Neighborhood Search and Greedy Levy Mutation.* Computational intelligence and neuroscience, 2018. **2018**: p. 4231647-4231647.
65. Nenavath, H. and R.K. Jatoth, *Hybridizing sine cosine algorithm with differential evolution for global optimization and object tracking.* Applied Soft Computing, 2018. **62**: p. 1019-1043.
66. Chen, H., et al., *A balanced whale optimization algorithm for constrained engineering design problems.* Applied Mathematical Modelling, 2019. **71**: p. 45-59.
67. Zhao, H., et al., *Learning practically feasible policies for online 3D bin packing.* Science China Information Sciences, 2022. **65**(1): p. 1-17.
68. Qin, X., et al., *User OCEAN Personality Model Construction Method Using a BP Neural Network.* Electronics, 2022. **11**(19): p. 3022.
69. Lu, H., et al., *Multimodal Fusion Convolutional Neural Network with Cross-attention Mechanism for Internal Defect Detection of Magnetic Tile.* IEEE Access, 2022.
70. Yin, M., et al., *Deep Feature Interaction Network for Point Cloud Registration, With Applications to Optical Measurement of Blade Profiles.* IEEE Transactions on Industrial Informatics, 2022.
71. Zhang, K., et al., *Training effective deep reinforcement learning agents for real-time life-cycle production optimization.* Journal of Petroleum Science and Engineering, 2022. **208**: p. 109766.
72. Cao, B., et al., *A multiobjective intelligent decision-making method for multistage placement of PMU in power grid enterprises.* IEEE Transactions on Industrial Informatics, 2022.
73. Zhou, W., et al., *GMNet: graded-feature multilabel-learning network for RGB-thermal urban scene semantic segmentation.* IEEE Transactions on Image Processing, 2021. **30**: p. 7790-7802.
74. Gandomi, A., et al., *Bat algorithm for constrained optimization tasks*. Vol. in press. 2013.
75. He, Q. and L. Wang, *Wang, L.: A Hybrid Particle Swarm Optimization with a Feasibility-based Rule for Constrained Optimization. Applied Mathematics and Computation 186, 1407-1422*. Vol. 186. 2007. 1407-1422.
76. Kaveh, A. and S. Talatahari, *Talatahari, S.: A Novel Heuristic Optimization Method: Charged System Search. Acta Mechanica 213(3-4), 267-289*. Vol. 213. 2010. 267-289.
77. He, Q. and L. Wang, *An effective co-evolutionary particle swarm optimization for constrained engineering design problems*. Vol. 20. 2007. 89-99.

78. Kaveh, A. and S. Talatahari, *An improved ant colony optimization for constrained engineering design problems*. Vol. 27. 2010. 155-182.
79. Mezura-Montes, E., et al., *Multiple trial vectors in differential evolution for engineering design*. Vol. 39. 2007. 567-589.
80. Sandgren, E., *Nonlinear integer and discrete programming in mechanical design*. Vol. 14. 1988.
81. Wagdy, A., *A novel differential evolution algorithm for solving constrained engineering optimization problems*. Journal of Intelligent Manufacturing, 2017.
82. Kaveh, A. and M. Khayatazad, *A new meta-heuristic method: Ray Optimization.* Computers & Structures, 2012. **112-113**: p. 283-294.
83. Mirjalili, S., et al., *Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems*. Advances in Engineering Software, 2017. **114**: p. 163-191.
84. Huang, F.-z., L. Wang, and Q. He, *An effective co-evolutionary differential evolution for constrained optimization.* Applied Mathematics and Computation, 2007. **186**(1): p. 340-356.
85. Mirjalili, S., *SCA: A Sine Cosine Algorithm for solving optimization problems.* Knowledge-Based Systems, 2016. **96**: p. 120-133.
86. Kennedy, J. and R. Eberhart. *Particle swarm optimization*. in *IEEE International Conference on Neural Networks - Conference Proceedings*. 1995.
87. Wang, G., *Adaptive Response Surface Method Using Inherited Latin Hypercube Design Points*. Vol. 125. 2003. 210-220.
88. Cheng, M.-Y. and D. Prayogo, *Symbiotic Organisms Search: A new metaheuristic optimization algorithm*. Vol. 139. 2014.
89. Gandomi, A., X.-S. Yang, and A. Alavi, *Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems*. Vol. 29. 2013. 1-19.
90. Adarsh, B.R., et al., *Economic dispatch using chaotic bat algorithm*. Vol. 96. 2016. 666-675.
91. Eskandar, H., et al., *Water cycle algorithm – A novel metaheuristic optimization method for solving constrained engineering optimization problems.* Computers & Structures, 2012. **110-111**: p. 151-166.
92. Savsani, P. and V. Savsani, *Passing vehicle search (PVS): A novel metaheuristic algorithm.* Applied Mathematical Modelling, 2016. **40**(5): p. 3951-3978.
93. Rao, R.V., V.J. Savsani, and D.P. Vakharia, *Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems.* Computer-Aided Design, 2011. **43**(3): p. 303-315.