Journal of Big Data

RESEARCH Open Access

Advanced RIME architecture for global optimization and feature selection

Ruba Abu Khurma[1,2*], Malik Braik[3†], Abdullah Alzaqebah[4†], Krishna Gopal Dhal[5†], Robertas Damaševičius[6†] and Bilal Abu-Salih[7,8*†]

†Malik Braik, Abdullah Alzaqebah, Krishna Gopal Dhal, Robertas Damaševičius and Bilal Abu-Salih contributed equally to this work.

*Correspondence: rubaabukhurma82@gmail.com; b.abusalih@ju.edu.jo

[1]MEU Research Unit, Faculty of Information Technology, Middle East University, Amman 11831, Jordan
[2]Applied Science Research Center, Applied Science Private University, Amman 11931, Jordan
[3]Computer Science Department, Prince Abdullah bin Ghazi Faculty of Information and Communication Technology, Al-Balqa Applied University, Salt, Jordan
[4]Computer Science Department, Faculty of Information Technology, The World Islamic Sciences & Education University, Amman, Jordan
[5]Dept. of Computer Science and Application, Midnapore College (Autonomous), Paschim Medinipur, West Bengal, India
[6]Center of Real Time Computer Systems, Kaunas University of Technology, 50186 Kaunas, Lithuania
[7]King Abdullah II School of Information Technology, The University of Jordan, Amman, Jordan
[8]School of Management and Marketing, Curtin University, Perth, Australia

Abstract

The article introduces an innovative approach to global optimization and feature selection (FS) using the RIME algorithm, inspired by RIME-ice formation. The RIME algorithm employs a soft-RIME search strategy and a hard-RIME puncture mechanism, along with an improved positive greedy selection mechanism, to resist getting trapped in local optima and enhance its overall search capabilities. The article also introduces Binary modified RIME (mRIME), a binary adaptation of the RIME algorithm to address the unique challenges posed by FS problems, which typically involve binary search spaces. Four different types of transfer functions (TFs) were selected for FS issues, and their efficacy was investigated for global optimization using CEC2011 and CEC2017 and FS tasks related to disease diagnosis. The results of the proposed mRIME were tested on ten reliable optimization algorithms. The advanced RIME architecture demonstrated superior performance in global optimization and FS tasks, providing an effective solution to complex optimization problems in various domains.

Keywords: Feature selection, RIME, Optimization, Metaheuristic, Transfer function

Introduction

The pursuit of optimal solutions within the expansive and intricate realms of global optimization problems is a critical and central endeavor across a multitude of scientific and engineering domains [1,2]. These domains, ranging from computer science and operations research to various branches of engineering and applied sciences, are continually faced with challenges that are high-dimensional and multifaceted. The inherent complexity and diversity of problems within these domains necessitate the development and implementation of innovative, sophisticated, and efficient algorithms. These algorithms must be capable of navigating through the vast landscapes of high-dimensional spaces, exploring a myriad of potential solutions, and ultimately converging to solutions that are optimal or near-optimal.

Global optimization problems are characterized by their extensive search spaces and the presence of numerous local optima, making the task of finding the global optimum a highly non-trivial endeavor. The challenges posed by these problems are further compounded by the increasing dimensionality and complexity of the search spaces, requiring algorithms

with enhanced exploration and exploitation capabilities. The exploration and exploitation dichotomy is crucial in global optimization, where algorithms must balance between exploring new, unvisited regions of the search space and exploiting known promising regions to refine solutions.

The exploration of nature-inspired algorithms has emerged as a promising and fruitful avenue in addressing the challenges posed by global optimization problems. Nature-inspired algorithms draw insights, principles, and strategies from various natural phenomena, biological processes, and ecological interactions observed in the natural world [3]. By mimicking the adaptive, evolutionary, and cooperative behaviors exhibited by biological entities and ecosystems, these algorithms design sophisticated computational models and search strategies capable of solving complex optimization problems.

Nature-inspired algorithms encompass a diverse array of approaches, each inspired by different aspects of the natural world. Evolutionary algorithms, for example, are inspired by the principles of natural selection and evolution, simulating the processes of selection, crossover, mutation, and reproduction to evolve populations of solutions over generations [4]. Swarm intelligence algorithms, on the other hand, draw inspiration from the collective behaviors of social insects and animal groups, utilizing mechanisms of cooperation, communication, and self-organization to explore and optimize search spaces [5,6].

The appeal of nature-inspired algorithms lies in their inherent adaptability, flexibility, and robustness. These algorithms are capable of adapting to dynamic and uncertain environments, adjusting their search strategies in response to changing landscapes and problem constraints. The flexibility of nature-inspired algorithms allows them to be applied to a wide range of optimization problems, with the potential for customization and hybridization to suit the specific characteristics and requirements of individual problems. Furthermore, the robustness of these algorithms enables them to handle noise, uncertainties, and imperfections in problem formulations and data, providing reliable and stable performance across different problem instances. The development of nature-inspired algorithms is driven by a continuous quest for innovation and improvement. Researchers and practitioners in the field are engaged in the design and analysis of new algorithms, the enhancement of existing algorithms, and the exploration of hybrid and multi-objective approaches. The advancements in nature-inspired algorithms are fueled by interdisciplinary collaborations, bringing together expertise from computer science, mathematics, biology, physics, and other disciplines to develop more effective and efficient algorithms. The integration of theoretical foundations, empirical studies, and computational experiments is essential in understanding the underlying mechanisms of nature-inspired algorithms and in validating their performance and applicability.

The application of nature-inspired algorithms extends beyond the realm of global optimization to various areas such as machine learning, medical image processing [7], human activity recognition [8], software defect prediction [9], network intrusion detection [10], power scheduling [11], and logistics. In machine learning and data mining, for example, nature-inspired algorithms are employed for FS [12], clustering [13], classification [14], and regression, contributing to the discovery of knowledge and insights from data. In image processing and computer vision, these algorithms are utilized for segmentation, edge detection, object recognition, and enhancement, aiding in the analysis and interpretation of visual information. The versatility and efficacy of nature-inspired algorithms in

addressing diverse problems underscore their significance and potential in advancing the frontiers of science and technology.

The RIME (RIME-Ice) Algorithm, the focal point of this article, is a new contribution to the realm of nature-inspired algorithms, deriving its conceptual framework from the intricate processes of RIME-ice formation [15]. RIME-ice, a meteorological phenomenon, occurs when super-cooled water droplets freeze upon contact with surfaces, forming crystalline structures. The algorithm simulates the distinct behaviors of soft and hard RIME formations, leveraging their unique characteristics to optimize search strategies and enhance convergence and solution accuracy in diverse conditions.

In this study, we present RIME: A physics-based optimization approach as a new method for solving continuous optimization and FS problems. Our research aims to evaluate the effectiveness, robustness, and applicability of the RIME algorithm in improving solution quality and computational efficiency in these domains by examining it across continuous and discrete optimization search spaces. The main contributions of this article are summarized as follows:

- The article introduces modified RIME (mRIME), which enhances the RIME algorithm's by adopting chaotic maps for initialization of solutions, crossover for improving exploration phase of the search space and greedy search for enhancing the selection of the best solution and decreasing the bias of the optimizer.
- To facilitate the transformation of a continuous search space into a binary one, four distinct types of TFs from the S-shaped, V-shaped, U-shaped, and X-shaped families were chosen for FS problems.
- The effectiveness of RIME is examined for global optimization through the use of CEC2011, CEC2017, and FS tasks in relation to applications for disease diagnosis.
- The suggested mRIME's algorithm has been tested compared to ten of the most reliable optimization algorithms. The rival algorithms to the one being described fall into three categories: the most well studied EAs, SFS [16], DE [17], BBO [18], and GA [19]. Some SI that are well-known and reliable are PSO [20], ACO [21], and AMO [22]. The three HBOs that are the most current and successful are TLBO [23], GSK [24], and WSO [25].
- The convergence curve, running time, number of selected features, fitness value, specificity, sensitivity, and accuracy are the six standard metrics that were used to assess the mRIME.
- A range of statistical numbers were given, including the worst, average, median, standard deviation, and best values. To assess the outcomes and demonstrate the robustness of the proposed mRIME, test methods proposed by Friedman and Holm were applied. Through meticulous development and extensive evaluations, it demonstrates the superior performance and versatility of the algorithm, offering valuable insights and methodologies for researchers and practitioners in the fields of optimization and FS.

Literature review

The discussed articles provide a comprehensive overview of the advancements in optimization algorithms, focusing on nature-inspired algorithms, enhancements to existing algorithms, and their diverse applications in medical, biological, engineering, and energy

domains. The development and enhancement of these algorithms are pivotal for navigating through high-dimensional spaces and converging to optimal or near-optimal solutions in various scientific and engineering domains. The applications of these algorithms in real-world problems demonstrate their potential in providing innovative solutions in diverse fields.

Numerous articles examine the creation of algorithms that are modeled after the traits and behaviors of animals and other natural occurrences. For instance, Mohapatra [26] discussed the Golden Jackal Optimization (GJO) algorithm, inspired by the collaborative hunting behaviors of golden jackals, and proposes an enhanced variant incorporating opposition-based learning (OBL) to overcome its disadvantages . Similarly, Houssein introduces the Liver Cancer Algorithm (LCA), a bio-inspired optimization algorithm mimicking liver tumor growth and takeover processes [27]. Abdel-Basset presents the Mantis Search Algorithm (MSA), inspired by the unique hunting behavior and sexual cannibalism of praying mantises [28]. A Chimp-inspired Optimization Algorithm (COA). Remora Optimization Algorithm (ROA) [29], [30]. Arithmetic Optimization Algorithm (AOA) [31], [32]. An efficient Equilibrium Optimizer (SLEO) [33]. These articles highlight the versatility of nature-inspired algorithms in navigating high-dimensional spaces and converging to optimal solutions.

Using a novel PRISMA technique, this research assesses the evolution of the World Wide Open Access (WOA) during the last 5 years and critically analyzes it [34]. Strict inclusion criteria and screening procedures are used to improve the evaluation stage. Effective methods for hybridizing WOA variants are outlined, and 59 enhanced WOA and 57 hybrid WOA variants were chosen. Along with highlighting the dearth of thorough comparisons with earlier WOA variants, the report also provides a graphic representation of the distribution of qualifying WOA variants. There are recommendations for future paths.

To get around some restrictions and boost efficiency, a number of papers suggest improving and changing current algorithms. Hassan introduces a multi-objective variant of the marine predator's algorithm (MPA), incorporating concepts from Quantum theory to enhance the MPA's ability to balance between exploration and exploitation [35]. Mehmood combines Archimedes' optimization algorithm (AOA) with chaotic maps to optimize complex engineering problems [36]. Zhou introduces LASMA, a local dimensional mutation strategy, and an all-dimensional neighborhood search strategy for the slime mould algorithm (SMA) to improve the algorithm's exploration and exploitation abilities [37]. These enhancements aim to address the limitations of the original algorithms, such as poor exploitation ability and susceptibility to local optima.

The use of optimization algorithms in the biological and medical fields is the subject of several articles. Painul [38], this examines the latest developments in deep learning and machine learning methods for the detection and diagnosis of six different types of cancer: pancreatic, skin, lung, liver, breast, and brain. It analyzes important performance measures on benchmark datasets, including precision, accuracy, area under the curve, sensitivity, and dice score; it ends with research challenges for the future. Yu presents a hybrid model, bERIME_FKNN, for early recognition and timely treatment of Pulmonary Hypertension (PH) [39]. Emam proposes an optimized residual learning architecture for classifying multiple brain tumors, utilizing an improved variant of the Hunger Games Search algorithm (I-HGS) [40]. Chen designs a new wrapper gene selection algorithm, ABHGS, integrating

hunger Games search (HGS) with an artificial bee strategy for high-dimensional genetic data [41]. These applications demonstrate the potential of optimization algorithms in addressing real-world problems in medicine and biology, such as disease diagnosis and gene selection.

Optimization algorithms are applied in engineering design and energy areas, as discussed in several articles. Deng develops the snow ablation optimizer (SAO) for numerical optimization and engineering design, focusing on real-world constrained optimization issues in process synthesis and mechanical engineering [42]. Dong introduces the boosting kernel search optimizer (BKSO) to solve the combined economic emission dispatch (CEED) problem in power systems [43]. Zhou presents a boosted atomic search optimization (ASO) with a new anti-sine-cosine mechanism (ASCASO) for parameter estimation of photovoltaic (PV) models [37]. These studies illustrate the versatility of optimization algorithms in optimizing engineering designs and enhancing energy conversion efficiency.

For FS application, there were many metaheuristic algorithms that have been used to tackle this optimization challenge. Fatahi [44], an algorithm for FSS in medical data preparation called the Improved Binary Quantum-based Avian Navigation Optimizer Algorithm (IBQANA) is proposed in this study. To handle less-than-ideal results from binary metaheuristic algorithms, it makes use of the Hybrid Binary Operator (HBO) and the Distance-based Binary Search Strategy (DBSS). HBO transforms continuous values into binary solutions, and DBSS speeds up convergence and improves search agent performance. Twelve medical datasets are used to examine the efficacy of HBO with thresholding and five different TF families. Additionally, IBQANA outperforms all compared algorithms in the detection of COVID-19. The suggested approach offers a possible remedy for the FSS issue in the preparation of medical data.

Based on the starling murmuration optimizer (SMO), a novel binary optimizer technique named BSMO is presented by Nadimi [45]. BSMO is capable of finding the best characteristics and resolving intricate engineering difficulties. To find useful characteristics in medical datasets, it employs two methods: first, it maps each dimension to either 0 or 1, utilizing a configurable threshold; second, it creates binary versions by utilizing multiple S-shaped and V-shaped TFs. Four medical datasets were used to assess the performance of BSMO, and it was contrasted with popular binary metaheuristic algorithms. The BSMO performed better in choosing useful features than rivals such as ACO, BBA, bGWO, and BWOA.

Based on the No-Free-Lunch theorem which demonstrates that optimization problems can be solved using different optimization algorithms with different outcomes each time. Moreover, the same optimization algorithm can be enhanced using various operators and strategies and tested on the same optimization problem and produce new results each time. The stochastic nature of optimization algorithms motivated us to conduct experiments using the recently developed RIME optimizer to test its accuracy, robustness and validity when it is utilized to solve both continuous and discrete optimization problems. The first one was accomplished by testing RIME on global engineering problems and the second one was accomplished by testing RIME on feature selection optimization problems and specifically for disease diagnosis in medical applications. The conversion of the original RIME optimizer was done using different TFs that belong to four families: S-shaped, V-shaped, U-shaped, and X-shaped families. Different enhancement strategies were embedded in original RIME to improve its performance such as crossover opera-

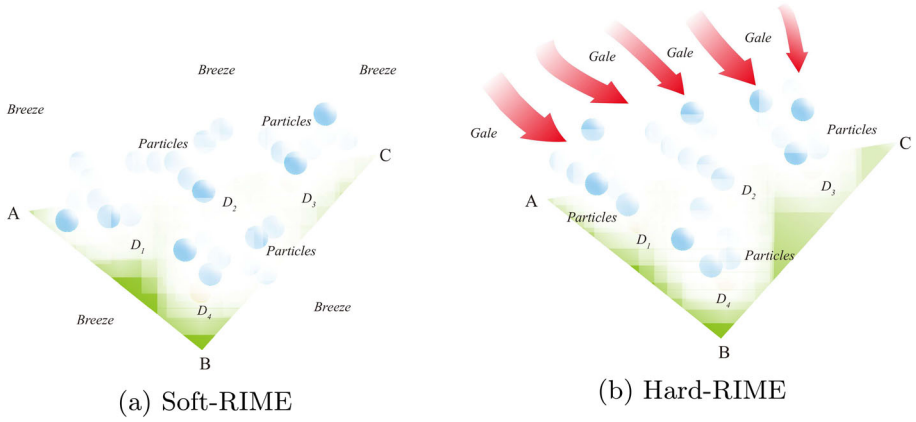

Fig. 1 The processes by which hard RIME and soft RIME form in various situations

tors to enhance exploration phase, positive greedy selection process to choose the best solution, and chaotic functions to initialize solutions.

Methodology

Original RIME algorithm background

The building up of evaporated water in the environment that hasn't yet solidified leads to RIME-Ice. At temperatures below zero, it succumbs to freezing and adheres to things like branches from trees. Some areas form a winter RIME-Ice landscape because of their distinct topography and weather-related traits. The formation of RIME ice is influenced by a number of factors, including temperature, moisture levels, wind velocity, and atmospheric pressure. While RIME-Ice can grow and expand, its growth is not limitless. External factors and the inherent nature of its formation ultimately halt its expansion, leading to a state of relative stability. To better understand the dynamics of RIME ice formation, consider the hypothetical scenario depicted in Fig. 1. The plane "ABC" represents the RIME formation area, and points (D_1, D_2, D_3, and D_4) represent the birthplaces sites where RIME ice begins to form. The formation of RIME ice typically falls into two distinct categories: soft RIME and hard RIME. Soft RIME, often associated with gentle breezes, is characterized by its fine, granular structure. The formation of soft RIME is a gradual process, as the wind slowly deposits ice crystals onto the nucleation sites. This delicate RIME ice is often ephemeral, susceptible to the whims of wind and temperature changes. In contrast, hard RIME, a product of strong winds, exhibits a denser, more compact structure. The formation of hard RIME is a rapid process, as the wind forcefully impinges ice crystals onto the nucleation sites. This robust RIME ice can withstand harsher conditions, persisting long after the winds have subsided. Figure 1a depicts a breeze as having low wind speeds, variable direction changes and constant presence at every angle at an identical level. As such, its delicate RIME develops slowly and unpredictably whereas gale winds can be identified by their fast speeds with an almost uniform direction, producing hard RIME growth more rapidly in one or several directions simultaneously. Gale winds can also produce large rainfall amounts in any given location at one or more height levels, producing rapid rainfall amounts with hard RIME formation as in Fig. 1b.

The RIME algorithm is motivated by the Ice-RIME development mechanism and provides a Soft-RIME search approach by simulating the motion of Soft-RIME particles. A

Hard-RIME puncture mechanism is also suggested to make use of the algorithm by imitating the behavior of hard-RIME agents in crossover situations. Finally, the metaheuristic algorithm's selection mechanism is enhanced, and a positive greedy selection mechanism is suggested. By fusing the aforementioned three methods, the RIME algorithm is developed, which has improved performance. Modeling the RIME mathematically: The process of forming each RIME strip in the RIME algorithm involves a detailed examination of various factors such as wind speed, freezing coefficient, the cross-sectional area of the connected material, and the duration of growth. These factors collectively influence the development of each RIME strip.

In contrast, the process of forming a RIME agent from RIME particles is simulated by modeling the progressive activity of each particle. This progression leads to the formation of the final RIME agent, which is akin to a piece of crystal. This simulation approach is inspired by the diffusion-limited aggregation method, commonly used for simulating the aggregation of metal particles. The RIME algorithm is structured into four distinct stages:

1. Initialization of RIME Clusters: This initial stage sets up the framework for the formation of RIME structures. It involves preparing the initial conditions and parameters that govern the growth and development of the RIME clusters.
2. Suggested Soft-RIME Search Method: In this stage, a method is proposed for the searching and growth of soft-RIME. This might involve algorithms or mechanisms that simulate the accumulation and adhesion processes characteristic of soft RIME.
3. Suggested Hard-RIME Puncture Mechanism: This stage deals with the transition of RIME from a soft to a hard state. It proposes a mechanism for the puncture process, which is a critical phase in the formation of hard RIME, known for its denser and more compact structure.
4. Suggested Greedy Selection Mechanism Enhancement: The final stage focuses on enhancing the selection mechanism, potentially using a greedy approach. This could involve selecting the most optimal or favorable conditions or parameters that facilitate the efficient growth and formation of RIME.

1. Initialization of the RIME group:

The RIME algorithm, which draws its inspiration from reality, views the population of the algorithm as the complete RIME-population and treats each agent's RIME as the algorithm's sought agent. At the start, R's full RIME population is initialized. According to Eq. (1), the population of RIME is composed of n RIME agents (S_i), each of which is composed of d RIME particles (x_{ij}). As a result, the RIME-particles x_{ij} can accurately reflect the RIME-population R, as demonstrated in Eq. (2).

$$R = \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{pmatrix}; S_i = [x_{i1} x_{i2} \cdots x_{ij}] \tag{1}$$

$$R = \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ \vdots & & \\ x_{i1} & x_{i2} & x_{i3} \end{pmatrix} \tag{2}$$

where (i) and (j) represent the ordinal numbers for a RIME agent and particle respectively, and $F(S_i)$ represents each agent's growth state or fitness value in the metaheuristic algorithm.

2. Soft-RIME searching approach:

1. Each particle, x_{ij}, follows its own set of laws before condensing into soft RIME agents; their ability to wander is subject to external influences and can vary accordingly.
2. if free-state RIME particles migrate near Soft-RIME agents, they may condense with its particles and alter its stability.
3. Given that each particle experiences variable condensation levels, the distance between centers of two particles that adhere is not constant.
4. Inter-particle condensation does not occur if particles move straight outside their escape radius.
5. As the random condensation process unfolds, the area to which each particle adheres expands, raising the probability of free particle condensation during soft RIME formation. Nevertheless, environmental factors ultimately lead the agent to reach a stable state, halting its growth.

Within the RIME algorithm, the estimation of RIME particle positions is calculated as outlined in Eq. (3), aligning with the five distinct motion traits of RIME particles.

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

The RIME algorithm employs a particle movement strategy that incorporates both Soft-RIME and Hard-RIME dynamics. The position of each particle, denoted by R_{ij}^{new}, is updated based on its current position i and j, the best RIME-Agent particle $R_{(best,j)}$, and a random parameter h. The direction of particle movement is influenced by the cosine function, which is modulated based on the number of iterations. The environmental component, represented by the term after the number of iterations in Eq. (5), ensures algorithm convergence and mimics the influence of external factors. The degree of adhesion, denoted by h, is a random value between 0 and 1 that controls the spacing between RIME particles.

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

This iterative process continues until the algorithm's current iteration count, denoted by t, reaches the maximum allowed iteration count, represented by T.

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

The default value of w is set to 5, which controls the number of segments in the step function. [.] denotes rounding in this context, where the step function is the mathematical model of the process. Referring back to Eq. (3), the upper and lower bounds of the escape space, represented by the letters Ub_{ij} and Lb_{ij}, respectively, define the boundaries of the particle motion's practical range. E represents the attachment coefficient, which influences an agent's likelihood of condensing and increases with the number of iterations. It is represented in Eq. (6).

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

Algorithm 1 The Soft-RIME search technique's pseudo-code

```
while t ≤ T do
  Coefficient of adherence E = √(t/T)
  for i = 1 : n do
    for j = 1 : d do
      if r2 < E then
        Adjustment of placement based on RIME particle properties using Eq (3)
      end if
    end for
  end for
  Change the existing optimum solution and optimum fitness. t = t + 1
end while
```

3. Technique for Hard-RIME pierces:

Hard-RIME growth is easier and more dependable in strong gale conditions than soft-RIME growth. The features of a hard RIME are as follows when the RIME particle condenses:

- Because the gale is so powerful, other influences are insignificant, which causes several Hard-RIME agents to snowball in the same direction.
- Because each RIME agent can readily cross across because the growth direction is the same, this is known as RIME puncture.
- Hard-RIME agents, like Soft-RIME agents, become larger as they mature, increasing the likelihood of puncturing between agents under favorable conditions for growth.

Consequently, the puncturing phenomenon and the associated Hard-RIME puncture mechanism enable the algorithm to exchange particles between solutions, thereby improving convergence and enhancing the ability to escape local optima. The particle replacement formula is depicted in Eq. (7).

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

where R_{ij}^{new} represents the updated position of the particle and $R_{best,j}$ represents the j_{th} particle of the best RIME-Agent in the RIME population R. The normalized value of the current agent's fitness value, denoted by $F^{normr}(S_i)$, represents the probability of selecting the i_{th} RIME-Agent. r_3 is a random number between (-1) and (1). Algorithm 2 presents the pseudo-code for the Hard-RIME puncture mechanism.

4. An efficient technique of greedy selection:

In conventional metaheuristic optimization algorithms, the greedy selection mechanism regularly updates and records the best fitness value and the corresponding solution. Following each update, the solution's updated fitness value is typically compared to the global optimum. If the updated value surpasses the existing global optimum, the optimal fitness value is replaced, and the solution is designated as the new optimum. While this approach is simple and efficient, it does not directly contribute to exploring or exploiting the population as it primarily serves as a record-keeping mechanism.

An aggressive greedy selection strategy is often employed in optimization algorithms to increase global exploration's effectiveness, using fitness values of agents before and after updates as an indication of effectiveness or proximity of optimal solutions. At each update step, the algorithm compares updated fitness values against their previous fitness values;

Algorithm 2 The Soft-RIME search technique's pseudo-code

Set up the RIME-population R
Obtain the most effective agent and fitness at the moment.
while $t \leq T$ **do**
  for $i = 1 : n$ **do**
    for $j = 1 : d$ **do**
      if $r_3 <$ *Normalize fitness of* S_i **then**
        changing the location depending on the RIME-particles' properties by Eq (7)
      end if
    end for
  end for
  Change the existing optimum solution and optimum fitness
  $t = t + 1$
end while

fitness can serve as a barometer for measuring how effectively agents solve their assigned problem, with those surpassing previous values being replaced or updated both regarding solution and properties to reflect this newfound success.

The implications of this strategy are two folds:

- By actively replacing agents with improved versions, the overall quality of the population is raised. Good agents, those with higher fitness values, are consistently maintained in the population, driving the collective towards better solutions.
- While this method ensures the population is always stocked with high-performing agents, the dramatic shift in the positions of the population's agents with each iteration carries a risk. Some agents, as a result of these shifts, may end up performing worse than they did before the update. This degradation can negatively impact the population in the following iteration, as not all changes lead to improvements.

Algorithm 3 displays the pseudo-code of the positive greedy selection mechanism for addressing the minimum value problem.

Algorithm 3 The Soft-RIME search technique's pseudo-code

Set up the RIME-Size R
Obtain the most effective agent and fitness at the moment.
while $t \leq T$ **do**
  for $i = 1 : n$ **do**
    for $j = 1 : d$ **do**
      if $F(R_i^{new}) < F(R_i)$ **then**
        $F(R_i) = F(R_i^{new})\ R_i = R_i^{new}$
        if $F(R_i^{new}) < F(R_{best})$ **then**
          $F(R_{best}) = F(R_i^{new})\ R_{best} = R_i^{new}$
        end if
      end if
    end for
  end for
  Change the existing optimum solution and optimal fitness $t = t + 1$
end while

The recommended RIME algorithm:

The overall organization of the algorithm in pseudo-code is illustrated in Algorithm 4 The algorithm presented here integrates several groundbreaking techniques inspired by the natural phenomena of RIME formation. These techniques contribute to the optimization process in unique ways:

1. Soft-RIME Search Strategy: This primary optimization strategy draws inspiration from the movement and accumulation of Soft-RIME particles. Characterized by its delicate, crystalline structure, soft RIME embodies a gentle and exploratory approach, particularly in the early stages of the optimization process. This method prioritizes exploration over exploitation, enabling the algorithm to comprehensively traverse the solution space.
2. Inspired by the crossover behavior of Hard-RIME particles: this mechanism facilitates dimensional crossover exchange between ordinary and ideal agents. Hard-RIME, with its denser and more compact structure, represents a more focused and intensive search strategy. This crossover interchange plays a crucial role in improving the algorithm's solution accuracy, leading to a more refined and precise optimization process.
3. Improved Positive Greedy Selection Mechanism: Building upon the traditional greedy selection mechanism, this improved version is designed to expand the population diversity. By actively selecting optimal solutions and constantly refreshing the population, the mechanism aims to avoid premature convergence to local optima. This approach ensures that the algorithm does not settle for suboptimal solutions too early and continues to search for better options, thus maximizing the potential for finding the global optimum.

Each strategy contributes significantly to the algorithm's overall performance. The soft-RIME search strategy facilitates extensive exploration of the solution landscape. Subsequently, the hard-RIME puncture mechanism implements a more concentrated approach to refine the solutions. Finally, the enhanced positive greedy selection mechanism preserves diversity and hinders stagnation, guaranteeing consistent progress toward the optimal solution. Collectively, these strategies establish a balanced and dynamic optimization process, effectively traversing intricate solution spaces.

Computational complexity of mRIME

The positive greedy selection procedure, the hard-rime puncture system, the soft-rime search approach, and the fitness value computation are the key components of the complexity of the RIME algorithm. First, the soft-rime search method has a complexity issue of $\mathcal{O}(n^2)$. Next, in the two extreme situations, $\mathcal{O}(n)$ and $\mathcal{O}(n^2)$ represent the complexity issue of the hard-rime puncture mechanism. The mechanism of positive greedy selection has a complexity issue of $\mathcal{O}(n)$. Ultimately, $\mathcal{O}(m * logn)$ represents the complexity issue of the fitness value computation. Hence, $\mathcal{O}(RIME) = \mathcal{O}((n + logn) * n)$ represents the overall complexity issue of the RIME method.

Algorithm 4 The soft-RIME search technique's pseudo-code

Set up the RIME-population R
Obtain the most effective solution and fitness at the moment.
while $t \leq T$ **do**
Coefficient of adherence $E = (t/T)^{0.5}$
if $r_2 < E$ **then**
Change RIME solution position by the soft-RIME search strategy
end if
if $r_3 <$ *Normalize fitness of* S_i **then**
Using the hard-RIME piercing method, solutions can cross-update information.
end if
if $F(R_i^{new}) < F(R_i)$ **then**
Use the positive greedy selection mechanism to choose the ideal answer and swap out the undesirable one.
end if
t=t+1
end while

Algorithm 5 A synopsis of the key pseudo-code steps of the mRIME.

$t \leftarrow$ Iteration counter.
$T \leftarrow$ maximum number of iterations.
Initialize the position of RIME search agents using a chaotic vector (comprises two different chaotic maps) and determine the cost function
Obtain the best current search agent and fitness
while $(t \leq T)$ **do**
Define the coefficient of adherence $E = (t/T)^{0.5}$
if $r_2 < E$ **then**
Amend RIME agent position with the use of the soft-RIME search technique
Modify chaotic vector for various chaotic maps
end if
if $r_3 <$ *Normalize fitness of* S_i **then**
Cross-amending between search agents with the use of the hard-RIME puncture technique
Modify chaotic vector for various chaotic maps
end if
if $F(R_i^{new}) < F(R_i)$ **then**
Use the positive greedy selection process to choose the best solution and swap out the sub-optimal one.
end if
$t = t + 1$
end while%STATE return T

Binary RIME for FS

The primary step in adapting the RIME method for a search approach in FS problems involves converting it into a binary format. This conversion is necessary because the original RIME is suited only for continuous optimization challenges. However, FS problems inherently require a binary search space, represented by values of "1" or "0". This adaptation is crucial for enabling an algorithm initially designed for continuous optimization to tackle binary optimization issues effectively. To achieve this, certain operators within RIME must be modified to create its binary version. This new binary mRIME then outputs results in binary form. The process of transforming RIME into mRIME is detailed in sub-

section "RIME using different TFs", and the corresponding objective function is described in subsection "Objective function of the proposed mRIME".

RIME using different TFs

To maintain the underlying structure of the RIME method while creating a binary version, a TF is introduced. This function determines the probability that an element y^i in RIME's solution subset will be restricted to binary choices: either selected ("1") or not selected ("0"). In essence, a value of "1" indicates that the corresponding feature has been included, while a value of "0" implies that the feature has been excluded.

Logistic transformation functions, characterized by their S-shaped curve, are particularly useful for mapping operations due to their ability to produce results within the desired range of [0, 1]. This range is crucial for representing the probability of switching an element in a binary solution between "1" and "0". Kennedy et al. underscored the significance of this feature in their work [46]. Mirjalili et al. [47] further introduced the V-shaped family of transformation functions, which exhibit comparable performance to the S-shaped family in various tasks. The slope of the transformation function plays a pivotal role in determining the effectiveness of both exploitation and exploration. A flatter or less steep curve may lead to insufficient exploitation and a tendency to get stuck in local optima, while an excessively steep curve can hinder exploration [48].

Subsequently, Mirjalili et al. proposed a U-shaped TF, distinguished by two control parameters, η and χ, which govern the slope and the width of the function's basin, respectively [49]. Recognizing the shortcomings of prevalent TFs in the literature, the study also implemented an X-shaped TF, initially introduced by Ghosh et al. ghosh2020binary to generate binary counterparts of continuous optimization algorithms. This diverse range of TFs facilitates the efficient conversion of continuous solutions into binary representations, effectively tackling various challenges and characteristics of the optimization problems.

This study investigates four distinct TFs from different categories: S-shaped, V-shaped, U-shaped, and X-shaped, to address the absence of a universally acknowledged best TF for FS problems Add a reference to support your claim. These TFs were adapted and evaluated to determine the most effective one when combined with the proposed mRIME and the basic RIME algorithms. Each TF plays a critical role in influencing the probability of updating elements in the binary solution, specifically toggling between "1" and "0". The study includes visual representations of these TFs, providing a clearer understanding of their operation and impact on the binary solution. This in-depth analysis aims to improve the effectiveness and efficiency of the mRIME and RIME algorithms in the context of FS problems.

The efficacy of the TFs will be assessed in the study's experimental results section. Four distinct types of TFs from the S-shaped, V-shaped, U-shaped, and X-shaped families were chosen to facilitate the conversion of a continuous search space into a binary one for addressing FS problems. These are briefly described as follows:

- S-shaped TFs: This function is represented by the sigmoid function, as defined in Eq. (8) and originally discussed in the work of Kennedy et al. [46]. It is depicted in Fig. 2a and is part of the S-shaped family. The primary role of this function is to convert the search space from a continuous format to a binary one, effectively adapting it for FS tasks. The sigmoid function is known for its characteristic 'S' shape,

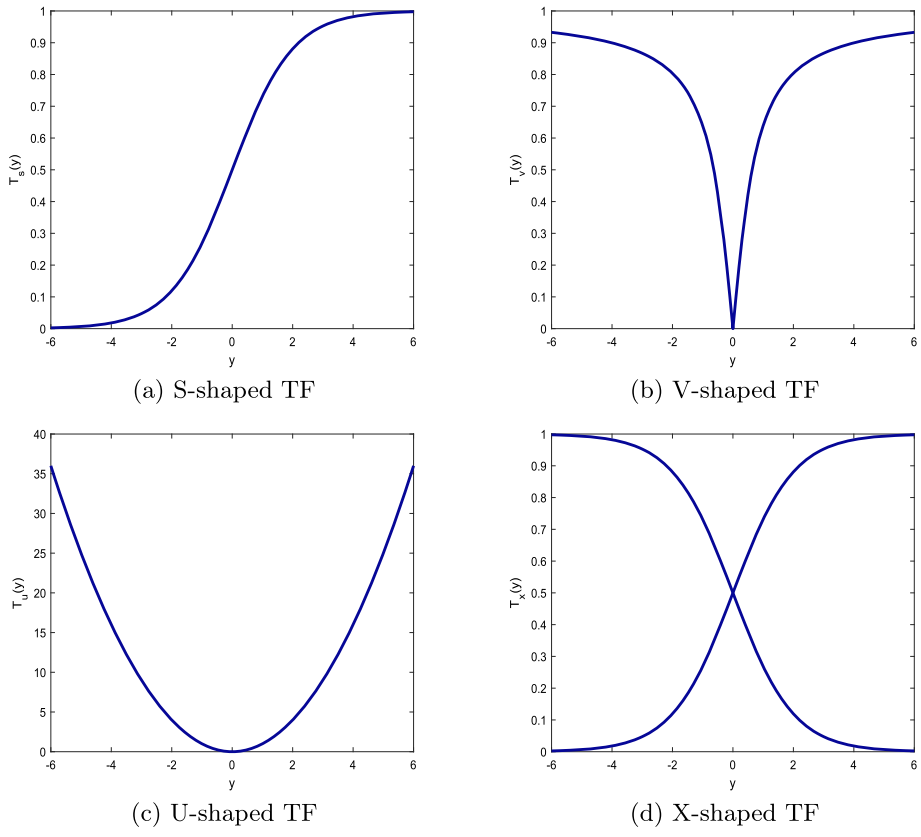

Fig. 2 Different types of TFs

which provides a smooth and gradual transition between the binary states, making it a suitable choice for this transformation process.

$$T_s\left(v_{t+1}^{i,j}\right) = \frac{1}{1+e^{-v_t^{i,j}}} \tag{8}$$

In this context, T_s is the transmutation vector, and $T_s\left(v_{t+1}^{i,j}\right)$ represents the probability value generated by the S-shaped TFs. The variables $v_t^{i,j}$ and $v_{t+1}^{i,j}$ signify the current and subsequent velocities, respectively, of the ith search agent in dimension j at iterations t and $t+1$.

The S-shaped TF, defined in Eq. (8), is a robust function that effectively transforms an unbounded input into a bounded output, mapping any input range to the interval [0, 1]. As depicted in Fig. 2a, the likelihood of modifying the position value within the search space increases as the slope of the S-shaped TF decreases. This characteristic can efficiently update the positions of search agents, thereby facilitating the discovery of optimal solutions. The effectiveness of the S-shaped TF is further enhanced due to its increasing computational speed for determining position values.

In this framework, the S-shaped TF in Eq. (8) serves to transform the search agent's velocity into a probability value. This probability value, in turn, guides the calculation of the next position, $y_{t+1}^{i,j}$, of the solution's elements. During the subsequent iteration, these elements will either transition to "1" or maintain their current state of "0". This transition is governed by a widely used stochastic threshold, ensuring that the output

of the Sigmoid function preserves its binary nature, as explained in Eq. (9). This approach enables a refined and probabilistic adjustment of the solution's elements, aligning with the requirements of binary optimization in feature selection (FS) tasks.

$$y_{t+1}^{i,j} = \begin{cases} 1 & \text{if } r_1 < T_s\left(v_{t+1}^{i,j}\right) \\ 0 & \text{if } r_1 \geq T_s\left(v_{t+1}^{i,j}\right) \end{cases} \tag{9}$$

In the context provided, r_1 is a uniformly distributed random number within the range [0, 1]. The term $y^{i,j}t+1$ represents the position of the ith search agent in dimension j at iteration $t+1$, which adopts a new binary value corresponding to the jth dimension of the ith solution at iteration $t+1$. The function $T_s\left(v_t^{i,j}\right)$ generates a probability value based on the TF described in Eq. (8).

According to Eq. (9), the velocity of the search agents is utilized to calculate the probability of changing their positions. Specifically, if the output from the TF in Eq. (9) is greater than the random value r_1, then the position $y^{i,j}t+1$ is set to "1". This indicates that the corresponding feature is considered significant and is selected. Conversely, if the output is less than or equal to r_1, the position $y_{t+1}^{i,j}$ is set to "0", suggesting that the feature is not essential and is hence excluded from consideration. The stochastic nature of the value r_1 plays a crucial role in this process. It introduces randomness into the decision-making process, determining whether the value of the solution $y_{t+1}^{i,j}$ will change. This randomness, in combination with the probability value $T_s\left(v_{t+1}^{i,j}\right)$ derived from Eq. (8), drives the update mechanism for the search agents' positions, ensuring a balanced approach between exploration and exploitation in the FS process. In the scenario where the value of $T_s\left(v_{t+1}^{i,j}\right)$ is low, the likelihood of changing the subsequent iteration value $y_{t+1}^{i,j}$ is also low. However, a critical observation regarding the sigmoid TF, as outlined in Eq. (9), is that its current form may not provide an optimal balance between exploration and exploitation. Ideally, the exploration rate should be higher than the exploitation rate at the early stages of the optimization process. Without this balance, some promising areas of the search space might not be adequately explored, leading to the possibility that the proposed Binary mRIME could become trapped in local optima.

This issue is also evident during the exploitation phase. One inherent limitation of the S-shaped family of TFs in some meta-heuristic algorithms is that the update of search agents is dependent on their velocity value. In situations where the velocity value is zero, the search agents should ideally not move. However, in practice, a zero velocity is converted to "1" or "0" with a probability of 0.5, as noted by Ghosh et al. [50]. Attempts have been made by various researchers to rectify this flaw, but they have not been entirely successful in preventing the entrapment in local optima.

- V-shaped TFs: Next, we consider the V-shaped TFs, as specified in Eq. (10) and shown in Fig. 2 (b), which was developed by Rashedi et al. [51]. This function is utilized to calculate the probability of altering the position of search agents from a continuous to a binary search space in both the fundamental and proposed FS algorithms. The V-shaped TFs, as its name suggests, have a distinctive V-shaped profile that influences how the probability of changing position values is computed, potentially offering different characteristics in the exploration and exploitation phases compared to the

S-shaped TFs.

$$T_v\left(v_{t+1}^{i,j}\right) = \left|\frac{2}{\pi}\arctan\left(\frac{\pi}{2}v_t^{i,j}\right)\right| \tag{10}$$

where T_v is the V-shaped TF, and $T_v\left(v_{t+1}^{i,j}\right)$ identifies the probability of the V-shaped TF of the velocity, $v_{t+1}^{i,j}y$, for the ith search agent at dimension j and iteration $t+1$. As illustrated in Fig. 2b, the V-shaped TF, as delineated in Eq. (10), distinguishes itself from the S-shaped TF outlined in Eq. (8) through its unique structure and rules. The V-shaped TF, characterized by its distinct 'V' shape, offers a different approach to transforming the continuous solution into a binary one. The process of this transformation utilizes Eq. (11), which effectively converts the continuous solutions derived from Eq. (10) into binary values. This conversion is based on the probability outcomes obtained from the V-shaped TF. The design of this function is such that it addresses certain aspects of optimization that the S-shaped function may not fully capture, especially in terms of the balance between exploration and exploitation in the search space. This approach underscores the importance of selecting appropriate TFs in FS algorithms, as different functions can significantly influence the performance and effectiveness of the algorithm in navigating the search space and avoiding local optima. The V-shaped TF, with its unique characteristics, is thus a critical component in the study's exploration of efficient FS methodologies.

$$y_{t+1}^{i,j} = \begin{cases} \neg y_t^{i,j} & \text{if } r_1 < T_v\left(v_{t+1}^{i,j}\right) \\ y_t^{i,j} & \text{if } r_1 \geq T_v\left(v_{t+1}^{i,j}\right) \end{cases} \tag{11}$$

In the given context, $y^{i,j}t$ represents the position of the ith search agent in dimension j at iteration t. The term $\neg y^{i,j}t$ is the complement of the solution at this position, meaning it inverts the binary value of $y^{i,j}t$. The variable r_1 is a uniformly distributed random number between 0 and 1. The function $T_v\left(v^{i,j}t+1\right)$ denotes the probability value generated by the V-shaped TF.

An important characteristic of the V-shaped TFs, as shown in Fig. 2b, is its symmetrical nature. This symmetry plays a role in how the positions of the search agents are updated. According to Eq. (11), the updating process for the search agents involves flipping their positions, rather than simply assigning them the values of "1" or "0" based on a threshold or probability. This method of position updating differs significantly from other TFs like the S-shaped one, potentially offering a more dynamic approach in the exploration and exploitation phases of the optimization process.

This flipping mechanism within the V-shaped TF allows for a more nuanced and flexible response to the search space, as it does not strictly bind the agents to binary extremes but rather provides a probabilistic approach to toggling their positions. This aspect is particularly relevant in complex FS problems, where the ability to adaptively explore and exploit the search space can lead to more effective solutions.

The operation of the V-shaped TF is such that if the velocity value of a search agent is high, the agent's position is switched to its opposite value. Studies, including those by Mirjalili et al. [47], have shown that V-shaped TFs can sometimes outperform S-shaped TFs in terms of efficiency. A key feature of the V-shaped TFs is that they encourage search agents to maintain their current positions when the velocity value

is low during an iteration, as noted by Ghosh et al. [50]. Conversely, when the velocity is high, the search agents are induced to switch to their complementary positions.
This characteristic has a significant impact on the updating of the positions of search agents and consequently on the identification of the best solution. While the V-shaped TF effectively addresses the issue of meta-heuristics encountering a zero-position value, it still faces challenges about falling into local optima. In essence, problems similar to those observed with the S-shaped TF may persist, leading to a potential imbalance between exploration and exploitation phases in meta-heuristic algorithms. In recognition of these challenges, this work explores the use of other types of TFs to achieve a more favorable balance between exploration and exploitation. Alongside the S-shaped and V-shaped TF, additional U-shaped and X-shaped TFs are employed to transform continuous algorithms into binary formats. The inclusion of these varied TFs aims to enhance the algorithm's ability to navigate the search space more effectively, reducing the likelihood of getting trapped in local optima and increasing the chances of finding optimal solutions in FS tasks.

- U-shaped TFs: The U-shaped TF, as specified in Eq. (12) and developed by Mirjalili et al. [49], is another approach explored in this study. This function is visually represented in Fig. 2c. The U-shaped TF is employed to calculate the probability of altering the positions of search agents from a continuous to a binary search space in both fundamental and proposed algorithms within the study.
The U-shaped TF is characterized by its U-shaped curve. This shape influences how the function processes and transforms velocity values into probabilities, which in turn determines the positional updates of the search agents. The distinctive aspect of the U-shaped TF is its ability to provide a different mechanism for balancing exploration and exploitation compared to the S-shaped and V-shaped functions.
By implementing the U-shaped TF, the study aims to explore whether this function offers better performance in terms of avoiding local optima and effectively navigating the search space, particularly in complex FS problems. The incorporation of the U-shaped TF into the optimization algorithms reflects an effort to diversify the strategies employed in transforming continuous solutions into binary ones, thereby potentially enhancing the overall effectiveness of the FS process.

$$T_u\left(v_{t+1}^{i,j}\right) = \eta \left| \left(v_t^{i,j}\right)^{\chi} \right| \qquad \eta = 1, \chi = 1.5, 2.0, 3.0, 4.0 \tag{12}$$

In the U-shaped TF, as outlined in Eq. (12), two crucial control parameters play a significant role: η and χ. The parameter η is responsible for defining the slope of the function, while χ determines the width of the curve's basin. The function $T_u\left(v_{t+1}^{i,j}\right)$ represents the probability of velocity for the solution concerning the search agent i at dimension j and iteration t.
The U-shaped TF, with its distinct shape and control parameters, offers a unique mechanism in the transformation process. The parameter η adjusts the saturation point of the function, and χ sets the width of the trough. The speed at which the U-shaped function reaches its saturation point affects the likelihood of flipping a bit in the solution. This characteristic promotes exploration by allowing for rapid variation in variables. A broader U-shaped curve translates to reduced exploratory behavior, while a steeper curve, proportional to the value of η, enhances exploration, as can be more distinctly seen in Fig. 2c.

The values of the continuous solution elements, as given in Eq. (12), can be converted into binary format using Eq. (13). This conversion mechanism leverages the properties of the U-shaped TF to navigate the search space effectively, balancing exploration and exploitation by adjusting the parameters η and χ. The U-shaped TF thus contributes to the overall strategy of optimizing FS algorithms by providing a distinct approach to managing the transformation of continuous solutions into binary ones.

$$y_{t+1}^{i,j} = \begin{cases} \neg y_t^{i,j} & \text{if } r_1 < T_u\left(v_{t+1}^{i,j}\right) \\ y_t^{i,j} & \text{if } r_1 \geq T_u\left(v_{t+1}^{i,j}\right) \end{cases} \tag{13}$$

In the context of the U-shaped TFs, as mentioned, r_1 signifies a uniformly distributed random number ranging between 0 and 1, and $T_u\left(v_{t+1}^{i,j}\right)$ denotes the probability value generated by the U-shaped TF.

As per Eq. (13), the future position of a search agent, $y_{t+1}^{i,j}$, is determined based on the probability value $T_u\left(v_{t+1}^{i,j}\right)$, as derived from Eq. (12). The role of the random values generated by r_1 is pivotal in deciding whether the current solution's value $y_t^{i,j}$ at a given iteration is inverted. Consequently, if the probability value $T_u\left(v_{t+1}^{i,j}\right)$ is small, the likelihood of inverting the value in the next iteration is also minimized.

In the context of optimization, the initial phases of iteration prioritize exploration to ensure a comprehensive search of the available space. This stage is crucial for identifying various potential solutions. As the process transitions from exploration to exploitation, the latter becomes vital in the final iterations for pinpointing the most effective solutions.

Comparing the U-shaped TF in Fig. 2c with the V-shaped TF in Fig. 2b, its observable that while both have similarities, the U-shaped TF might offer a higher rate of exploration compared to the V-shaped TF. This enhanced exploratory capability could potentially make the U-shaped TF more effective in certain scenarios, particularly where a broader search of the solution space is required. Thus, in the quest for optimal FS, the U-shaped TF could be a superior choice over other TFs, especially in cases where avoiding premature convergence to local optima is crucial.

- X-shaped TFs: The study also incorporates an X-shaped TF to address the limitations of the more traditionally used TFs found in the literature. As depicted in Fig. 2d, the X-shaped TF is distinctive in its approach, utilizing two components to generate different outcomes. The process involves comparing these outcomes to the previous solution to determine the best result.

  If the newly generated solution surpasses the previous solution in terms of effectiveness, it is adopted as the next position. However, if it is not superior, a crossover operation is implemented between the new and former solutions. The crossover operation is designed to combine elements of both solutions, to retain advantageous properties from the previous iteration. The outcome that emerges as the most effective from this crossover process is then selected as the new position.

  This methodology introduces a new element to the optimization process, providing a mechanism for the new solution to inherit beneficial attributes from the solution of the previous iteration. Such an approach enhances both the exploration and exploitation capabilities of the proposed mRIME.

To facilitate these operations, Eqs. (14) and (16) are employed. Notably, Eq. (16) acts as a mirror image of the first, as stated by Ghosh et al. [50]. This mirrored structure of the X-shaped TF allows for a more dynamic and adaptive optimization process, potentially leading to more effective exploration of the search space and a better balance between exploration and exploitation in the quest for optimal solutions.

$$T_{y1}\left(v_{t+1}^{i,j}\right) = \frac{1}{1+e^{-v_t^{i,j}}} \tag{14}$$

$$y_{1_{t+1}}^{i,j} = \begin{cases} 1 & \text{if } r_1 < T_{y1}\left(v_{t+1}^{i,j}\right) \\ 0 & \text{if } r_1 \geq T_{y1}\left(v_{t+1}^{i,j}\right) \end{cases} \tag{15}$$

$$T_{y2}\left(v_{t+1}^{i,j}\right) = \frac{1}{1+e^{v_t^{i,j}}} \tag{16}$$

$$y_{2_{t+1}}^{i,j} = \begin{cases} 1 & \text{if } r_2 > T_{y2}\left(v_{t+1}^{i,j}\right) \\ 0 & \text{if } r_2 \leq T_{y2}\left(v_{t+1}^{i,j}\right) \end{cases} \tag{17}$$

where $y_{1_{t+1}}^{i,j}$ and $y_{2_{t+1}}^{i,j}$ are the binary versions of the solutions produced by Eqs. (14) and (16), respectively, and r_1 and r_2 are random numbers created within the range [0, 1].

As per Eqs. (15) and (17), the new solutions can be formed as follows:

$$\acute{y}_{t+1}^{i,j} = \begin{cases} y_{1_{t+1}}^{i,j} & \text{if } fit(y_{1_{t+1}}^{i,j}) < fit(y_{2_{t+1}}^{i,j}) \\ y_{2_{t+1}}^{i,j} & \text{if } fit(y_{1_{t+1}}^{i,j}) \geq fit(y_{2_{t+1}}^{i,j}) \end{cases} \tag{18}$$

where *fit* represents the fitness function of the FS problems.

In the optimization process involving the X-shaped TF, a critical step is the evaluation and comparison of fitness values of the current and newly generated solutions. Specifically, if the fitness of the new solution, denoted as $\acute{y}^{i,j}t+1$, is better than that of the current solution $y^{i,j}t$ (i.e., $fit(\acute{y}^{i,j}t+1) < fit(y^{i,j}t)$), then the new solution $\acute{y}_{t+1}^{i,j}$ is adopted for the next iteration.

However, if this condition is not met, a crossover operation is performed between $\acute{y}^{i,j}t+1$ and $y^{i,j}t$. This crossover process generates two offspring, from which the one with the best fitness is selected as the subsequent solution. This approach allows the 'child' solution to potentially retain beneficial attributes of the 'parent' solution $y_t^{i,j}$, thereby preserving advantageous characteristics while exploring new possibilities.

The specific type of crossover used in this study is the uniform crossover, as described by Syswerda in 1993 [52]. In uniform crossover, each bit of the offspring is independently chosen from one of the corresponding bits of the parents, offering a more diverse and random mixing of parental features. This method is summarized in Algorithm 6.

The inclusion of the uniform crossover in the optimization process adds a layer of diversity and adaptability to the algorithm. This can be particularly beneficial in avoiding local optima and ensuring a more thorough exploration of the search space, thereby enhancing the overall efficacy of the FS process.

To summarize the process of transforming the continuous RIME into a binary version, the velocity of each search agent is mapped to a probability value within the range [0, 1]. This mapping is accomplished through the use of various equations that represent different TFs. Specifically, Eqs. (8), (10), (12), (14), and (16) correspond to the functions

Algorithm 6 Crossover operator

if $fit(\acute{y}_{t+1}^{i,j}) < fit(y_t^{i,j})$ **then**
  $y_{t+1}^{i,j} = \acute{y}_{t+1}^{i,j}$
else
  $[child1, child2] = crossover(\acute{y}_{t+1}^{i,j}, y_t^{i,j})$
  if $fit(child1) < fit(child2)$ **then**
    $y_{t+1}^{i,j} = child1$
  else
    $y_{t+1}^{i,j} = child2$
  end if
end if

T_s, T_v, T_u, T_{y1}, and T_{y2}, respectively. Each of these equations defines a different method for transforming the agents' velocities into probabilities, reflecting the distinct characteristics of the S-shaped, V-shaped, U-shaped, and X-shaped TFs.

Following this mapping, the newly obtained probability values are then utilized to update the positions of the search agents. This is achieved using the corresponding update equations for each TF: (9), (11), (13), (15), and (17). These equations determine the new positions for each agent, effectively converting the continuous search space of the RIME algorithm into a binary format suitable for FS tasks.

In the results section of the study, a comprehensive comparison is made between the different TFs as applied to the basic RIME and the proposed Binary mRIME. The goal of this comparison is to identify the best FS algorithm, considering the effectiveness of each TF in the conversion process. This step is crucial as the performance of the classifier used for FS problems significantly depends on how well the continuous search space is transformed into a binary one. The appropriate selection of a TF, therefore, has a direct impact on the accuracy and efficiency of the FS algorithm.

Objective function of the proposed mRIME

In FS methods, particularly those utilizing a wrapper-based approach, it's essential to incorporate a learning algorithm to evaluate the efficacy of the selected feature subset. In this study, the

k-Nearest Neighbor (k-NN) classifier, as referenced in the work by Keller et al. [53], is employed for this purpose. The k-NN classifier provides a measure of classification accuracy for the solutions generated by the FS process.

When designing an FS method, two critical aspects must be considered:

1. How to formulate the solution for the FS problem.
2. How to assess the quality of this solution.

In this study, the feature subset is represented as a binary vector, with its length equal to the number of attributes in the dataset. This representation allows for a straightforward interpretation of which features are selected (denoted by 1) and which are not (denoted by 0).

FS is inherently a multi-objective optimization problem, striving to achieve two main goals: (i) reducing the number of selected attributes, and (ii) improving the classification accuracy as determined by the k-NN classifier. There is a natural trade-off between these

goals: generally, decreasing the number of attributes enhances the model's simplicity and potentially its generalizability, while maintaining or improving classification accuracy ensures the utility and effectiveness of the selected features.

The best solution in this context is one that balances these two objectives effectively—it should have the fewest number of attributes while achieving the highest possible classification accuracy. To address this multi-objective nature, the FS methods need to integrate these contradictory goals into a single objective function. In this study, this integration is accomplished by formulating a fitness criterion for each solution. This fitness criterion is calculated using the k-NN classifier, as shown in Eq. (19). It quantifies how well a given solution balances the trade-off between attribute reduction and classification accuracy, guiding the iterative process of the mRIME toward finding the most effective feature subset.

$$fitness = \alpha\zeta_k + \beta\frac{|R|}{|N|} \tag{19}$$

In the context of the objective function described in Eq. (19) for FS, several key components and parameters are involved:

- ζ_k: This represents the classification rate achieved by the k-NN classifier. It is a measure of how accurately the classifier can predict the class labels of the dataset using the selected features. A higher classification rate indicates better predictive performance.
- $|R|$ and $|N|$: These symbols denote the number of selected attributes in the solution vector ($|R|$) and the total number of native attributes in the dataset ($|N|$), respectively. The objective is to reduce $|R|$ while maintaining or improving the classification rate ζ_k.
- α and β: These are balancing parameters, both ranging from 0 to 1. They are used to assign relative weights to the two aspects of the objective function: the classification rate (α) and the selection ratio of the selected attributes (β). The parameter α specifically weights the importance of the classification rate in the objective function, while β, being the complement of α (i.e., $\beta = 1-\alpha$), weights the importance of minimizing the number of selected attributes.

The objective function thus formulated effectively balances the dual goals of maximizing the classification rate (a measure of the effectiveness of the selected features in predicting outcomes) and minimizing the number of features used (a measure of model simplicity and efficiency). The values of α and β can be adjusted depending on the specific requirements of the FS task, allowing for flexibility in prioritizing either classification accuracy or model simplicity. This balance is crucial in creating an FS model that is not only accurate but also efficient and interpretable.

In the realm of classification tasks, a variety of classifiers are available, each with its strengths and applications. While the k-NN classifier is chosen in this study for its simplicity and effectiveness, especially in pattern recognition and FS contexts, other significant classifiers are also widely used in different scenarios. These include: Artificial Neural Networks (ANN) as discussed by Bishop [54], are powerful tools for pattern recognition and have the ability to learn complex nonlinear input–output relationships. Bayesian Classifier, as explained by Russell and Norvig [55], is effective in probabilistic classification and is known for handling uncertainty and incomplete data well. Support Vector

Machines (SVM) as explored by Ding et al. [56], are renowned for their effectiveness in high-dimensional spaces and are widely used in applications such as image classification and bioinformatics. Decision trees, as described by Rokach and Maimon [57], are simple to understand and interpret, making them popular for tasks where the explanation of the model's decision is important.

Despite their strengths, classifiers can struggle in situations where patterns from different classes are closely clustered or overlap under complex conditions. The k-NN classifier, as studied by Denoeux [58] and others, is selected in this study due to its non-parametric nature and straightforwardness, making it one of the easiest machine learning classification methods, as highlighted by Qin et al. [59]. Its effectiveness in pattern recognition and data mining has been recognized in various fields, as noted by Shakhnarovich et al. [60], and it often outperforms more advanced classifiers in practice [61].

In the specific context of FS, the k-NN classifier's ability to provide a clear and direct evaluation of the quality of a feature subset makes it a popular choice. This is evidenced in the works of Braik et al. [62] and Khurma et al. [63], among others. The selection of a base classifier for FS tasks largely depends on the specific requirements and characteristics of the application at hand.

The objective function detailed in Eq. (19) plays a crucial role in evaluating the selected feature subsets in FS tasks. This objective function is designed to create a balance between the number of features selected in the solution vector. It operates under the principle that an effective feature subset should not only be compact (i.e., contain a smaller number of features) but also maintain or enhance the classification accuracy of the model. However, it's important to note that there's an inherent dichotomy like this problem. On one hand, the objective function is part of a minimization problem where the goal is to reduce the number of features in the solution vector. Fewer features can lead to a more streamlined and potentially more interpretable model, which is less likely to overfit the training data. This reduction is particularly important in contexts where computational efficiency or model simplicity is valued. On the other hand, the aspect of classification accuracy, which is crucial for the effectiveness of the model, represents a maximization problem. Higher classification accuracy implies that the model is better at making correct predictions, which is the primary goal of most classification tasks. The challenge, therefore, is to maximize classification accuracy while simultaneously minimizing the number of features used.

The objective function in Eq. (19) seeks to address this challenge by integrating both aspects—the minimization of feature count and the maximization of classification accuracy— into a single evaluative criterion. This integrated approach ensures that the selected feature subset is not only compact but also effective in terms of classification performance, striking a balance that is vital for the development of robust and efficient predictive models.

ExpeRIMEntal results

The RIME optimization algorithm's study claims are generally focused on how well it works, how efficient it is, and how many different optimization problems it can solve. The RIME optimization algorithm makes the following frequent claims, which may be supported by experimental evaluations. We can offer proof of the efficiency and suit-

ability of optimization algorithms for resolving practical issues by carrying out thorough experimental assessments that refute these assertions.

- Claim: mRIME achieves higher solution quality than current approaches. ExpeRIMEntal evaluation: Conduct comparative experiments where the mRIME is compared against state-of-the-art methods using benchmark instances. Metrics such as solution quality, convergence rate, and computational time can be used to assess performance.
- Claim: mRIME is more resilient to changes in the types of problems that arise. Experimental evaluation: Test the algorithm on a variety of problem instances with different characteristics (e.g., size, structure, complexity). Measure the stability of the algorithm's performance across these instances and compare it with other methods.
- Claim: mRIME can tackle complex optimization issues and is scalable. Experimental evaluation: Measure the algorithm's scalability in terms of computational time, and solution quality as problem size rises. Assess the algorithm's performance on progressively larger problem instances.
- Claim: "Our algorithm is efficient and converges to near-optimal solutions quickly. Experimental evaluation: Analyze the algorithm's convergence behavior by monitoring the evolution of the solution quality or objective function values over time. On benchmark examples, compare the convergence speed with alternative techniques.
- Claim: mRIME can be used to solve optimization issues in the real world. Experimental evaluation: Test the algorithm using real-world datasets or examples of issues that are pertinent to particular fields such as engineering, and FS problems. Examine its functionality and usefulness in these real-world situations.
- Claim: Better trade-offs between processing resources and solution quality are made by mRIME. Experimental evaluation: Test the trade-offs between computing resources, such as runtime, and solution quality for the suggested mRIME in comparison to other methods. This can entail examining the answers obtained by running the algorithm with different computational budgets.
- Claim: mRIME is flexible and may be tailored to many variants of problem domains. Experimental evaluation: Test the algorithm with a range of problem samples from different domains, with varying constraints or problem formats. Examine how consistently and adaptably it performs in these different environments.

Evaluation metrics

In the proposed algorithm's evaluation, six standard metrics were employed: accuracy, specificity, sensitivity, fitness value, running time, number of selected features, and convergence curves. With these measurements, we were able to gauge its efficiency and effectiveness for continuous grid optimization (global optimization) and binary optimization (FS) in the context of disease diagnosis applications.

- TP: Correct classification as a disease.
- TN: Correct classification as a non-disease.
- FP: misclassification as a disease.
- FN:misclassification as a non-disease.

Accuracy metric alone may not suffice when evaluating a classification algorithm's performance when dealing with imbalanced datasets; in such cases, additional metrics like Specificity, Sensitivity, and F-measure, and Convergence Curves may provide a more comprehensive evaluation of its efficacy.

- Accuracy in machine learning is often used as the standard metric to assess classification algorithms' success, serving as an indication of the overall correctness of prediction made by these programs, such as representing the percentage of instances correctly classified by an algorithm. To calculate accuracy simply needs to divide the number of correctly classified instances by the total count in their dataset. Accuracy can be represented mathematically as in Eq. (20):

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{20}$$

- Sensitivity, referred to as True Positive Rate or Recall. It calculates the number of correctly classified attack files out of an entire dataset as in Eq. (21):

$$Sensitivity = \frac{TP}{TP + FN} \tag{21}$$

- Specificity, referred to as True Negative Rate. It measures the percentage of actual negatives identified correctly by the classifier as in Eq. (22):

$$Specificity = \frac{TN}{TN + FP} \tag{22}$$

- F-Measure is an often-utilized metric for gauging the overall performance of classification algorithms when the cost of false positives and false negatives differ significantly. A higher F-measure indicates better performance with 1 being representative of perfect precision and sensitivity. It measures precision and recall by taking their harmonic mean values of precision and sensitivity, representing it mathematically using Eq. (23):

$$F - measure = \frac{2 * precision * recall}{precision + recall} \tag{23}$$

- Fitness Values are used to evaluate the quality of solutions produced by optimization algorithms, particularly feature selection techniques. They serve as a gauge to indicate which solutions produce quality solutions; in terms of classification accuracy vs number of selected features this value represents a trade-off between classification accuracy and feature selection frequency, higher values indicate better solutions.
- Running time is an invaluable metric of an algorithm's computational efficiency, measured in seconds to represent its total execution period which includes FS and classification tasks.
- Number of selected features serves as an important evaluation metric for FS algorithms, reflecting their ability to reduce dimensionality thereby having a direct bearing on model complexity, interpretability, and generalization performance.
- Convergence curves provide insight into an algorithm's optimization process and convergence behavior, representing its optimization iterations or generation number on one axis while fitness value on another (y-axis) remains constant over time (the x-axis represents several iterations or generation), helping analyze speed, stability, and ability to escape local optima on another (x-axis represents iterations or genera-

Table 1 Setting up the parameters for mRIME and other search algorithms

Algorithm	Parameters elaboration	Population size
mRIME	$r_1 \in (-1, 1)$, β and E are adaptive parameters, $w = 5$, $h \in (0, 1)$	100
WSO	$f_{min} = 0.07, f_{max} = 0.75$ $\tau = 4.125, a_0 = 6.25, a_1 = 100, a_2 = 0.0005$	100
TLBO	No unique parameters	50 (2 phases in search process)
SFS	The upper limit for diffusion is set to 1	50 (2 phases in search process)
DE	F = 0.5, CR = 0.9	100
GA	Roulette wheel picking tactics, Probability of mutation = 0.01, one-point crossover with a probability of one.	100
GSK	P = 0.1, k_f = 0.5, k_r = 0.9, K = 10	100
AMO	Each set of animals has exactly 5 animals	50 (2 phases in search process)
PSO	$\omega = 0.6$, $c_1 = 2$ and $c_2 = 2$	100
BBO	Habitat modification probability = 1, immigration probability bounds for each gene = [0, 1], mutation probability = 0, maximum migration and immigration rates for each island = 1, and the step size for numerical integration of probabilities = 1.	100
ACO	Pheromone update constant = 20, the pheromone's starting value = 1E−06, exploration constant = 1, rates of regional and global pheromone degradation = 0.5 and 0.9, respectively, and levels of visibility and pheromone sensitivity = 5 and 1, respectively.	100

tion count on another axis and fitness value on third). The convergence curve helps evaluate speed stability and ability to escape local optima).

Controlling parameter setup

The suggested mRIME's findings have been evaluated with 10 of the most reputable optimization algorithms in the appropriate field of research when examined on the aforementioned test suites to support its overall performance and thorough evaluation. The competing algorithms to the one that is being provided can be classified into three groups: (i) GA [19], DE [17], BBO [18] and SFS [16] as the most studied EAs, (ii) PSO [20], ACO [21] and AMO [22] as hot and reliable SI algorithms, WSO [25] and (iii) TLBO [23] and GSK [24] as efficacious and recent human-based optimizers. The control parameters and settings of RIME and other competing algorithms are shown in Table 1.

The parameter values for the competing algorithms listed in Table 1 are those listed in [24]," which were obtained straight from those sources' native references. mRIME's initialization procedure is comparable to that of other rival algorithms. This will allow mRIME and other rival algorithms to be fairly compared. The accuracy and stability of the algorithms were evaluated using the average (Ave) and standard deviation (Std) score metrics. These statistical evaluation metrics were calculated in this work for each method and each function as the top two options. The algorithms' correctness was assessed using the mean measure, and their stability was intended to be ensured by analyzing the findings of the standard deviation study.

ExpeRIMEnts series1: global optimization using CEC 2017 test suit functions

The CEC2017 [64] consists of several functions that each stand in for a different set of optimization issues. Variability, complexity, and dynamism are the hallmarks of these functions. For the evaluation of suggested optimization algorithms, they are frequently employed. They are used to the suggested mRIME in this study to assess it. They can also explain how the optimization algorithm explores and exploits data.

The Thirty functions are listed in Table 31, and they are divided into four sets as follows: The Unimodal set spans from F1 to F3. The Multimodal set ranges from F4 through F10. The Hybrid set spans from F11 to F20, and the Composition set spans from F21 to F30. F2 is not applied during evaluation. Thus, the suggested mBWO and other algorithms are evaluated using 29 functions. Additionally, Table 31 demonstrates that the dimension is equivalent to 30 and the search scope for all test functions is from −100 to 100.

Experiments series1: Applying mRIME for global optimization using CEC2017 test suit functions

Table 2 presents the results of a comparative study between two algorithms called mRIME and RIME. The study aimed to evaluate the statistical performance of these algorithms on 51 different runs using the CEC2017 functions. Each run used 10 variables and performed 100,000 function evaluations.

Analyzing the results presented in the table, we can observe the following patterns and insights:

- Best value: The mRIME algorithm outperforms the original RIME algorithm in terms of the best objective function values for most of the functions. In general, mRIME achieves lower best values, indicating superior performance in finding optimal solutions.
- Median value: For some functions, mRIME obtains lower median values compared to RIME, indicating better performance in terms of the middle-range objective function values. However, for a few functions like CEC17(F4), CEC17(F15), and CEC17(F18), RIME achieves lower median values, suggesting better performance in those cases.
- Average value: The average objective function values achieved by mRIME are generally lower than those obtained by RIME. This indicates that mRIME performs better on average across the tested functions, as it provides solutions closer to the optimal values.
- Worst value: In terms of the worst objective function values, mRIME consistently outperforms RIME for all the functions. The worst values achieved by mRIME are lower, indicating that it avoids poor solutions more effectively.
- Std: The standard deviation values provide insights into the spread or variability of the objective function values obtained by each algorithm. In most cases, mRIME has lower standard deviation values, indicating that its solutions are more consistent and less variable compared to RIME.

Based on the analysis of these metrics, it can be concluded that mRIME generally exhibits superior performance compared to the original RIME algorithm. It achieves lower best and average objective function values, avoids poor solutions (lower worst values), and provides more consistent results (lower standard deviation). However, there are a few

Table 2 Results of 51 separate runs comparing mRIME to the original RIME and other optimization methods on the CEC2017 test functions of 10 variables with 100,000 function evaluations

Function		mRIME	RIME	WSO	TLBO	SFS	DE	GA	GSK	AMO	PSO	BBO	ACO
CEC17(F1)	Ave	1.50×10^{3}	3.11×10^{0}	$\mathbf{0.00 \times 10^{0}}$	1.96×10^{3}	5.60×10^{3}	$\mathbf{0.00 \times 10^{0}}$	1.14×10^{6}	$\mathbf{0.00 \times 10^{0}}$	9.84×10^{0}	2.30×10^{3}	1.12×10^{6}	1.57×10^{10}
	Std	1.52×10^{3}	9.63×10^{0}	$\mathbf{0.00 \times 10^{0}}$	2.52×10^{3}	3.26×10^{3}	$\mathbf{0.00 \times 10^{0}}$	2.18×10^{5}	$\mathbf{0.00 \times 10^{0}}$	2.69×10^{1}	3.05×10^{3}	2.26×10^{5}	3.75×10^{9}
CEC17(F3)	Ave	7.61×10^{-9}	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	2.05×10^{-2}	$\mathbf{0.00 \times 10^{0}}$	1.24×10^{4}	$\mathbf{0.00 \times 10^{0}}$	2.90×10^{-8}	$\mathbf{0.00 \times 10^{0}}$	8.35×10^{2}	6.32×10^{3}
	Std	3.49×10^{-8}	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	1.17×10^{-2}	$\mathbf{0.00 \times 10^{0}}$	7.87×10^{3}	$\mathbf{0.00 \times 10^{0}}$	9.16×10^{-8}	$\mathbf{0.00 \times 10^{0}}$	8.07×10^{2}	1.74×10^{3}
CEC17(F4)	Ave	2.28×10^{0}	2.65×10^{0}	$\mathbf{0.00 \times 10^{0}}$	1.79×10^{-1}	9.37×10^{-1}	$\mathbf{0.00 \times 10^{0}}$	1.17×10^{1}	$\mathbf{0.00 \times 10^{0}}$	1.39×10^{0}	2.82×10^{0}	$6.20 \times^{0}$	1.83×10^{2}
	Std	1.46×10^{0}	1.21×10^{1}	$\mathbf{0.00 \times 10^{0}}$	3.45×10^{-1}	6.74×10^{-1}	$\mathbf{0.00 \times 10^{0}}$	3.29×10^{0}	$\mathbf{0.00 \times 10^{0}}$	6.29×10^{-1}	1.21×10^{0}	1.72×10^{0}	4.26×10^{1}
CEC17(F5)	Ave	7.15×10^{0}	1.73×10^{1}	8.52×10^{0}	8.24×10^{0}	8.68×10^{0}	2.48×10^{1}	4.90×10^{1}	2.03×10^{1}	$\mathbf{6.36 \times 10^{0}}$	1.59×10^{1}	7.96×10^{0}	6.86×10^{1}
	Std	1.96×10^{0}	7.54×10^{0}	3.21×10^{0}	3.38×10^{0}	3.30×10^{0}	4.13×10^{0}	1.22×10^{1}	2.79×10^{0}	$\mathbf{1.49 \times 10^{0}}$	7.08×10^{0}	2.77×10^{0}	8.53×10^{0}
CEC17(F6)	Ave	1.08×10^{-3}	1.20×10^{-1}	1.40×10^{-7}	3.76×10^{-2}	5.35×10^{-3}	$\mathbf{0.00 \times 10^{0}}$	1.24×10^{1}	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	8.27×10^{-2}	9.19×10^{-1}	3.57×10^{1}
	Std	1.35×10^{-3}	5.51×10^{-1}	5.67×10^{-7}	1.28×10^{-1}	1.59×10^{-3}	$\mathbf{0.00 \times 10^{0}}$	4.54×10^{0}	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	3.36×10^{-1}	$\mathbf{0.00 \times 10^{0}}$	5.40×10^{0}
CEC17(F7)	Ave	1.82×10^{1}	2.34×10^{1}	1.79×10^{1}	1.76×10^{1}	2.33×10^{1}	3.43×10^{1}	6.87×10^{1}	3.07×10^{1}	1.82×10^{1}	$\mathbf{1.72 \times 10^{1}}$	2.31×10^{1}	1.23×10^{2}
	Std	3.37×10^{0}	7.85×10^{0}	4.60×10^{0}	3.25×10^{0}	3.60×10^{0}	4.84×10^{0}	1.46×10^{1}	3.08×10^{0}	$\mathbf{1.42 \times 10^{0}}$	4.46×10^{0}	3.82×10^{0}	1.43×10^{1}
CEC17(F8)	Ave	7.34×10^{0}	1.59×10^{1}	8.72×10^{0}	$\mathbf{6.84 \times 10^{0}}$	7.42×10^{0}	2.37×10^{1}	3.54×10^{1}	2.02×10^{1}	$\mathbf{6.84 \times 10^{0}}$	1.23×10^{1}	8.66×10^{0}	4.85×10^{1}
	Std	2.14×10^{0}	9.14×10^{0}	4.92×10^{0}	2.75×10^{0}	2.76×10^{0}	3.76×10^{0}	8.94×10^{0}	2.92×10^{0}	$\mathbf{1.49 \times 10^{0}}$	5.33×10^{0}	3.05×10^{0}	6.66×10^{0}
CEC17(F9)	Ave	$\mathbf{0.00 \times 10^{0}}$	4.26×10^{-3}	1.76×10^{-2}	2.64×10^{-1}	6.68×10^{-6}	$\mathbf{0.00 \times 10^{0}}$	2.39×10^{1}	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	3.15×10^{-1}	5.56×10^{2}
	Std	$\mathbf{0.00 \times 10^{0}}$	1.95×10^{-2}	7.02×10^{-2}	4.22×10^{-1}	4.34×10^{6}	$\mathbf{0.00 \times 10^{0}}$	1.75×10^{1}	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	8.16×10^{-2}	1.53×10^{2}
CEC17(F10)	Ave	$\mathbf{1.54 \times 10^{2}}$	5.22×10^{2}	5.44×10^{2}	4.95×10^{2}	3.63×10^{2}	4.94×10^{2}	7.52×10^{2}	1.06×10^{3}	3.65×10^{2}	6.30×10^{2}	2.57×10^{2}	1.45×10^{3}
	Std	1.04×10^{2}	1.96×10^{2}	2.84×10^{2}	2.92×10^{2}	1.91×10^{2}	2.99×10^{2}	2.36×10^{2}	1.33×10^{2}	$\mathbf{1.01 \times 10^{2}}$	2.64×10^{2}	1.55×10^{2}	1.34×10^{2}
CEC17(F11)	Ave	6.08×10^{0}	8.79×10^{0}	4.19×10^{0}	7.06×10^{0}	4.61×10^{0}	4.33×10^{-2}	8.21×10^{1}	$\mathbf{0.00 \times 10^{0}}$	2.76×10^{0}	1.10×10^{1}	8.92×10^{0}	1.96×10^{2}
	Std	4.01×10^{0}	4.43×10^{0}	3.49×10^{0}	5.17×10^{0}	1.25×10^{0}	1.95×10^{-1}	7.89×10^{1}	$\mathbf{0.00 \times 10^{0}}$	7.90×10^{-1}	7.27×10^{0}	4.26×10^{0}	6.99×10^{1}
CEC17(F12)	Ave	1.01×10^{4}	1.30×10^{3}	3.66×10^{2}	1.28×10^{4}	5.04×10^{3}	$\mathbf{8.06 \times 10^{0}}$	1.12×10^{6}	8.93×10^{1}	1.13×10^{4}	1.33×10^{4}	2.55×10^{5}	9.79×10^{8}
	Std	5.68×10^{3}	1.89×10^{3}	1.77×10^{2}	9.07×10^{3}	2.11×10^{3}	$\mathbf{1.89 \times 10^{1}}$	1.78×10^{6}	7.26×10^{1}	6.49×10^{3}	1.24×10^{4}	1.60×10^{5}	9.01×10^{8}

Table 2 continued

Function		mRIME	RIME	WSO	TLBO	SFS	DE	GA	GSK	AMO	PSO	BBO	ACO
CEC17(F13)	Ave	2.07×10^{3}	9.07×10^{1}	2.34×10^{1}	1.70×10^{3}	4.55×10^{1}	$\mathbf{6.56 \times 10^{0}}$	2.62×10^{4}	$\mathbf{6.56 \times 10^{0}}$	2.67×10^{1}	6.45×10^{3}	8.09×10^{4}	7.02×10^{6}
	Std	2.21×10^{3}	9.22×10^{1}	1.93×10^{1}	1.91×10^{3}	9.83×10^{0}	1.86×10^{0}	2.19×10^{4}	$\mathbf{1.41 \times 10^{0}}$	8.99×10^{0}	5.72×10^{3}	7.12×10^{4}	2.36×10^{7}
CEC17(F14)	Ave	8.19×10^{1}	4.43×10^{1}	2.15×10^{1}	3.40×10^{1}	2.20×10^{1}	$\mathbf{4.30 \times 10^{-2}}$	1.48×10^{4}	5.88×10^{0}	3.83×10^{0}	4.57×10^{1}	1.20×10^{4}	4.33×10^{2}
	Std	3.91×10^{1}	1.90×10^{1}	1.28×10^{1}	7.43×10^{0}	3.82×10^{0}	$\mathbf{1.96 \times 10^{-1}}$	6.43×10^{3}	3.06×10^{0}	1.67×10^{0}	1.93×10^{1}	1.01×10^{4}	1.10×10^{2}
CEC17(F15)	Ave	1.84×10^{2}	3.38×10^{1}	6.57×10^{0}	4.75×10^{1}	1.00×10^{1}	$\mathbf{3.60 \times 10^{-2}}$	1.38×10^{4}	2.22×10^{-1}	1.75×10^{0}	5.62×10^{1}	1.95×10^{4}	3.74×10^{3}
	Std	1.85×10^{2}	4.03×10^{1}	5.39×10^{0}	2.09×10^{1}	2.14×10^{0}	$\mathbf{1.03 \times 10^{-1}}$	1.26×10^{4}	2.13×10^{-1}	5.40×10^{-1}	5.89×10^{1}	1.94×10^{4}	2.37×10^{3}
CEC17(F16)	Ave	3.91×10^{1}	1.36×10^{1}	1.27×10^{1}	1.44×10^{1}	4.17×10^{0}	2.93×10^{0}	3.00×10^{1}	4.27×10^{0}	$\mathbf{1.13 \times 10^{0}}$	2.05×10^{2}	1.06×10^{2}	2.65×10^{2}
	Std	5.29×10^{1}	1.53×10^{1}	3.57×10^{1}	3.48×10^{1}	3.16×10^{0}	4.65×10^{0}	4.18×10^{1}	4.95×10^{0}	$\mathbf{2.64 \times 10^{-1}}$	1.19×10^{2}	8.75×10^{1}	5.71×10^{1}
CEC17(F17)	Ave	2.12×10^{1}	3.04×10^{1}	2.52×10^{1}	2.76×10^{1}	2.34×10^{1}	$\mathbf{4.82 \times 10^{0}}$	4.12×10^{1}	1.17×10^{1}	4.92×10^{0}	4.56×10^{1}	2.74×10^{1}	1.14×10^{2}
	Std	1.86×10^{1}	1.21×10^{1}	1.25×10^{1}	8.83×10^{0}	5.66×10^{0}	6.77×10^{0}	1.29×10^{1}	7.09×10^{0}	$\mathbf{2.53 \times 10^{0}}$	2.30×10^{1}	3.49×10^{1}	2.31×10^{1}
CEC17(F18)	Ave	2.45×10^{3}	5.90×10^{1}	2.15×10^{1}	4.55×10^{3}	5.22×10^{1}	$\mathbf{8.18 \times 10^{-2}}$	8.30×10^{4}	3.20×10^{-1}	3.13×10^{1}	5.10×10^{3}	1.45×10^{5}	6.67×10^{7}
	Std	1.95×10^{3}	3.31×10^{1}	1.12×10^{1}	3.67×10^{3}	1.05×10^{1}	$\mathbf{1.73 \times 10^{-1}}$	9.59×10^{4}	1.92×10^{-1}	8.02×10^{0}	5.76×10^{3}	1.01×10^{5}	1.65×10^{8}
CEC17(F19)	Ave	5.54×10^{1}	1.64×10^{1}	2.73×10^{0}	2.94×10^{1}	5.84×10^{0}	$\mathbf{6.27 \times 10^{-3}}$	6.30×10^{4}	1.55×10^{-1}	1.11×10^{0}	9.57×10^{1}	4.34×10^{4}	1.24×10^{4}
	Std	5.55×10^{1}	2.14×10^{1}	2.24×10^{0}	1.83×10^{1}	8.94×10^{-1}	$\mathbf{1.11 \times 10^{-2}}$	4.21×10^{4}	3.51×10^{-1}	4.28×10^{-1}	2.95×10^{2}	4.02×10^{4}	1.31×10^{4}
CEC17(F20)	Ave	6.68×10^{0}	1.98×10^{1}	2.43×10^{1}	1.75×10^{1}	1.21×10^{1}	1.10×10^{-1}	1.02×10^{2}	1.19×10^{0}	$\mathbf{6.19 \times 10^{-4}}$	5.51×10^{1}	1.29×10^{1}	1.20×10^{2}
	Std	9.26×10^{0}	9.32×10^{0}	1.13×10^{1}	1.04×10^{1}	3.31×10^{0}	1.63×10^{-1}	5.01×10^{1}	3.99×10^{0}	$\mathbf{1.74 \times 10^{-3}}$	5.42×10^{1}	5.91×10^{0}	2.77×10^{1}
CEC17(F21)	Ave	2.10×10^{2}	1.01×10^{2}	1.68×10^{2}	1.62×10^{2}	$\mathbf{1.00 \times 10^{2}}$	1.80×10^{2}	1.54×10^{2}	1.93×10^{2}	1.37×10^{2}	1.79×10^{2}	2.00×10^{2}	1.37×10^{2}
	Std	2.52×10^{0}	1.28×10^{0}	5.31×10^{1}	5.26×10^{1}	$\mathbf{5.06 \times 10^{-2}}$	6.22×10^{1}	1.49×10^{1}	5.07×10^{1}	5.02×10^{1}	5.65×10^{1}	3.07×10^{1}	1.29×10^{1}
CEC17(F22)	Ave	1.02×10^{2}	8.49×10^{1}	9.40×10^{1}	9.98×10^{1}	$\mathbf{9.24 \times 10^{1}}$	9.48×10^{1}	1.25×10^{2}	1.00×10^{2}	9.91×10^{1}	9.38×10^{1}	1.06×10^{2}	2.57×10^{2}
	Std	9.58×10^{-1}	3.19×10^{1}	2.47×10^{1}	1.07×10^{1}	3.00×10^{1}	2.30×10^{1}	2.17×10^{1}	$\mathbf{4.87 \times 10^{-1}}$	7.18×10^{0}	2.55×10^{1}	7.47×10^{0}	6.14×10^{1}
CEC17(F23)	Ave	3.10×10^{2}	3.15×10^{2}	3.11×10^{2}	3.10×10^{2}	$\mathbf{3.03 \times 10^{2}}$	3.17×10^{2}	3.63×10^{2}	3.18×10^{2}	3.08×10^{2}	3.28×10^{2}	3.13×10^{2}	3.78×10^{2}
	Std	3.63×10^{0}	6.97×10^{0}	5.34×10^{0}	4.31×10^{0}	4.35×10^{1}	5.20×10^{0}	1.51×10^{1}	3.99×10^{0}	$\mathbf{1.61 \times 10^{0}}$	1.24×10^{1}	4.97×10^{0}	9.61×10^{0}

Table 2 continued

Function		mRIME	RIME	WSO	TLBO	SFS	DE	GA	GSK	AMO	PSO	BBO	ACO
CEC17(F24)	Ave	3.38×10^{2}	1.33×10^{2}	2.88×10^{2}	3.23×10^{2}	$\mathbf{2.18 \times 10^{2}}$	3.43×10^{2}	3.90×10^{2}	3.44×10^{2}	2.85×10^{2}	3.24×10^{2}	3.36×10^{2}	2.93×10^{2}
	Std	2.88×10^{0}	7.64×10^{1}	9.43×10^{1}	5.62×10^{1}	1.18×10^{2}	4.97×10^{1}	3.40×10^{1}	$\mathbf{1.87 \times 10^{1}}$	8.09×10^{1}	8.33×10^{1}	2.66×10^{1}	3.63×10^{1}
CEC17(F25)	Ave	4.27×10^{2}	4.07×10^{2}	$\mathbf{4.10 \times 10^{2}}$	4.26×10^{2}	4.21×10^{2}	4.11×10^{2}	4.30×10^{2}	4.27×10^{2}	4.18×10^{2}	4.25×10^{2}	4.34×10^{2}	5.76×10^{2}
	Std	2.30×10^{1}	7.41×10^{1}	2.13×10^{1}	2.23×10^{1}	2.30×10^{1}	2.10×10^{1}	2.52×10^{1}	$\mathbf{2.05 \times 10^{1}}$	2.23×10^{1}	2.29×10^{1}	2.26×10^{1}	4.23×10^{1}
CEC17(F26)	Ave	3.80×10^{2}	3.12×10^{2}	2.94×10^{2}	3.31×10^{2}	2.92×10^{2}	3.00×10^{2}	4.19×10^{2}	3.00×10^{2}	3.00×10^{2}	$\mathbf{2.74 \times 10^{2}}$	3.40×10^{2}	7.47×10^{2}
	Std	1.33×10^{2}	3.99×10^{1}	2.37×10^{1}	5.60×10^{1}	4.40×10^{1}	$\mathbf{0.00 \times 10^{0}}$	8.88×10^{1}	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	7.63×10^{1}	1.51×10^{2}	8.18×10^{1}
CEC17(F27)	Ave	3.99×10^{2}	3.94×10^{2}	3.95×10^{2}	3.93×10^{2}	3.92×10^{2}	3.90×10^{2}	4.10×10^{2}	$\mathbf{3.89 \times 10^{2}}$	3.91×10^{2}	4.03×10^{2}	3.97×10^{2}	4.40×10^{2}
	Std	3.21×10^{0}	9.33×10^{0}	3.40×10^{0}	2.62×10^{0}	1.85×10^{0}	2.63×10^{-1}	9.10×10^{0}	$\mathbf{2.17 \times 10^{-1}}$	2.38×10^{0}	1.97×10^{1}	4.39×10^{0}	8.57×10^{0}
CEC17(F28)	Ave	5.99×10^{2}	3.45×10^{2}	3.97×10^{2}	4.01×10^{2}	3.06×10^{2}	3.63×10^{2}	5.82×10^{2}	3.12×10^{2}	$\mathbf{2.99 \times 10^{2}}$	4.54×10^{2}	5.48×10^{2}	5.85×10^{2}
	Std	1.40×10^{1}	1.36×10^{2}	1.49×10^{2}	1.26×10^{2}	3.97×10^{1}	1.21×10^{2}	1.56×10^{2}	3.20×10^{1}	$\mathbf{3.99 \times 10^{0}}$	1.57×10^{2}	9.68×10^{1}	4.64×10^{1}
CEC17(F29)	Ave	2.55×10^{2}	2.76×10^{2}	2.52×10^{2}	2.63×10^{2}	2.59×10^{2}	$\mathbf{2.40 \times 10^{2}}$	3.04×10^{2}	2.48×10^{2}	2.64×10^{2}	3.05×10^{2}	2.65×10^{2}	3.90×10^{2}
	Std	1.43×10^{1}	3.06×10^{1}	1.38×10^{1}	1.46×10^{1}	1.18×10^{1}	$\mathbf{4.73 \times 10^{0}}$	4.09×10^{1}	4.76×10^{0}	8.17×10^{0}	4.50×10^{1}	1.56×10^{1}	3.41×10^{1}
CEC17(F30)	Ave	3.65×10^{3}	3.97×10^{4}	4.58×10^{2}	1.25×10^{5}	2.03×10^{3}	1.64×10^{4}	3.73×10^{5}	$\mathbf{4.56 \times 10^{2}}$	7.42×10^{3}	2.00×10^{5}	4.63×10^{5}	2.23×10^{7}
	Std	1.51×10^{3}	1.78×10^{5}	1.10×10^{2}	2.98×10^{5}	1.63×10^{3}	1.14×10^{5}	3.78×10^{5}	3.44×10^{1}	5.17×10^{3}	3.79×10^{5}	5.39×10^{5}	2.32×10^{7}
Ranking (W|T|L)		1|1|27	0|1|28	1|3|25	0|2|27	4|0|25	7|6|16	0|0|29	3|6|20	4|2|23	2|2|25	0|0|29	0|0|29

functions where RIME performs better in terms of median values, indicating that the two algorithms have different strengths depending on the specific problem (Tables 3, 4).

Analyzing the results presented in Table 5, we can derive the following observations and insights:

- Best value: The mRIME algorithm performs better than the original RIME algorithm in terms of the best objective function values for most of the functions. mRIME achieves lower best values, indicating its superior capability to find optimal solutions.
- Median value: For some functions, mRIME obtains lower median values compared to RIME, indicating better performance in terms of the middle-range objective function values. However, for a few functions like CEC17(F3), CEC17(F10), CEC17(F11), and CEC17(F14), RIME achieves lower median values, suggesting better performance in those cases.
- Average value: The average objective function values obtained by mRIME are generally lower than those achieved by RIME. This indicates that mRIME performs better on average across the tested functions, as it provides solutions closer to the optimal values.
- Worst value: In terms of the worst objective function values, mRIME consistently outperforms RIME for most of the functions. The worst values achieved by mRIME are lower, indicating that it avoids poor solutions more effectively.
- Std: The standard deviation values provide insights into the spread or variability of the objective function values obtained by each algorithm. In most cases, mRIME has lower standard deviation values, indicating that its solutions are more consistent and less variable compared to RIME.

Based on the analysis of these metrics, it can be concluded that mRIME generally exhibits superior performance compared to the original RIME algorithm. It achieves lower best and average objective function values, avoids poor solutions (lower worst values), and provides more consistent results (lower standard deviation). However, there are a few functions where RIME performs better in terms of median values, indicating that the two algorithms have different strengths depending on the specific problem.

Analyzing the results presented in Table 6, the following observations and insights can be derived:

- Best value: The mRIME algorithm generally performs better than the original RIME algorithm in terms of the best objective function values for most of the functions. mRIME achieves lower best values, indicating its superior capability to find optimal solutions even with a higher number of variables.
- Median value: For some functions, mRIME obtains lower median values compared to RIME, indicating better performance in terms of the middle-range objective function values. However, there are also functions (e.g., CEC17(F1), CEC17(F3), CEC17(F15)) where RIME achieves lower median values, suggesting better performance in those cases.
- Average value: The average objective function values obtained by mRIME are generally lower than those achieved by RIME. This indicates that mRIME performs better on average across the tested functions, providing solutions closer to the optimal values even with a higher number of variables.

Table 3 Results of 51 separate runs using the CEC2017 functions of 10 variables and 100,000 function evaluations to compare the statistical performance of mRIME and original RIME

Function	Best value		Median value		Average value		Worst value		Std	
	RIME	mRIME	RIME	mRIME	RIME	mRIME	RIME	mRIME	RIME	mRIME
CEC17(F1)	$\mathbf{7.21 \times 10^{0}}$	6.45×10^{4}	8.68×10^{2}	$\mathbf{7.50 \times 10^{-2}}$	1.50×10^{3}	$\mathbf{3.11 \times 10^{0}}$	4.98×10^{3}	$\mathbf{4.39 \times 10^{1}}$	1.52×10^{3}	$\mathbf{9.63 \times 10^{0}}$
CEC17(F3)	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	7.61×10^{-9}	$\mathbf{0.00 \times 10^{0}}$	1.60×10^{-7}	$\mathbf{0.00 \times 10^{0}}$	3.49×10^{-8}	$\mathbf{0.00 \times 10^{0}}$
CEC17(F4)	2.22×10^{-2}	$\mathbf{0.00 \times 10^{0}}$	2.00×10^{0}	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{2.28 \times 10^{0}}$	2.65×10^{0}	$\mathbf{4.33 \times 10^{0}}$	5.56×10^{1}	$\mathbf{1.46 \times 10^{0}}$	1.21×10^{1}
CEC17(F5)	$\mathbf{3.98 \times 10^{0}}$	6.96×10^{0}	$\mathbf{6.96 \times 10^{0}}$	1.79×10^{1}	$\mathbf{7.15 \times 10^{0}}$	1.73×10^{1}	$\mathbf{1.09 \times 10^{1}}$	3.22×10^{1}	$\mathbf{1.96 \times 10^{0}}$	7.54×10^{0}
CEC17(F6)	1.43×10^{-6}	$\mathbf{0.00 \times 10^{0}}$	4.18×10^{-4}	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{1.08 \times 10^{-3}}$	1.20×10^{-1}	$\mathbf{3.92 \times 10^{-3}}$	2.52×10^{0}	$\mathbf{1.35 \times 10^{-3}}$	5.51×10^{-1}
CEC17(F7)	1.27×10^{1}	$\mathbf{1.05 \times 10^{1}}$	$\mathbf{1.75 \times 10^{1}}$	2.21×10^{1}	$\mathbf{1.82 \times 10^{1}}$	2.34×10^{1}	$\mathbf{2.61 \times 10^{1}}$	4.44×10^{1}	$\mathbf{3.37 \times 10^{0}}$	7.85×10^{0}
CEC17(F8)	2.98×10^{0}	$\mathbf{1.99 \times 10^{0}}$	$\mathbf{6.96 \times 10^{0}}$	1.49×10^{1}	$\mathbf{7.34 \times 10^{0}}$	1.59×10^{1}	$\mathbf{1.09 \times 10^{1}}$	3.48×10^{1}	$\mathbf{2.14 \times 10^{0}}$	9.14×10^{0}
CEC17(F9)	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	4.26×10^{-3}	$\mathbf{0.00 \times 10^{0}}$	8.95×10^{-2}	$\mathbf{0.00 \times 10^{0}}$	1.95×10^{-2}
CEC17(F10)	$\mathbf{3.79 \times 10^{0}}$	1.50×10^{2}	$\mathbf{1.47 \times 10^{2}}$	4.73×10^{2}	$\mathbf{1.54 \times 10^{2}}$	5.22×10^{2}	$\mathbf{3.47 \times 10^{2}}$	8.77×10^{2}	$\mathbf{1.04 \times 10^{2}}$	1.96×10^{2}
CEC17(F11)	$\mathbf{1.02 \times 10^{0}}$	1.99×10^{0}	$\mathbf{5.17 \times 10^{0}}$	7.96×10^{0}	$\mathbf{6.08 \times 10^{0}}$	8.79×10^{0}	$\mathbf{1.38 \times 10^{1}}$	1.89×10^{1}	$\mathbf{4.01 \times 10^{0}}$	4.43×10^{0}
CEC17(F12)	2.06×10^{3}	$\mathbf{2.39 \times 10^{2}}$	8.22×10^{3}	$\mathbf{7.36 \times 10^{2}}$	1.01×10^{4}	$\mathbf{1.30 \times 10^{3}}$	2.40×10^{4}	$\mathbf{9.06 \times 10^{3}}$	5.68×10^{3}	$\mathbf{1.89 \times 10^{3}}$
CEC17(F13)	2.23×10^{1}	$\mathbf{6.36 \times 10^{0}}$	9.06×10^{2}	$\mathbf{6.47 \times 10^{1}}$	2.07×10^{3}	$\mathbf{9.07 \times 10^{1}}$	6.98×10^{3}	$\mathbf{4.06 \times 10^{2}}$	2.21×10^{3}	$\mathbf{9.22 \times 10^{1}}$
CEC17(F14)	2.95×10^{1}	$\mathbf{1.80 \times 10^{1}}$	8.33×10^{1}	$\mathbf{3.89 \times 10^{1}}$	8.19×10^{1}	$\mathbf{4.43 \times 10^{1}}$	1.63×10^{2}	$\mathbf{9.96 \times 10^{1}}$	3.91×10^{1}	$\mathbf{1.90 \times 10^{1}}$
CEC17(F15)	3.21×10^{0}	$\mathbf{2.21 \times 10^{0}}$	1.23×10^{2}	$\mathbf{1.41 \times 10^{1}}$	1.84×10^{2}	$\mathbf{3.38 \times 10^{1}}$	6.80×10^{2}	$\mathbf{1.49 \times 10^{2}}$	1.85×10^{2}	$\mathbf{4.03 \times 10^{1}}$
CEC17(F16)	$\mathbf{2.44 \times 10^{-1}}$	1.10×10^{0}	$\mathbf{3.20 \times 10^{0}}$	6.40×10^{0}	3.91×10^{1}	$\mathbf{1.36 \times 10^{1}}$	1.33×10^{2}	$\mathbf{3.96 \times 10^{1}}$	5.29×10^{1}	$\mathbf{1.53 \times 10^{1}}$
CEC17(F17)	$\mathbf{1.31 \times 10^{0}}$	2.63×10^{0}	$\mathbf{2.21 \times 10^{1}}$	3.66×10^{1}	$\mathbf{2.12 \times 10^{1}}$	3.04×10^{1}	8.05×10^{1}	$\mathbf{4.54 \times 10^{1}}$	1.86×10^{1}	$\mathbf{1.21 \times 10^{1}}$

Table 3 continued

Function	Best value		Median value		Average value		Worst value		Std	
	RIME	mRIME	RIME	mRIME	RIME	mRIME	RIME	mRIME	RIME	mRIME
CEC17(F18)	1.51×10^{2}	$\mathbf{2.26 \times 10^{1}}$	2.00×10^{3}	$\mathbf{5.45 \times 10^{1}}$	2.45×10^{3}	$\mathbf{5.90 \times 10^{1}}$	6.44×10^{3}	$\mathbf{1.38 \times 10^{2}}$	1.95×10^{3}	$\mathbf{3.31 \times 10^{1}}$
CEC17(F19)	8.16×10^{0}	$\mathbf{1.02 \times 10^{0}}$	2.28×10^{1}	$\mathbf{1.00 \times 10^{1}}$	5.54×10^{1}	$\mathbf{1.64 \times 10^{1}}$	1.84×10^{2}	$\mathbf{1.03 \times 10^{2}}$	5.55×10^{1}	$\mathbf{2.14 \times 10^{1}}$
CEC17(F20)	3.13×10^{-1}	$\mathbf{3.12 \times 10^{-1}}$	$\mathbf{2.30 \times 10^{0}}$	2.10×10^{1}	$\mathbf{6.68 \times 10^{0}}$	1.98×10^{1}	$\mathbf{3.40 \times 10^{1}}$	4.11×10^{1}	$\mathbf{9.26 \times 10^{0}}$	9.32×10^{0}
CEC17(F21)	2.04×10^{2}	$\mathbf{1.00 \times 10^{2}}$	2.10×10^{2}	$\mathbf{1.00 \times 10^{2}}$	2.10×10^{2}	$\mathbf{1.01 \times 10^{2}}$	2.16×10^{2}	$\mathbf{1.04 \times 10^{2}}$	$\mathbf{2.52 \times 10^{0}}$	41.28×10^{0}
CEC17(F22)	1.01×10^{2}	$\mathbf{1.16 \times 10^{1}}$	1.02×10^{2}	$\mathbf{1.01 \times 10^{2}}$	1.02×10^{2}	$\mathbf{8.49 \times 10^{1}}$	1.05×10^{2}	$\mathbf{1.03 \times 10^{2}}$	$\mathbf{9.58 \times 10^{-1}}$	3.19×10^{1}
CEC17(F23)	$\mathbf{3.03 \times 10^{2}}$	3.05×10^{2}	$\mathbf{3.09 \times 10^{2}}$	3.14×10^{2}	$\mathbf{3.10 \times 10^{2}}$	3.15×10^{2}	$\mathbf{3.18 \times 10^{2}}$	3.27×10^{2}	$\mathbf{3.63 \times 10^{0}}$	6.97×10^{0}
CEC17(F24)	3.32×10^{2}	$\mathbf{1.00 \times 10^{2}}$	3.39×10^{2}	$\mathbf{1.00 \times 10^{2}}$	3.38×10^{2}	$\mathbf{1.33 \times 10^{2}}$	$\mathbf{3.43 \times 10^{2}}$	3.44×10^{2}	$\mathbf{2.88 \times 10^{0}}$	7.64×10^{1}
CEC17(f25)	3.98×10^{2}	$\mathbf{1.00 \times 10^{2}}$	4.43×10^{2}	$\mathbf{4.00 \times 10^{2}}$	4.27×10^{2}	$\mathbf{4.07 \times 10^{2}}$	$\mathbf{4.46 \times 10^{2}}$	4.49×10^{2}	$\mathbf{2.30 \times 10^{1}}$	7.41×10^{1}
CEC17(F26)	$\mathbf{2.00 \times 10^{2}}$	$\mathbf{2.00 \times 10^{2}}$	$\mathbf{3.00 \times 10^{2}}$	$\mathbf{3.00 \times 10^{2}}$	3.80×10^{2}	$\mathbf{3.12 \times 10^{2}}$	6.59×10^{2}	$\mathbf{3.84 \times 10^{2}}$	1.33×10^{2}	$\mathbf{3.99 \times 10^{1}}$
CEC17(F27)	3.90×10^{2}	$\mathbf{3.89 \times 10^{2}}$	4.00×10^{2}	$\mathbf{3.90 \times 10^{2}}$	3.99×10^{2}	$\mathbf{3.94 \times 10^{2}}$	$\mathbf{4.04 \times 10^{2}}$	4.33×10^{2}	$\mathbf{3.21 \times 10^{0}}$	9.33×10^{0}
CEC17(F28)	5.84×10^{2}	$\mathbf{0.00 \times 10^{0}}$	6.12×10^{2}	$\mathbf{3.00 \times 10^{2}}$	5.99×10^{2}	$\mathbf{3.45 \times 10^{2}}$	$\mathbf{6.12 \times 10^{2}}$	$\mathbf{6.12 \times 10^{2}}$	$\mathbf{1.40 \times 10^{1}}$	1.36×10^{2}
CEC17(F29)	2.36×10^{2}	$\mathbf{2.35 \times 10^{2}}$	$\mathbf{2.51 \times 10^{2}}$	2.69×10^{2}	$\mathbf{2.55 \times 10^{2}}$	2.76×10^{2}	$\mathbf{2.90 \times 10^{2}}$	3.23×10^{2}	$\mathbf{1.43 \times 10^{1}}$	3.06×10^{1}
CEC17(F30)	1.11×10^{3}	$\mathbf{4.81 \times 10^{2}}$	3.84×10^{3}	$\mathbf{6.27 \times 10^{2}}$	$\mathbf{3.65 \times 10^{3}}$	3.97×10^{4}	$\mathbf{6.18 \times 10^{3}}$	8.18×10^{5}	$\mathbf{1.51 \times 10^{3}}$	1.78×10^{5}
Ranking (W|T|L)	8|2|19	19|2|8	10|3|16	16|3|10	13|0|16	16|0|13	15|1|13	13|1|15	18|0|11	11|0|18

Table 4 Results of 51 separate runs using the CEC2017 functions of 30 variables and 300,000 function evaluations to compare the statistical performance of mRIME and original RIME

Function	Best value		Median value		Average value		Worst value		Std	
	RIME	mRIME	RIME	mRIME	RIME	mRIME	RIME	mRIME	RIME	mRIME
CEC17(F1)	2.41×10^{1}	$\mathbf{2.85 \times 10^{-3}}$	2.00×10^{3}	$\mathbf{1.03 \times 10^{1}}$	4.64×10^{3}	$\mathbf{2.13 \times 10^{2}}$	1.47×10^{4}	$\mathbf{4.12 \times 10^{3}}$	5.06×10^{3}	$\mathbf{8.94 \times 10^{2}}$
CEC17(F3)	6.16×10^{3}	$\mathbf{0.00 \times 10^{0}}$	7.67×10^{3}	$\mathbf{0.00 \times 10^{0}}$	1.04×10^{4}	$\mathbf{0.00 \times 10^{0}}$	2.33×10^{4}	$\mathbf{0.00 \times 10^{0}}$	4.83×10^{3}	$\mathbf{0.00 \times 10^{0}}$
CEC17(F4)	5.86×10^{1}	$\mathbf{2.84 \times 10^{-7}}$	8.76×10^{1}	$\mathbf{8.52 \times 10^{1}}$	8.89×10^{1}	$\mathbf{7.30 \times 10^{1}}$	1.35×10^{2}	$\mathbf{8.80 \times 10^{1}}$	$\mathbf{1.87 \times 10^{1}}$	2.09×10^{1}
CEC17(F5)	$\mathbf{2.19 \times 10^{1}}$	5.27×10^{1}	$\mathbf{3.18 \times 10^{1}}$	7.72×10^{1}	$\mathbf{3.25 \times 10^{1}}$	8.27×10^{1}	$\mathbf{5.27 \times 10^{1}}$	1.40×10^{2}	$\mathbf{7.19 \times 10^{0}}$	2.32×10^{1}
CEC17(F6)	$\mathbf{3.07 \times 10^{-2}}$	3.37×10^{0}	$\mathbf{8.25 \times 10^{-2}}$	5.77×10^{0}	$\mathbf{9.73 \times 10^{-2}}$	6.87×10^{0}	$\mathbf{2.27 \times 10^{-1}}$	1.46×10^{1}	$\mathbf{5.33 \times 10^{-2}}$	3.67×10^{0}
CEC17(F7)	$\mathbf{6.04 \times 10^{1}}$	9.45×10^{1}	$\mathbf{7.44 \times 10^{1}}$	1.41×10^{2}	$\mathbf{7.55 \times 10^{1}}$	1.50×10^{2}	$\mathbf{1.12 \times 10^{2}}$	2.50×10^{2}	$\mathbf{1.30 \times 10^{1}}$	4.02×10^{1}
CEC17(F8)	$\mathbf{1.59 \times 10^{1}}$	3.98×10^{1}	$\mathbf{2.98 \times 10^{1}}$	9.18×10^{1}	$\mathbf{3.21 \times 10^{1}}$	8.88×10^{1}	$\mathbf{4.88 \times 10^{1}}$	1.48×10^{2}	$\mathbf{8.60 \times 10^{0}}$	2.83×10^{1}
CEC17(F9)	$\mathbf{8.52 \times 10^{-4}}$	4.48×10^{2}	$\mathbf{1.79 \times 10^{-1}}$	9.76×10^{2}	$\mathbf{3.05 \times 10^{-1}}$	1.06×10^{3}	$\mathbf{1.09 \times 10^{0}}$	2.36×10^{3}	$\mathbf{3.26 \times 10^{-1}}$	4.98×10^{2}
CEC17(F10)	$\mathbf{7.16 \times 10^{2}}$	2.72×10^{3}	$\mathbf{1.37 \times 10^{3}}$	4.05×10^{3}	$\mathbf{1.46 \times 10^{3}}$	3.90×10^{3}	$\mathbf{2.87 \times 10^{3}}$	5.01×10^{3}	$\mathbf{5.75 \times 10^{2}}$	5.91×10^{2}
CEC17(F11)	$\mathbf{3.01 \times 10^{1}}$	5.23×10^{1}	$\mathbf{6.81 \times 10^{1}}$	2.54×10^{2}	$\mathbf{7.49 \times 10^{1}}$	2.89×10^{2}	$\mathbf{1.23 \times 10^{2}}$	4.88×10^{2}	$\mathbf{2.97 \times 10^{1}}$	1.07×10^{2}
CEC17(F12)	1.09×10^{4}	$\mathbf{4.89 \times 10^{3}}$	3.74×10^{4}	$\mathbf{2.16 \times 10^{4}}$	4.90×10^{4}	$\mathbf{3.18 \times 10^{4}}$	$\mathbf{1.34 \times 10^{5}}$	1.58×10^{5}	3.33×10^{4}	$\mathbf{3.32 \times 10^{4}}$
CEC17(F13)	$\mathbf{4.14 \times 10^{2}}$	7.20×10^{2}	1.01×10^{4}	$\mathbf{3.27 \times 10^{3}}$	1.23×10^{4}	$\mathbf{4.00 \times 10^{3}}$	3.53×10^{4}	$\mathbf{1.05 \times 10^{4}}$	1.11×10^{4}	$\mathbf{2.67 \times 10^{3}}$
CEC17(F14)	6.31×10^{2}	$\mathbf{6.64 \times 10^{1}}$	3.63×10^{3}	$\mathbf{3.01 \times 10^{2}}$	5.27×10^{3}	$\mathbf{2.88 \times 10^{2}}$	2.78×10^{4}	$\mathbf{5.38 \times 10^{2}}$	6.03×10^{3}	$\mathbf{1.21 \times 10^{2}}$
CEC17(F15)	$\mathbf{1.93 \times 10^{2}}$	2.60×10^{2}	2.59×10^{3}	$\mathbf{5.27 \times 10^{2}}$	4.98×10^{3}	$\mathbf{5.94 \times 10^{2}}$	1.97×10^{4}	$\mathbf{1.52 \times 10^{3}}$	5.84×10^{3}	$\mathbf{3.16 \times 10^{2}}$
CEC17(F16)	$\mathbf{4.85 \times 10^{1}}$	3.97×10^{2}	$\mathbf{5.09 \times 10^{2}}$	8.72×10^{2}	$\mathbf{4.39 \times 10^{2}}$	9.05×10^{2}	$\mathbf{7.02 \times 10^{2}}$	1.42×10^{3}	$\mathbf{1.87 \times 10^{2}}$	2.77×10^{2}
CEC17(F17)	$\mathbf{4.10 \times 10^{1}}$	1.76×10^{2}	$\mathbf{2.81 \times 10^{2}}$	4.34×10^{2}	$\mathbf{2.61 \times 10^{2}}$	4.09×10^{2}	$\mathbf{4.91 \times 10^{2}}$	6.54×10^{2}	$\mathbf{1.33 \times 10^{2}}$	1.42×10^{2}

Table 4 continued

Function	Best value		Median value		Average value		Worst value		Std	
	RIME	mRIME	RIME	mRIME	RIME	mRIME	RIME	mRIME	RIME	mRIME
CEC17(F18)	4.88×10^{4}	$\mathbf{6.09 \times 10^{2}}$	2.19×10^{5}	$\mathbf{1.44 \times 10^{3}}$	2.34×10^{5}	$\mathbf{2.09 \times 10^{3}}$	5.55×10^{5}	$\mathbf{6.57 \times 10^{3}}$	1.43×10^{5}	$\mathbf{1.62 \times 10^{3}}$
CEC17(F19)	1.44×10^{2}	$\mathbf{3.73 \times 10^{1}}$	1.07×10^{3}	$\mathbf{2.30 \times 10^{2}}$	4.81×10^{3}	$\mathbf{2.39 \times 10^{2}}$	2.64×10^{4}	$\mathbf{3.81 \times 10^{2}}$	7.64×10^{3}	$\mathbf{1.04 \times 10^{2}}$
CEC17(F20)	$\mathbf{3.63 \times 10^{1}}$	1.16×10^{2}	$\mathbf{1.86 \times 10^{2}}$	4.53×10^{2}	$\mathbf{2.18 \times 10^{2}}$	4.22×10^{2}	$\mathbf{4.59 \times 10^{2}}$	6.75×10^{2}	$\mathbf{1.27 \times 10^{2}}$	1.69×10^{2}
CEC17(F21)	2.21×10^{2}	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{2.31 \times 10^{2}}$	2.80×10^{2}	$\mathbf{2.32 \times 10^{2}}$	2.81×10^{2}	$\mathbf{2.47 \times 10^{2}}$	3.45×10^{2}	$\mathbf{7.85 \times 10^{0}}$	4.87×10^{1}
CEC17(F22)	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$	1.02×10^{3}	$\mathbf{1.00 \times 10^{2}}$	9.91×10^{2}	$\mathbf{1.02 \times 10^{2}}$	2.26×10^{3}	$\mathbf{1.05 \times 10^{2}}$	9.26×10^{2}	$\mathbf{1.88 \times 10^{0}}$
CEC17(F23)	$\mathbf{3.82 \times 10^{2}}$	4.04×10^{2}	$\mathbf{3.96 \times 10^{2}}$	4.52×10^{2}	$\mathbf{3.97 \times 10^{2}}$	4.58×10^{2}	$\mathbf{4.14 \times 10^{2}}$	5.16×10^{2}	$\mathbf{7.77 \times 10^{0}}$	2.81×10^{1}
CEC17(F24)	$\mathbf{4.37 \times 10^{2}}$	4.81×10^{2}	$\mathbf{4.58 \times 10^{2}}$	5.33×10^{2}	$\mathbf{4.60 \times 10^{2}}$	5.35×10^{2}	$\mathbf{4.86 \times 10^{2}}$	6.03×10^{2}	$\mathbf{1.09 \times 10^{1}}$	3.58×10^{1}
CEC17(F25)	$\mathbf{3.83 \times 10^{2}}$	3.84×10^{2}	$\mathbf{3.87 \times 10^{2}}$	3.88×10^{2}	$\mathbf{3.87 \times 10^{2}}$	3.89×10^{2}	$\mathbf{3.88 \times 10^{2}}$	4.24×10^{2}	$\mathbf{1.33 \times 10^{0}}$	8.27×10^{0}
CEC17(F26)	1.34×10^{3}	$\mathbf{3.00 \times 10^{2}}$	$\mathbf{1.57 \times 10^{3}}$	1.99×10^{3}	$\mathbf{1.57 \times 10^{3}}$	1.66×10^{3}	$\mathbf{1.89 \times 10^{3}}$	2.62×10^{3}	$\mathbf{1.26 \times 10^{2}}$	8.59×10^{2}
CEC17(F27)	$\mathbf{5.14 \times 10^{2}}$	5.17×10^{2}	5.34×10^{2}	$\mathbf{5.33 \times 10^{2}}$	$\mathbf{5.34 \times 10^{2}}$	5.42×10^{2}	$\mathbf{5.56 \times 10^{2}}$	5.81×10^{2}	$\mathbf{1.09 \times 10^{1}}$	2.08×10^{1}
CEC17(F28)	3.95×10^{2}	$\mathbf{3.00 \times 10^{2}}$	4.12×10^{2}	$\mathbf{4.03 \times 10^{2}}$	4.14×10^{2}	$\mathbf{3.69 \times 10^{2}}$	$\mathbf{4.64 \times 10^{2}}$	4.67×10^{2}	$\mathbf{1.54 \times 10^{1}}$	6.80×10^{1}
CEC17(F29)	$\mathbf{4.82 \times 10^{2}}$	5.25×10^{2}	$\mathbf{6.04 \times 10^{2}}$	1.04×10^{3}	$\mathbf{6.42 \times 10^{2}}$	1.02×10^{3}	$\mathbf{9.69 \times 10^{2}}$	1.44×10^{3}	$\mathbf{1.39 \times 10^{2}}$	2.62×10^{2}
CEC17(F30)	2.24×10^{3}	$\mathbf{2.13 \times 10^{3}}$	6.80×10^{3}	$\mathbf{4.08 \times 10^{3}}$	6.08×10^{3}	$\mathbf{4.94 \times 10^{3}}$	$\mathbf{1.04 \times 10^{4}}$	1.41×10^{4}	$\mathbf{2.93 \times 10^{3}}$	3.01×10^{3}
Ranking (W|T|L)	17|1|11	11|1|17	16|0|13	13|0|16	17|0|12	12|0|17	20|0|9	9|0|20	20|0|9	9|0|20

Table 5 Results of 51 separate runs using the CEC2017 functions of 50 variables and 500,000 function evaluations to compare the statistical performance of mRIME and original RIME

Function	Best value		Median value		Average value		Worst value		Std	
	RIME	mRIME	RIME	mRIME	RIME	mRIME	RIME	mRIME	RIME	mRIME
CEC17(F1)	1.80×10^{2}	$\mathbf{4.53 \times 10^{-1}}$	$\mathbf{4.86 \times 10^{3}}$	7.54×10^{3}	2.68×10^{4}	$\mathbf{1.15 \times 10^{4}}$	2.99×10^{5}	$\mathbf{3.00 \times 10^{4}}$	7.19×10^{4}	$\mathbf{1.15 \times 10^{4}}$
CEC17(F3)	3.96×10^{4}	$\mathbf{0.00 \times 10^{0}}$	5.72×10^{4}	$\mathbf{8.55 \times 10^{-8}}$	5.71×10^{4}	$\mathbf{1.24 \times 10^{-5}}$	7.78×10^{4}	$\mathbf{1.67 \times 10^{-4}}$	1.14×10^{4}	$\mathbf{4.02 \times 10^{-5}}$
CEC17(F4)	2.75×10^{1}	$\mathbf{8.06 \times 10^{0}}$	1.21×10^{2}	$\mathbf{9.90 \times 10^{1}}$	1.17×10^{2}	$\mathbf{9.32 \times 10^{1}}$	2.24×10^{2}	$\mathbf{1.89 \times 10^{2}}$	$\mathbf{4.94 \times 10^{1}}$	5.96×10^{1}
CEC17(F5)	$\mathbf{5.41 \times 10^{1}}$	1.34×10^{2}	$\mathbf{6.97 \times 10^{1}}$	1.80×10^{2}	$\mathbf{6.69 \times 10^{1}}$	1.99×10^{2}	$\mathbf{7.76 \times 10^{1}}$	2.82×10^{2}	$\mathbf{8.44 \times 10^{0}}$	4.92×10^{1}
CEC17(F6)	$\mathbf{1.88 \times 10^{-1}}$	1.07×10^{1}	$\mathbf{2.57 \times 10^{-1}}$	1.76×10^{1}	$\mathbf{2.88 \times 10^{-1}}$	1.84×10^{1}	$\mathbf{4.81 \times 10^{-1}}$	2.76×10^{1}	$\mathbf{7.50 \times 10^{-2}}$	4.56×10^{0}
CEC17(F7)	$\mathbf{9.70 \times 10^{1}}$	3.13×10^{2}	$\mathbf{1.32 \times 10^{2}}$	4.15×10^{2}	$\mathbf{1.29 \times 10^{2}}$	4.30×10^{2}	$\mathbf{1.61 \times 10^{2}}$	6.04×10^{2}	$\mathbf{1.50 \times 10^{1}}$	8.44×10^{1}
CEC17(F8)	$\mathbf{4.59 \times 10^{1}}$	1.19×10^{2}	$\mathbf{7.16 \times 10^{1}}$	1.96×10^{2}	$\mathbf{6.96 \times 10^{1}}$	2.04×10^{2}	$\mathbf{9.85 \times 10^{1}}$	3.17×10^{2}	$\mathbf{1.47 \times 10^{1}}$	4.98×10^{1}
CEC17(F9)	$\mathbf{5.37 \times 10^{-1}}$	2.76×10^{3}	$\mathbf{1.36 \times 10^{0}}$	4.87×10^{3}	$\mathbf{1.08 \times 10^{1}}$	4.98×10^{3}	$\mathbf{1.89 \times 10^{2}}$	9.34×10^{3}	$\mathbf{4.09 \times 10^{1}}$	1.62×10^{3}
CEC17(F10)	$\mathbf{2.03 \times 10^{3}}$	5.45×10^{3}	$\mathbf{3.38 \times 10^{3}}$	7.21×10^{3}	$\mathbf{3.58 \times 10^{3}}$	6.93×10^{3}	$\mathbf{5.92 \times 10^{3}}$	8.73×10^{3}	1.10×10^{3}	$\mathbf{8.64 \times 10^{2}}$
CEC17(F11)	$\mathbf{7.11 \times 10^{1}}$	2.49×10^{2}	$\mathbf{1.25 \times 10^{2}}$	4.27×10^{2}	$\mathbf{1.34 \times 10^{2}}$	4.02×10^{2}	$\mathbf{2.16 \times 10^{2}}$	5.57×10^{2}	$\mathbf{3.83 \times 10^{1}}$	1.01×10^{2}
CEC17(F12)	1.98×10^{5}	$\mathbf{1.12 \times 10^{5}}$	9.50×10^{5}	$\mathbf{5.57 \times 10^{5}}$	1.54×10^{6}	$\mathbf{1.24 \times 10^{6}}$	8.94×10^{6}	$\mathbf{4.74 \times 10^{6}}$	1.94×10^{6}	$\mathbf{1.43 \times 10^{6}}$
CEC17(F13)	$\mathbf{1.88 \times 10^{3}}$	7.44×10^{3}	$\mathbf{6.55 \times 10^{3}}$	2.12×10^{4}	$\mathbf{7.38 \times 10^{3}}$	2.58×10^{4}	$\mathbf{2.88 \times 10^{4}}$	5.54×10^{4}	$\mathbf{5.78 \times 10^{3}}$	1.24×10^{4}
CEC17(F14)	1.09×10^{3}	$\mathbf{2.11 \times 10^{2}}$	3.61×10^{4}	$\mathbf{5.38 \times 10^{2}}$	3.43×10^{4}	$\mathbf{5.73 \times 10^{2}}$	5.87×10^{4}	$\mathbf{9.38 \times 10^{2}}$	$\mathbf{1.61 \times 10^{4}}$	1.73×10^{2}
CEC17(F15)	$\mathbf{7.75 \times 10^{2}}$	9.52×10^{2}	5.32×10^{3}	$\mathbf{3.23 \times 10^{3}}$	7.18×10^{3}	$\mathbf{5.08 \times 10^{3}}$	2.53×10^{4}	$\mathbf{1.81 \times 10^{4}}$	6.77×10^{3}	$\mathbf{4.90 \times 10^{3}}$
CEC17(F16)	$\mathbf{3.26 \times 10^{2}}$	8.35×10^{2}	$\mathbf{9.44 \times 10^{2}}$	1.91×10^{3}	$\mathbf{9.61 \times 10^{2}}$	1.81×10^{3}	$\mathbf{1.51 \times 10^{3}}$	2.31×10^{3}	$\mathbf{3.09 \times 10^{2}}$	3.80×10^{2}
CEC17(F17)	$\mathbf{2.66 \times 10^{2}}$	6.75×10^{2}	$\mathbf{5.41 \times 10^{2}}$	1.51×10^{3}	$\mathbf{5.70 \times 10^{2}}$	1.42×10^{3}	$\mathbf{1.10 \times 10^{3}}$	1.90×10^{3}	$\mathbf{1.81 \times 10^{2}}$	3.30×10^{2}
CEC17(F18)	1.18×10^{5}	$\mathbf{2.55 \times 10^{3}}$	5.83×10^{5}	$\mathbf{1.83 \times 10^{4}}$	7.72×10^{5}	$\mathbf{3.40 \times 10^{4}}$	2.51×10^{6}	$\mathbf{1.03 \times 10^{5}}$	6.43×10^{5}	$\mathbf{3.30 \times 10^{4}}$

Table 5 continued

Function	Best value		Median value		Average value		Worst value		Std	
	RIME	mRIME	RIME	mRIME	RIME	mRIME	RIME	mRIME	RIME	mRIME
CEC17(F19)	1.16×10^3	$\mathbf{2.22 \times 10^2}$	1.31×10^4	$\mathbf{6.12 \times 10^2}$	1.48×10^4	$\mathbf{6.70 \times 10^2}$	3.03×10^4	$\mathbf{1.23 \times 10^3}$	9.29×10^3	$\mathbf{2.86 \times 10^2}$
CEC17(F20)	$\mathbf{1.28 \times 10^2}$	5.69×10^2	$\mathbf{3.71 \times 10^2}$	9.63×10^2	$\mathbf{4.60 \times 10^2}$	9.84×10^2	$\mathbf{9.10 \times 10^2}$	1.40×10^3	2.45×10^2	$\mathbf{1.97 \times 10^2}$
CEC17(F21)	$\mathbf{2.48 \times 10^2}$	2.96×10^2	$\mathbf{2.63 \times 10^2}$	3.89×10^2	$\mathbf{2.66 \times 10^2}$	3.89×10^2	$\mathbf{2.99 \times 10^2}$	4.42×10^2	$\mathbf{1.20 \times 10^1}$	3.30×10^1
CEC17(F22)	$\mathbf{2.49 \times 10^3}$	5.55×10^3	$\mathbf{4.00 \times 10^3}$	7.26×10^3	$\mathbf{4.90 \times 10^3}$	7.40×10^3	1.23×10^4	$\mathbf{9.10 \times 10^3}$	2.35×10^3	$\mathbf{1.06 \times 10^3}$
CEC17(F23)	$\mathbf{4.69 \times 10^2}$	5.42×10^2	$\mathbf{5.25 \times 10^2}$	6.58×10^2	$\mathbf{5.23 \times 10^2}$	6.60×10^2	$\mathbf{5.66 \times 10^2}$	7.38×10^2	$\mathbf{2.07 \times 10^1}$	4.75×10^1
CEC17(F24)	$\mathbf{5.53 \times 10^2}$	6.36×10^2	$\mathbf{5.78 \times 10^2}$	7.12×10^2	$\mathbf{5.80 \times 10^2}$	7.09×10^2	$\mathbf{6.46 \times 10^2}$	8.10×10^2	$\mathbf{1.98 \times 10^1}$	4.79×10^1
CEC17(F25)	4.67×10^2	$\mathbf{4.63 \times 10^2}$	$\mathbf{5.30 \times 10^2}$	5.56×10^2	5.37×10^2	$\mathbf{5.34 \times 10^2}$	5.86×10^2	$\mathbf{5.81 \times 10^2}$	$\mathbf{3.31 \times 10^1}$	3.76×10^1
CEC17(F26)	$\mathbf{1.82 \times 10^3}$	3.00×10^2	$\mathbf{2.23 \times 10^3}$	3.62×10^3	$\mathbf{2.22 \times 10^3}$	3.40×10^3	$\mathbf{2.65 \times 10^3}$	4.55×10^3	$\mathbf{2.14 \times 10^2}$	1.12×10^3
CEC17(F27)	$\mathbf{6.44 \times 10^2}$	6.55×10^2	$\mathbf{7.34 \times 10^2}$	8.22×10^2	$\mathbf{7.48 \times 10^2}$	8.90×10^2	$\mathbf{8.65 \times 10^2}$	1.35×10^3	$\mathbf{5.87 \times 10^1}$	1.90×10^2
CEC17(F28)	5.01×10^2	$\mathbf{4.59 \times 10^2}$	5.09×10^2	$\mathbf{5.03 \times 10^2}$	5.11×10^2	$\mathbf{5.03 \times 10^2}$	$\mathbf{5.21 \times 10^2}$	6.00×10^2	$\mathbf{5.06 \times 10^0}$	3.34×10^1
CEC17(F29)	$\mathbf{4.10 \times 10^2}$	1.36×10^3	$\mathbf{8.63 \times 10^2}$	2.09×10^3	$\mathbf{8.06 \times 10^2}$	2.00×10^3	$\mathbf{1.11 \times 10^3}$	3.05×10^3	$\mathbf{1.85 \times 10^2}$	4.69×10^2
CEC17(F30)	8.26×10^5	$\mathbf{6.54 \times 10^5}$	$\mathbf{9.26 \times 10^5}$	1.37×10^6	$\mathbf{9.65 \times 10^5}$	1.57×10^6	$\mathbf{1.44 \times 10^6}$	2.60×10^6	$\mathbf{1.65 \times 10^5}$	5.82×10^5
Ranking (W|T|L)	19|0|10	10|0|19	21|0|8	8|0|21	19|0|10	10|0|19	19|0|10	10|0|19	20|0|9	9|0|20

- Worst value: In terms of the worst objective function values, mRIME consistently outperforms RIME for most of the functions. The worst values achieved by mRIME are lower, indicating that it avoids poor solutions more effectively.
- Std: The standard deviation values provide insights into the spread or variability of the objective function values obtained by each algorithm. In most cases, mRIME has lower standard deviation values, indicating that its solutions are more consistent and less variable compared to RIME.

Based on the analysis of these metrics, it can be concluded that mRIME generally exhibits superior performance compared to the original RIME algorithm, even when the number of variables is increased to 100. It achieves lower best and average objective function values, avoids poor solutions (lower worst values), and provides more consistent results (lower standard deviation). However, there are a few functions where RIME performs better in terms of median values, indicating that the two algorithms have different strengths depending on the specific problem.

The table presents the results of 51 independent runs on the CEC2017 test functions of 100 variables with 1,000,000 function evaluations. The table compares the performance of different optimization techniques, including mRIME, RIME, WSO, TLBO, SFS, DE, GA, GSK, AMO, PSO, BBO, and ACO (Table 7).

To analyze the results, we can focus on a few key observations:

- Performance comparison: By comparing the average values in the "Ave" column, we can assess the overall performance of each technique. Lower values generally indicate better performance in optimization problems. For example, in functions F1, F3, F6, F10, F12, F14, F16, F19, F21, F22, F26, F27, and F29, the technique "mRIME" achieves the lowest average values, suggesting its effectiveness in solving these functions.
- Variability: The "Std" column represents the standard deviation, which indicates the variability of the results. Lower standard deviation values generally indicate more stable and consistent performance. For example, in functions F1, F3, F4, F5, F7, F8, F9, F12, F13, F14, F15, F16, F17, F19, F21, F23, F24, F25, F26, F27, F28, F29, and F30, the technique "mRIME" has lower standard deviation compared to other techniques, indicating more consistent performance.
- Best performance: The best-performing technique for each function is indicated in bold in the table. These are the techniques that achieved the lowest average values among all the techniques for a particular function. For example, in function F1, the technique "mRIME" achieves the lowest average value of 2.51×10^3.

Based on the results presented in the table, it appears that the "mRIME" technique performs well across multiple functions and demonstrates competitive performance compared to other optimization techniques. However, it is important to consider the specific requirements and characteristics of the optimization problem at hand when selecting the most suitable technique. Further analysis and comparisons may be needed to make more definitive conclusions about the superiority of any specific technique (Tables 8, 9, 10).

Table 6 Results of 51 separate runs using the CEC2017 functions of 100 variables and 1,000,000 function evaluations to compare the statistical performance of mRIME and original RIME

Function	Best value		Median value		Average value		Worst value		Std	
	RIME	mRIME	RIME	mRIME	RIME	mRIME	RIME	mRIME	RIME	mRIME
CEC17(F1)	4.67×10^{2}	$\mathbf{2.13 \times 10^{2}}$	1.97×10^{4}	$\mathbf{1.20 \times 10^{4}}$	8.10×10^{4}	$\mathbf{1.58 \times 10^{4}}$	5.66×10^{5}	$\mathbf{5.75 \times 10^{4}}$	1.36×10^{5}	$\mathbf{1.78 \times 10^{4}}$
CEC17(F3)	1.89×10^{5}	$\mathbf{1.71 \times 10^{0}}$	2.32×10^{5}	$\mathbf{3.15 \times 10^{1}}$	2.35×10^{5}	$\mathbf{3.92 \times 10^{2}}$	2.72×10^{5}	$\mathbf{4.26 \times 10^{3}}$	2.17×10^{4}	$\mathbf{1.05 \times 10^{3}}$
CEC17(F4)	$\mathbf{1.65 \times 10^{2}}$	1.92×10^{2}	2.50×10^{2}	$\mathbf{2.38 \times 10^{2}}$	2.49×10^{2}	$\mathbf{2.37 \times 10^{2}}$	$\mathbf{3.02 \times 10^{2}}$	3.16×10^{2}	4.03×10^{1}	$\mathbf{3.30 \times 10^{1}}$
CEC17(F5)	$\mathbf{1.13 \times 10^{2}}$	3.64×10^{2}	$\mathbf{1.67 \times 10^{2}}$	4.94×10^{2}	$\mathbf{1.63 \times 10^{2}}$	5.04×10^{2}	$\mathbf{1.98 \times 10^{2}}$	6.14×10^{2}	$\mathbf{2.18 \times 10^{1}}$	5.51×10^{1}
CEC17(F6)	$\mathbf{4.91 \times 10^{-1}}$	2.61×10^{1}	$\mathbf{6.94 \times 10^{-1}}$	3.45×10^{1}	$\mathbf{6.98 \times 10^{-1}}$	3.47×10^{1}	$\mathbf{1.00 \times 10^{0}}$	4.58×10^{1}	$\mathbf{1.43 \times 10^{-1}}$	4.29×10^{0}
CEC17(F7)	$\mathbf{2.05 \times 10^{2}}$	1.00×10^{3}	$\mathbf{2.93 \times 10^{2}}$	1.37×10^{3}	$\mathbf{2.86 \times 10^{2}}$	1.43×10^{3}	$\mathbf{3.44 \times 10^{2}}$	2.07×10^{3}	$\mathbf{3.69 \times 10^{1}}$	2.84×10^{2}
CEC17(F8)	$\mathbf{1.33 \times 10^{2}}$	3.28×10^{2}	$\mathbf{1.57 \times 10^{2}}$	5.00×10^{2}	$\mathbf{1.60 \times 10^{2}}$	4.93×10^{2}	$\mathbf{2.23 \times 10^{2}}$	6.21×10^{2}	$\mathbf{1.89 \times 10^{1}}$	7.62×10^{1}
CEC17(F9)	$\mathbf{3.70 \times 10^{0}}$	1.17×10^{4}	$\mathbf{8.30 \times 10^{0}}$	1.74×10^{4}	$\mathbf{1.05 \times 10^{2}}$	1.93×10^{4}	$\mathbf{6.81 \times 10^{2}}$	4.10×10^{4}	$\mathbf{2.37 \times 10^{2}}$	6.91×10^{3}
CEC17(F10)	$\mathbf{9.27 \times 10^{3}}$	1.18×10^{4}	1.86×10^{4}	$\mathbf{1.43 \times 10^{4}}$	1.97×10^{4}	$\mathbf{1.47 \times 10^{4}}$	2.87×10^{4}	$\mathbf{1.72 \times 10^{4}}$	5.91×10^{3}	$\mathbf{1.52 \times 10^{3}}$
CEC17(F11)	$\mathbf{7.74 \times 10^{2}}$	1.53×10^{3}	$\mathbf{1.05 \times 10^{3}}$	2.10×10^{3}	$\mathbf{1.04 \times 10^{3}}$	2.17×10^{3}	$\mathbf{1.28 \times 10^{3}}$	3.05×10^{3}	$\mathbf{1.39 \times 10^{2}}$	3.64×10^{2}
CEC17(F12)	$\mathbf{5.34 \times 10^{5}}$	1.10×10^{6}	4.58×10^{6}	$\mathbf{3.44 \times 10^{6}}$	7.24×10^{6}	$\mathbf{4.95 \times 10^{6}}$	2.26×10^{7}	$\mathbf{1.52 \times 10^{7}}$	5.73×10^{6}	$\mathbf{4.38 \times 10^{6}}$
CEC17(F13)	$\mathbf{6.20 \times 10^{3}}$	7.14×10^{3}	$\mathbf{1.11 \times 10^{4}}$	1.79×10^{4}	$\mathbf{1.20 \times 10^{4}}$	2.09×10^{4}	$\mathbf{2.18 \times 10^{4}}$	4.68×10^{4}	$\mathbf{4.04 \times 10^{3}}$	1.01×10^{4}
CEC17(F14)	1.42×10^{5}	$\mathbf{1.82 \times 10^{3}}$	4.28×10^{5}	$\mathbf{9.37 \times 10^{3}}$	4.93×10^{5}	$\mathbf{2.82 \times 10^{4}}$	9.79×10^{5}	$\mathbf{3.21 \times 10^{5}}$	2.43×10^{5}	$\mathbf{6.80 \times 10^{4}}$
CEC17(F15)	$\mathbf{9.29 \times 10^{2}}$	1.89×10^{3}	$\mathbf{3.00 \times 10^{3}}$	1.13×10^{4}	$\mathbf{5.06 \times 10^{3}}$	1.50×10^{4}	$\mathbf{2.13 \times 10^{4}}$	4.70×10^{4}	$\mathbf{4.95 \times 10^{3}}$	1.11×10^{4}
CEC17(F16)	$\mathbf{1.60 \times 10^{3}}$	2.83×10^{3}	$\mathbf{2.75 \times 10^{3}}$	4.30×10^{3}	$\mathbf{3.10 \times 10^{3}}$	4.40×10^{3}	7.03×10^{3}	$\mathbf{5.75 \times 10^{3}}$	1.53×10^{3}	$\mathbf{8.63 \times 10^{2}}$
CEC17(F17)	$\mathbf{1.07 \times 10^{3}}$	2.72×10^{3}	$\mathbf{1.79 \times 10^{3}}$	3.74×10^{3}	$\mathbf{2.06 \times 10^{3}}$	3.79×10^{3}	$\mathbf{3.77 \times 10^{3}}$	4.67×10^{3}	7.28×10^{2}	$\mathbf{5.78 \times 10^{2}}$
CEC17(F18)	3.59×10^{5}	$\mathbf{2.17 \times 10^{4}}$	1.18×10^{6}	$\mathbf{5.60 \times 10^{4}}$	1.32×10^{6}	$\mathbf{7.55 \times 10^{4}}$	4.75×10^{6}	$\mathbf{1.75 \times 10^{5}}$	1.03×10^{6}	$\mathbf{4.62 \times 10^{4}}$

Table 6 continued

Function	Best value		Median value		Average value		Worst value		Std	
	RIME	mRIME	RIME	mRIME	RIME	mRIME	RIME	mRIME	RIME	mRIME
CEC17(F19)	$\mathbf{4.20 \times 10^2}$	9.53×10^2	$\mathbf{1.26 \times 10^3}$	2.80×10^3	$\mathbf{2.82 \times 10^3}$	6.74×10^3	$\mathbf{1.28 \times 10^4}$	4.01×10^4	$\mathbf{3.28 \times 10^3}$	9.35×10^3
CEC17(F20)	$\mathbf{1.40 \times 10^3}$	2.38×10^3	$\mathbf{2.76 \times 10^3}$	3.20×10^3	$\mathbf{2.79 \times 10^3}$	3.14×10^3	4.56×10^3	$\mathbf{3.84 \times 10^3}$	9.45×10^2	$\mathbf{3.57 \times 10^2}$
CEC17(F21)	$\mathbf{3.84 \times 10^2}$	6.48×10^2	$\mathbf{4.19 \times 10^2}$	7.67×10^2	$\mathbf{4.16 \times 10^2}$	7.70×10^2	$\mathbf{4.45 \times 10^2}$	8.89×10^2	$\mathbf{1.54 \times 10^1}$	6.28×10^1
CEC17(F22)	$\mathbf{9.31 \times 10^3}$	1.45×10^4	$\mathbf{1.56 \times 10^4}$	1.64×10^4	1.76×10^4	$\mathbf{1.64 \times 10^4}$	2.90×10^4	$\mathbf{1.88 \times 10^4}$	5.50×10^3	$\mathbf{1.20 \times 10^3}$
CEC17(F23)	$\mathbf{7.48 \times 10^2}$	8.55×10^2	$\mathbf{7.96 \times 10^2}$	1.19×10^3	$\mathbf{8.07 \times 10^2}$	1.19×10^3	$\mathbf{9.02 \times 10^2}$	1.58×10^3	$\mathbf{4.34 \times 10^1}$	1.48×10^2
CEC17(F24)	$\mathbf{1.10 \times 10^3}$	1.41×10^3	$\mathbf{1.19 \times 10^3}$	1.77×10^3	$\mathbf{1.19 \times 10^3}$	1.83×10^3	$\mathbf{1.27 \times 10^3}$	2.84×10^3	$\mathbf{3.84 \times 10^1}$	3.12×10^2
CEC17(F25)	$\mathbf{6.18 \times 10^2}$	6.66×10^2	8.16×10^2	$\mathbf{7.77 \times 10^2}$	8.07×10^2	$\mathbf{7.81 \times 10^2}$	$\mathbf{8.65 \times 10^2}$	9.35×10^2	$\mathbf{5.21 \times 10^1}$	5.75×10^1
CEC17(F26)	$\mathbf{5.40 \times 10^3}$	8.94×10^3	$\mathbf{6.17 \times 10^3}$	1.11×10^4	$\mathbf{6.16 \times 10^3}$	1.13×10^4	$\mathbf{6.97 \times 10^3}$	1.39×10^4	$\mathbf{4.01 \times 10^2}$	1.35×10^3
CEC17(F27)	$\mathbf{7.36 \times 10^2}$	7.45×10^2	$\mathbf{8.11 \times 10^2}$	1.07×10^3	$\mathbf{8.10 \times 10^2}$	1.07×10^3	$\mathbf{9.08 \times 10^2}$	1.38×10^3	$\mathbf{0.08 \times 10^1}$	1.41×10^2
CEC17(F28)	5.62×10^2	$\mathbf{5.27 \times 10^2}$	6.04×10^2	$\mathbf{5.78 \times 10^2}$	$\mathbf{6.19 \times 10^2}$	1.17×10^3	$\mathbf{6.86 \times 10^2}$	1.29×10^4	3.28×10^1	$\mathbf{2.70 \times 10^1}$
CEC17(F29)	$\mathbf{2.08 \times 10^3}$	2.81×10^3	$\mathbf{2.63 \times 10^3}$	4.59×10^3	$\mathbf{2.65 \times 10^3}$	4.60×10^3	$\mathbf{3.62 \times 10^3}$	6.64×10^3	$\mathbf{3.97 \times 10^2}$	7.84×10^2
CEC17(F30)	$\mathbf{4.87 \times 10^3}$	1.84×10^4	$\mathbf{1.27 \times 10^4}$	1.06×10^5	$\mathbf{2.01 \times 10^4}$	1.88×10^5	$\mathbf{1.06 \times 10^5}$	9.24×10^5	$\mathbf{2.37 \times 10^4}$	2.16×10^5
Ranking (W|T|L)	24|0|5	5|0|24	20|0|9	9|0|20	20|0|9	9|0|20	20|0|9	9|0|20	17|0|12	12|0|17

Table 7 Results of 51 separate runs of mRIME versus original RIME and other optimization algorithms on the CEC2017 test functions with 30 variables and 300,000 function evaluations

Function		mRIME	RIME	WSO	TLBO	SFS	DE	GA	GSK	AMO	PSO	BBO	ACO
CEC17(F1)	Ave	4.64×10^{3}	2.13×10^{2}	1.37×10^{-3}	3.36×10^{3}	3.17×10^{3}	$\mathbf{0.00 \times 10^{0}}$	6.06×10^{6}	$\mathbf{0.00 \times 10^{0}}$	5.34×10^{0}	3.43×10^{3}	4.20×10^{6}	8.95×10^{10}
	Std	5.06×10^{3}	8.94×10^{2}	4.39×10^{-3}	3.19×10^{3}	3.76×10^{3}	$\mathbf{0.00 \times 10^{0}}$	1.62×10^{6}	$\mathbf{0.00 \times 10^{0}}$	7.57×10^{0}	$3.97\times^{3}$	1.87×10^{5}	2.59×10^{10}
CEC17(F3)	Ave	1.04×10^{4}	0.00×10^{0}	7.65×10^{-3}	1.03×10^{-4}	5.36×10^{1}	1.36×10^{2}	6.81×10^{4}	$\mathbf{7.07 \times 10^{-7}}$	4.56×10^{3}	1.48×10^{2}	4.39×10^{4}	2.08×10^{7}
	Std	4.83×10^{3}	0.00×10^{0}	1.92×10^{-2}	2.38×10^{-4}	2.72×10^{1}	1.36×10^{2}	2.02×10^{4}	$\mathbf{1.82 \times 10^{-6}}$	1.48×10^{3}	5.90×10^{1}	2.58×10^{4}	1.23×10^{8}
CEC17(F4)	Ave	8.89×10^{1}	7.30×10^{1}	$\mathbf{1.32 \times 10^{0}}$	5.63×10^{1}	5.91×10^{1}	5.92×10^{1}	1.51×10^{2}	1.11×10^{1}	4.6×10^{0}	8.59×10^{1}	9.99×10^{1}	9.61×10^{3}
	Std	1.87×10^{1}	2.09×10^{1}	1.15×10^{1}	3.51×10^{1}	4.04×10^{1}	$\mathbf{1.81 \times 10^{0}}$	4.09×10^{1}	2.17×10^{1}	1.07×10^{1}	3.22×10^{1}	2.27×10^{1}	1.72×10^{3}
CEC17(F5)	Ave	3.25×10^{1}	8.27×10^{1}	6.35×10^{1}	9.06×10^{1}	6.76×10^{1}	1.79×10^{2}	2.26×10^{2}	1.60×10^{2}	5.41×10^{1}	1.14×10^{2}	$\mathbf{4.21 \times 10^{1}}$	4.11×10^{2}
	Std	7.19×10^{0}	2.32×10^{1}	1.80×10^{1}	2.00×10^{1}	1.34×10^{1}	1.27×10^{1}	2.93×10^{1}	8.87×10^{0}	$\mathbf{6.68 \times 10^{0}}$	2.67×10^{1}	1.06×10^{1}	2.27×10^{1}
CEC17(F6)	Ave	9.73×10^{-2}	6.87×10^{0}	1.44×10^{0}	8.45×10^{0}	1.18×10^{-2}	$\mathbf{0.00 \times 10^{0}}$	3.86×10^{1}	1.52×10^{-6}	$\mathbf{0.00 \times 10^{0}}$	3.09×10^{0}	8.98×10^{-1}	8.31×10^{1}
	Std	5.33×10^{-2}	3.67×10^{0}	1.93×10^{0}	4.18×10^{0}	2.04×10^{-2}	$\mathbf{0.00 \times 10^{0}}$	9.95×10^{0}	2.36×10^{-6}	$\mathbf{0.00 \times 10^{0}}$	4.01×10^{0}	4.07×10^{-2}	6.11×10^{0}
CEC17(F7)	Ave	7.55×10^{1}	1.50×10^{2}	1.45×10^{2}	1.48×10^{2}	1.01×10^{2}	2.12×10^{2}	3.24×10^{2}	1.87×10^{2}	$\mathbf{9.15 \times 10^{1}}$	9.71×10^{1}	1.14×10^{2}	9.44×10^{2}
	Std	1.30×10^{1}	4.02×10^{1}	4.26×10^{1}	3.18×10^{1}	2.19×10^{1}	9.99×10^{0}	5.43×10^{1}	8.40×10^{0}	$\mathbf{6.90 \times 10^{0}}$	1.66×10^{1}	1.46×10^{1}	9.83×10^{1}
CEC17(F8)	Ave	3.21×10^{1}	8.88×10^{1}	6.54×10^{1}	7.06×10^{1}	7.50×10^{1}	1.80×10^{2}	2.43×10^{2}	1.55×10^{2}	5.42×10^{1}	9.73×10^{1}	$\mathbf{4.31 \times 10^{1}}$	3.63×10^{2}
	Std	8.60×10^{0}	2.83×10^{1}	1.67×10^{1}	1.44×10^{1}	1.96×10^{1}	1.14×10^{1}	3.07×10^{1}	1.11×10^{1}	$\mathbf{5.76 \times 10^{0}}$	2.11×10^{1}	1.00×10^{1}	1.97×10^{1}
CEC17(F9)	Ave	3.05×10^{-1}	1.06×10^{3}	2.91×10^{2}	2.07×10^{2}	2.47×10^{1}	$\mathbf{0.00 \times 10^{0}}$	1.12×10^{3}	$\mathbf{0.00 \times 10^{0}}$	1.05×10^{-1}	1.01×10^{3}	1.09×10^{2}	1.39×10^{4}
	Std	3.26×10^{-1}	4.98×10^{2}	3.54×10^{2}	1.29×10^{2}	6.83×10^{1}	$\mathbf{0.00 \times 10^{0}}$	1.42×10^{3}	$\mathbf{0.00 \times 10^{0}}$	2.07×10^{-1}	1.15×10^{3}	8.42×10^{1}	1.46×10^{3}
CEC17(F10)	Ave	1.46×10^{3}	3.90×10^{3}	2.94×10^{3}	6.01×10^{3}	2.35×10^{3}	6.61×10^{3}	4.48×10^{3}	6.69×10^{3}	3.62×10^{3}	3.20×10^{3}	$\mathbf{2.29 \times 10^{3}}$	7.38×10^{3}
	Std	5.75×10^{2}	5.91×10^{2}	$\mathbf{2.21 \times 10^{2}}$	1.17×10^{3}	5.20×10^{2}	4.36×10^{2}	8.44×10^{2}	3.54×10^{2}	2.66×10^{2}	5.52×10^{2}	4.48×10^{2}	2.24×10^{2}
CEC17(F11)	Ave	7.49×10^{1}	2.89×10^{2}	1.04×10^{2}	1.36×10^{2}	4.57×10^{1}	7.49×10^{1}	1.33×10^{3}	$\mathbf{3.30 \times 10^{1}}$	4.91×10^{1}	1.14×10^{2}	1.43×10^{3}	5.54×10^{3}

Table 7 continued

Function		mRIME	RIME	WSO	TLBO	SFS	DE	GA	GSK	AMO	PSO	BBO	ACO
	Std	2.97×10^{1}	1.07×10^{2}	4.23×10^{1}	4.75×10^{1}	2.85×10^{1}	3.10×10^{1}	1.01×10^{3}	3.83×10^{1}	$\mathbf{2.47 \times 10^{1}}$	3.55×10^{1}	1.40×10^{3}	1.53×10^{3}
CEC17(F12)	Ave	4.90×10^{4}	3.18×10^{4}	1.92×10^{4}	4.44×10^{4}	2.75×10^{4}	7.55×10^{3}	4.29×10^{6}	$\mathbf{6.62 \times 10^{3}}$	6.07×10^{4}	1.43×10^{5}	3.52×10^{6}	1.89×10^{10}
	Std	3.33×10^{4}	3.32×10^{4}	1.35×10^{4}	6.48×10^{4}	1.38×10^{4}	7.17×10^{3}	3.34×10^{6}	$\mathbf{4.59 \times 10^{3}}$	6.88×10^{4}	9.98×10^{4}	2.19×10^{6}	4.59×10^{9}
CEC17(F13)	Ave	1.23×10^{4}	4.00×10^{3}	1.96×10^{2}	1.40×10^{4}	9.39×10^{2}	$\mathbf{8.24 \times 10^{1}}$	2.54×10^{6}	9.83×10^{1}	6.78×10^{3}	1.49×10^{4}	1.51×10^{6}	1.23×10^{10}
	Std	1.11×10^{4}	2.67×10^{3}	1.53×10^{2}	1.41×10^{4}	2.79×10^{2}	$\mathbf{9.84 \times 10^{0}}$	2.22×10^{6}	3.42×10^{1}	3.23×10^{3}	1.63×10^{4}	4.77×10^{5}	7.17×10^{9}
CEC17(F14)	Ave	5.27×10^{3}	2.88×10^{2}	1.27×10^{2}	2.99×10^{3}	8.72×10^{1}	6.33×10^{1}	9.85×10^{5}	$\mathbf{5.69 \times 10^{1}}$	2.22×10^{3}	1.05×10^{4}	8.23×10^{5}	9.44×10^{6}
	Std	6.03×10^{3}	1.21×10^{2}	3.56×10^{1}	2.58×10^{3}	9.94×10^{0}	$\mathbf{4.72 \times 10^{0}}$	9.57×10^{5}	5.49×10^{0}	1.39×10^{3}	8.32×10^{3}	8.54×10^{5}	2.25×10^{7}
CEC17(F15)	Ave	4.98×10^{3}	5.94×10^{2}	1.49×10^{2}	5.17×10^{3}	1.52×10^{2}	3.91×10^{1}	7.21×10^{5}	$\mathbf{1.45 \times 10^{1}}$	4.22×10^{2}	7.96×10^{3}	7.01×10^{5}	3.56×10^{9}
	Std	5.84×10^{3}	3.16×10^{2}	5.10×10^{1}	6.89×10^{3}	2.55×10^{1}	$\mathbf{5.79 \times 10^{0}}$	2.53×10^{5}	1.49×10^{1}	5.61×10^{2}	8.62×10^{3}	3.26×10^{5}	2.14×10^{9}
CEC17(F16)	Ave	4.39×10^{2}	9.05×10^{2}	5.53×10^{2}	5.67×10^{2}	$\mathbf{4.41 \times 10^{2}}$	7.78×10^{2}	1.17×10^{3}	7.96×10^{2}	5.52×10^{2}	8.59×10^{2}	8.95×10^{2}	2.90×10^{3}
	Std	1.87×10^{2}	2.772×10^{2}	2.29×10^{2}	2.18×10^{2}	1.58×10^{2}	4.11×10^{2}	3.00×10^{2}	1.94×10^{2}	$\mathbf{1.19 \times 10^{2}}$	2.38×10^{2}	3.11×10^{2}	2.10×10^{2}
CEC17(F17)	Ave	2.61×10^{2}	4.09×10^{2}	1.80×10^{2}	2.00×10^{2}	1.05×10^{2}	1.02×10^{2}	6.43×10^{2}	1.89×10^{2}	$\mathbf{8.49 \times 10^{2}}$	3.57×10^{2}	3.81×10^{2}	1.28×10^{3}

Table 7 continued

Function		mRIME	RIME	WSO	TLBO	SFS	DE	GA	GSK	AMO	PSO	BBO	ACO
	Std	1.33×10^{2}	1.42×10^{2}	8.08×10^{1}	9.15×10^{1}	6.29×10^{1}	5.03×10^{1}	2.04×10^{2}	9.44×10^{1}	$\mathbf{1.77 \times 10^{1}}$	1.64×10^{2}	2.16×10^{2}	1.66×10^{2}
CEC17(F18)	Ave	2.34×10^{5}	2.09×10^{3}	3.45×10^{2}	2.05×10^{5}	3.32×10^{2}	3.83×10^{1}	3.77×10^{6}	$\mathbf{3.68 \times 10^{1}}$	1.43×10^{5}	1.84×10^{5}	1.95×10^{6}	1.78×10^{8}
	Std	1.43×10^{5}	1.62×10^{3}	7.28×10^{2}	1.40×10^{5}	7.40×10^{1}	$\mathbf{4.10 \times 10^{0}}$	4.84×10^{6}	5.42×10^{0}	5.21×10^{4}	1.36×10^{5}	1.96×10^{6}	1.13×10^{8}
CEC17(F19)	Ave	4.81×10^{3}	2.39×10^{2}	7.67×10^{1}	5.57×10^{3}	5.83×10^{1}	1.89×10^{1}	8.32×10^{5}	$\mathbf{1.29 \times 10^{1}}$	1.32×10^{3}	7.77×10^{3}	8.37×10^{5}	3.91×10^{9}
	Std	7.64×10^{3}	1.04×10^{2}	3.08×10^{1}	6.13×10^{3}	9.10×10^{0}	$\mathbf{5.75 \times 10^{0}}$	3.31×10^{5}	6.02×10^{0}	1.53×10^{3}	1.12×10^{4}	3.40×10^{5}	2.60×10^{9}
CEC17(F20)	Ave	2.18×10^{2}	4.22×10^{2}	1.80×10^{2}	2.20×10^{2}	1.30×10^{2}	$\mathbf{6.03 \times 10^{1}}$	4.10×10^{2}	1.08×10^{2}	1.42×10^{2}	3.78×10^{2}	4.33×10^{2}	8.37×10^{2}
	Std	1.27×10^{2}	1.69×10^{2}	7.37×10^{1}	7.33×10^{1}	6.05×10^{1}	6.50×10^{1}	1.11×10^{2}	1.14×10^{2}	$\mathbf{4.76 \times 10^{1}}$	1.36×10^{2}	1.85×10^{2}	1.02×10^{2}
CEC17(F21)	Ave	2.32×10^{2}	2.81×10^{2}	2.73×10^{2}	2.70×10^{2}	2.56×10^{2}	3.66×10^{2}	4.57×10^{2}	3.46×10^{2}	2.53×10^{2}	3.06×10^{2}	$\mathbf{2.49 \times 10^{2}}$	5.90×10^{2}
	Std	7.85×10^{0}	4.87×10^{1}	1.61×10^{1}	1.83×10^{1}	2.71×10^{1}	1.341×10^{1}	3.28×10^{1}	8.30×10^{0}	$\mathbf{6.95 \times 10^{0}}$	2.35×10^{1}	1.04×10^{1}	1.72×10^{1}
CEC17(F22)	Ave	9.91×10^{2}	1.02×10^{2}	1.02×10^{2}	2.08×10^{2}	$\mathbf{1.00 \times 10^{2}}$	1.81×10^{3}	5.16×10^{3}	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$	9.62×10^{2}	1.58×10^{3}	5.74×10^{3}
	Std	9.26×10^{2}	1.88×10^{0}	2.86×10^{0}	7.55×10^{2}	5.50×10^{-4}	2.95×10^{3}	1.60×10^{3}	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	1.61×10^{3}	1.49×10^{3}	4.29×10^{2}
CEC17(F23)	Ave	3.97×10^{2}	4.58×10^{2}	$\mathbf{3.12 \times 10^{2}}$	4.42×10^{2}	4.08×10^{2}	5.26×10^{2}	6.77×10^{2}	4.70×10^{2}	3.98×10^{2}	5.08×10^{2}	4.02×10^{2}	9.40×10^{2}
	Std	7.77×10^{0}	2.81×10^{1}	3.22×10^{1}	2.62×10^{1}	1.35×10^{1}	9.51×19^{0}	3.70×10^{1}	4.49×10^{1}	$\mathbf{9.02 \times 10^{0}}$	5.60×10^{1}	1.24×10^{1}	4.40×10^{1}
CEC17(F24)	Ave	4.60×10^{2}	5.35×10^{2}	5.02×10^{2}	4.97×10^{2}	4.93×10^{2}	5.96×10^{2}	8.14×10^{2}	5.68×10^{2}	$\mathbf{4.64 \times 10^{2}}$	5.73×10^{2}	4.67×10^{2}	1.06×10^{3}

Table 7 continued

Function		mRIME	RIME	WSO	TLBO	SFS	DE	GA	GSK	AMO	PSO	BBO	ACO
	Std	1.09×10^{1}	3.58×10^{1}	4.53×10^{1}	2.37×10^{1}	2.27×10^{1}	$\mathbf{7.08 \times 10^{0}}$	6.40×10^{1}	1.45×10^{1}	8.44×10^{0}	6.32×10^{1}	1.33×10^{1}	5.58×10^{1}
CEC17(F25)	Ave	3.87×10^{2}	3.89×10^{2}	$\mathbf{3.86 \times 10^{2}}$	4.08×10^{2}	$\mathbf{3.86 \times 10^{2}}$	3.87×10^{2}	5.61×10^{2}	3.87×10^{2}	3.87×10^{2}	3.90×10^{2}	3.93×10^{2}	3.49×10^{3}
	Std	1.33×10^{0}	8.27×10^{0}	2.03×10^{1}	2.28×10^{1}	2.68×10^{0}	$\mathbf{2.19 \times 10^{-2}}$	9.50×10^{1}	2.11×10^{-1}	1.07×10^{0}	8.76×10^{0}	9.81×10^{0}	6.51×10^{2}
CEC17(F26)	Ave	1.57×10^{3}	1.66×10^{3}	1.52×10^{3}	2.10×10^{3}	1.32×10^{3}	2.58×10^{3}	3.61×10^{3}	9.87×10^{2}	1.49×10^{3}	$\mathbf{1.27 \times 10^{3}}$	1.63×10^{3}	7.10×10^{3}
	Std	1.26×10^{2}	8.59×10^{2}	1.16×10^{3}	1.08×10^{3}	8.12×10^{2}	2.61×10^{2}	8.15×10^{2}	2.49×10^{2}	1.95×10^{2}	1.39×10^{3}	$\mathbf{1.56 \times 10^{2}}$	4.27×10^{2}
CEC17(F27)	Ave	5.34×10^{2}	5.42×10^{2}	5.74×10^{2}	5.34×10^{2}	5.16×10^{2}	4.96×10^{2}	6.13×10^{2}	$\mathbf{4.93 \times 10^{2}}$	5.16×10^{2}	5.43×10^{2}	5.26×10^{2}	1.14×10^{3}
	Std	1.09×10^{1}	2.08×10^{1}	2.97×10^{1}	2.02×10^{1}	1.25×10^{1}	7.01×10^{0}	4.04×10^{1}	8.02×10^{0}	$\mathbf{4.81 \times 10^{0}}$	3.02×10^{1}	7.66×10^{0}	7.37×10^{1}
CEC17(f28)	Ave	4.14×10^{2}	3.69×10^{2}	3.23×10^{2}	3.85×10^{2}	3.61×10^{2}	3.20×10^{2}	5.66×10^{2}	3.21×10^{2}	$\mathbf{3.14 \times 10^{2}}$	4.14×10^{2}	4.33×10^{2}	3.51×10^{3}
	Std	1.54×10^{1}	6.80×10^{1}	4.45×10^{1}	5.63×10^{1}	4.99×10^{1}	4.41×10^{1}	4.52×10^{1}	4.22×10^{1}	3.61×10^{1}	2.58×10^{1}	$\mathbf{2.47 \times 10^{1}}$	4.77×10^{2}
CEC17(F29)	Ave	6.42×10^{2}	1.02×10^{3}	5.66×10^{2}	8.37×10^{2}	5.53×10^{2}	5.42×10^{2}	9.07×10^{2}	5.77×10^{2}	$\mathbf{5.33 \times 10^{2}}$	7.56×10^{2}	7.18×10^{2}	2.59×10^{3}
	Std	1.39×10^{2}	2.62×10^{2}	1.01×10^{2}	1.80×10^{2}	1.04×10^{2}	9.62×10^{1}	1.78×10^{2}	1.01×10^{2}	$\mathbf{2.46 \times 10^{1}}$	1.79×10^{2}	1.45×10^{2}	2.20×10^{2}
CEC17(F30)	Ave	6.08×10^{3}	4.94×10^{3}	2.95×10^{3}	5.24×10^{3}	8.22×10^{3}	$\mathbf{2.00 \times 10^{3}}$	3.45×10^{5}	2.08×10^{3}	4.71×10^{3}	5.95×10^{3}	3.31×10^{5}	3.16×10^{9}
	Std	2.93×10^{3}	3.01×10^{3}	1.56×10^{3}	2.76×10^{3}	3.40×10^{3}	$\mathbf{5.62 \times 10^{1}}$	1.43×10^{5}	9.27×10^{1}	7.99×10^{2}	2.82×10^{3}	1.23×10^{5}	9.12×10^{8}
Ranking (W|T|L)		0|0|29	0|0|29	2|1|26	0|0|29	1|2|26	3|3|23	0|0|29	8|3|18	5|2|22	1|0|28	4|0|25	0|0|29

Table 8 Results of 51 separate runs using the 50 variable, 500,000 function evaluation CEC2017 test functions to compare mRIME to the original RIME and other techniques for optimization

Function		mRIME	RIME	WSO	TLBO	SFS	DE	GA	GSK	AMO	PSO	BBO	ACO
CEC17(F1)	Ave	2.68×10^{4}	1.15×10^{4}	2.29×10^{1}	2.56×10^{3}	3.65×10^{3}	$\mathbf{7.43 \times 10^{-1}}$	1.93×10^{7}	1.09×10^{3}	1.30×10^{3}	3.87×10^{3}	6.10×10^{6}	1.94×10^{11}
	Std	7.19×10^{4}	1.15×10^{4}	3.73×10^{1}	3.07×10^{3}	5.38×10^{3}	$\mathbf{1.81 \times 10^{0}}$	4.99×10^{6}	1.24×10^{3}	1.27×10^{3}	6.33×10^{3}	7.65×10^{5}	5.32×10^{10}
CEC17(F3)	Ave	5.71×10^{4}	1.24×10^{-5}	$\mathbf{1.97 \times 10^{2}}$	1.40×10^{3}	4.44×10^{3}	9.29×10^{4}	8.00×10^{4}	3.85×10^{3}	3.15×10^{4}	1.90×10^{3}	1.23×10^{5}	1.31×10^{6}
	Std	1.14×10^{4}	4.02×10^{-5}	$\mathbf{2.18 \times 10^{1}}$	1.03×10^{3}	1.28×10^{3}	1.68×10^{4}	1.78×10^{4}	1.51×10^{3}	4.71×10^{3}	3.35×10^{2}	4.22×10^{4}	4.92×10^{6}
CEC17(F4)	Ave	1.17×10^{2}	9.32×10^{1}	$\mathbf{6.59 \times 10^{1}}$	1.03×10^{2}	1.10×10^{2}	8.03×10^{1}	3.03×10^{2}	8.33×10^{1}	7.53×10^{1}	1.57×10^{2}	1.43×10^{2}	3.28×10^{4}
	Std	4.94×10^{1}	5.96×10^{1}	$\mathbf{4.30 \times 10^{1}}$	4.48×10^{1}	5.13×10^{1}	4.94×10^{1}	8.39×10^{1}	5.00×10^{1}	4.77×10^{1}	4.67×10^{1}	5.04×10^{1}	3.36×10^{3}
CEC17(F5)	Ave	6.69×10^{1}	1.99×10^{2}	2.05×10^{2}	1.87×10^{2}	2.13×10^{2}	3.50×10^{2}	4.47×10^{2}	3.20×10^{2}	1.34×10^{2}	2.31×10^{2}	$\mathbf{8.37 \times 10^{1}}$	8.46×10^{2}
	Std	8.44×10^{0}	4.92×10^{1}	3.65×10^{1}	3.14×10^{1}	4.68×10^{1}	1.40×10^{1}	5.07×10^{1}	1.79×10^{1}	$\mathbf{1.30 \times 10^{1}}$	4.15×10^{1}	1.39×10^{1}	2.80×10^{1}
CEC17(F6)	Ave	2.88×10^{-1}	1.84×10^{1}	4.80×10^{0}	2.17×10^{1}	1.01×10^{-1}	$\mathbf{2.60 \times 10^{-7}}$	4.11×10^{1}	3.78×10^{-6}	2.10×10^{-4}	1.55×10^{1}	8.96×10^{-1}	1.05×10^{2}
	Std	7.50×10^{-2}	4.56×10^{0}	3.02×10^{0}	5.12×10^{0}	1.99×10^{-1}	$\mathbf{1.09 \times 10^{-6}}$	4.95×10^{0}	3.52×10^{-6}	1.22×10^{-3}	1.18×10^{1}	3.65×10^{-2}	4.46×10^{0}
CEC17(F7)	Ave	1.29×10^{2}	4.30×10^{2}	4.23×10^{2}	3.84×10^{2}	2.56×10^{2}	4.07×10^{2}	5.87×10^{2}	3.70×10^{2}	2.00×10^{2}	$\mathbf{1.99 \times 10^{2}}$	2.30×10^{2}	2.05×10^{3}
	Std	1.50×10^{1}	8.44×10^{1}	8.17×10^{1}	6.37×10^{1}	5.48×10^{1}	$\mathbf{1.09 \times 10^{1}}$	6.93×10^{1}	1.41×10^{1}	1.64×10^{1}	2.83×10^{1}	2.38×10^{1}	1.71×10^{2}
CEC17(F8)	Ave	6.96×10^{1}	2.04×10^{2}	2.00×10^{2}	2.02×10^{2}	2.03×10^{2}	3.53×10^{2}	4.53×10^{2}	3.24×10^{2}	1.34×10^{2}	2.33×10^{2}	$\mathbf{8.71 \times 10^{1}}$	8.06×10^{2}
	Std	1.47×10^{1}	4.98×10^{1}	2.57×10^{1}	2.96×10^{1}	4.35×10^{1}	1.42×10^{1}	5.51×10^{1}	1.36×10^{1}	$\mathbf{1.35 \times 10^{1}}$	3.73×10^{1}	1.82×10^{1}	2.61×10^{1}
CEC17(F9)	Ave	1.08×10^{1}	4.98×10^{3}	4.88×10^{3}	2.68×10^{3}	2.90×10^{2}	3.99×10^{-2}	3.96×10^{3}	$\mathbf{1.07 \times 10^{-2}}$	3.94×10^{0}	6.53×10^{3}	5.69×10^{2}	4.52×10^{4}
	Std	4.09×10^{1}	1.62×10^{3}	9.58×10^{2}	1.47×10^{3}	4.52×10^{2}	1.26×10^{-1}	2.17×10^{3}	$\mathbf{2.79 \times 10^{-2}}$	4.75×10^{0}	2.49×10^{3}	3.49×10^{2}	5.03×10^{3}
CEC17(F10)	Ave	3.58×10^{3}	6.93×10^{3}	5.19×10^{3}	1.02×10^{4}	4.68×10^{3}	1.30×10^{4}	8.76×10^{3}	1.30×10^{4}	6.96×10^{3}	5.56×10^{3}	$\mathbf{4.11 \times 10^{3}}$	1.36×10^{4}
	Std	1.10×10^{3}	8.64×10^{2}	7.09×10^{2}	2.67×10^{3}	6.67×10^{2}	7.47×10^{2}	7.41×10^{2}	4.50×10^{2}	4.28×10^{2}	8.76×10^{2}	7.06×10^{2}	$\mathbf{3.20 \times 10^{2}}$
CEC17(F11)	Ave	1.34×10^{2}	4.02×10^{2}	2.40×10^{2}	2.16×10^{2}	1.28×10^{2}	1.43×10^{2}	6.84×10^{3}	$\mathbf{3.45 \times 10^{1}}$	9.62×10^{1}	1.63×10^{2}	3.99×10^{3}	2.00×10^{4}
	Std	3.83×10^{1}	1.01×10^{2}	3.53×10^{1}	6.69×10^{1}	3.60×10^{1}	2.27×10^{1}	4.28×10^{3}	2.32×10^{1}	$\mathbf{1.51 \times 10^{1}}$	3.71×10^{1}	3.26×10^{3}	3.07×10^{3}
CEC17(F12)	Ave	$1.54 \times^{6}$	1.24×10^{6}	9.78×10^{4}	5.90×10^{5}	3.19×10^{5}	6.19×10^{4}	1.65×10^{7}	$\mathbf{9.46 \times 10^{3}}$	4.74×10^{5}	1.87×10^{6}	1.10×10^{7}	1.11×10^{11}

Table 8 continued

Function		mRIME	RIME	WSO	TLBO	SFS	DE	GA	GSK	AMO	PSO	BBO	ACO
	Std	1.94×10^{6}	1.43×10^{6}	4.47×10^{4}	1.31×10^{6}	2.42×10^{5}	3.75×10^{4}	8.55×10^{6}	$\mathbf{7.01 \times 10^{3}}$	2.23×10^{5}	1.32×10^{6}	4.55×10^{6}	1.89×10^{10}
CEC17(F13)	Ave	7.38×10^{3}	2.58×10^{4}	3.36×10^{3}	5.23×10^{3}	3.50×10^{3}	$\mathbf{5.33 \times 10^{2}}$	2.17×10^{6}	1.49×10^{3}	1.45×10^{3}	3.93×10^{3}	2.12×10^{6}	6.60×10^{10}
	Std	5.78×10^{3}	1.24×10^{4}	2.24×10^{3}	3.91×10^{3}	3.20×10^{3}	1.39×10^{3}	5.56×10^{5}	2.16×10^{3}	$\mathbf{1.00 \times 10^{3}}$	4.85×10^{3}	7.10×10^{5}	1.62×10^{10}
CEC17(F14)	Ave	3.43×10^{4}	5.73×10^{2}	$\mathbf{1.17 \times 10^{2}}$	4.92×10^{4}	1.68×10^{2}	1.25×10^{2}	2.22×10^{6}	1.24×10^{2}	3.45×10^{4}	5.32×10^{4}	3.70×10^{6}	8.63×10^{7}
	Std	1.61×10^{4}	1.73×10^{2}	5.54×10^{1}	4.29×10^{4}	1.53×10^{1}	$\mathbf{9.35 \times 10^{0}}$	1.10×10^{6}	1.87×10^{1}	1.68×10^{4}	3.68×10^{4}	2.67×10^{6}	3.34×10^{7}
CEC17(F15)	Ave	7.18×10^{3}	5.08×10^{3}	2.95×10^{2}	6.65×10^{3}	3.00×10^{2}	1.08×10^{2}	1.59×10^{6}	$\mathbf{4.20 \times 10^{1}}$	2.24×10^{3}	5.16×10^{3}	1.49×10^{6}	2.04×10^{10}
	Std	6.77×10^{3}	4.90×10^{3}	2.29×10^{2}	5.53×10^{3}	4.76×10^{1}	$\mathbf{1.00 \times 10^{1}}$	3.13×10^{5}	1.68×10^{1}	1.63×10^{3}	4.69×10^{3}	4.43×10^{5}	7.77×10^{9}
CEC17(F16)	Ave	9.61×10^{2}	1.81×10^{3}	1.33×10^{3}	1.17×10^{3}	1.18×10^{3}	2.51×10^{3}	2.67×10^{3}	1.83×10^{3}	$\mathbf{1.05 \times 10^{3}}$	1.50×10^{3}	1.70×10^{3}	5.80×10^{3}
	Std	3.09×10^{2}	3.80×10^{2}	5.51×10^{2}	3.09×10^{2}	3.37×10^{2}	6.16×10^{2}	5.68×10^{2}	6.59×10^{2}	$\mathbf{1.69 \times 10^{2}}$	3.84×10^{2}	3.68×10^{2}	3.21×10^{2}
CEC17(F17)	Ave	5.70×10^{2}	1.42×10^{3}	1.09×10^{3}	9.96×10^{2}	7.49×10^{2}	1.19×10^{3}	1.26×10^{3}	1.35×10^{3}	$\mathbf{7.37 \times 10^{2}}$	1.14×10^{3}	1.20×10^{3}	5.14×10^{3}
	Std	1.81×10^{2}	3.30×10^{2}	2.45×10^{2}	2.23×10^{2}	1.96×10^{2}	4.77×10^{2}	3.48×10^{2}	1.90×10^{2}	$\mathbf{1.11 \times 10^{2}}$	2.81×10^{2}	3.13×10^{2}	6.89×10^{2}
CEC17(F18)	Ave	7.72×10^{5}	3.40×10^{4}	1.25×10^{3}	5.45×10^{5}	$\mathbf{4.53 \times 10^{2}}$	7.67×10^{2}	2.72×10^{6}	5.98×10^{2}	6.43×10^{5}	1.02×10^{6}	8.01×10^{6}	5.13×10^{8}
	Std	6.43×10^{5}	3.30×10^{4}	5.74×10^{2}	3.28×10^{5}	$\mathbf{1.55 \times 10^{2}}$	1.31×10^{3}	2.05×10^{6}	3.37×10^{2}	2.78×10^{5}	5.61×10^{5}	5.70×10^{6}	5.72×10^{8}
CEC17(F19)	Ave	1.48×10^{4}	6.70×10^{2}	1.54×10^{3}	1.32×10^{4}	1.22×10^{2}	6.24×10^{1}	5.69×10^{5}	$\mathbf{3.05 \times 10^{1}}$	1.01×10^{4}	1.36×10^{4}	6.51×10^{5}	9.17×10^{9}
	Std	9.29×10^{3}	2.86×10^{2}	1.25×10^{3}	7.59×10^{3}	2.92×10^{1}	$\mathbf{6.16 \times 10^{0}}$	2.18×10^{5}	9.59×10^{0}	3.71×10^{3}	7.22×10^{3}	1.96×10^{5}	2.48×10^{9}
CEC17(F20)	Ave	4.60×10^{2}	9.84×10^{2}	5.10×10^{2}	5.81×10^{2}	$\mathbf{5.01 \times 10^{2}}$	9.98×10^{2}	1.52×10^{3}	1.37×10^{3}	5.02×10^{2}	9.06×10^{2}	1.16×10^{3}	2.05×10^{3}
	Std	2.45×10^{2}	1.97×10^{2}	1.85×10^{2}	2.76×10^{2}	2.20×10^{2}	5.60×10^{2}	2.68×10^{2}	1.28×10^{2}	$\mathbf{9.84 \times 10^{1}}$	2.68×10^{2}	3.44×10^{2}	1.01×10^{2}
CEC17(F21)	Ave	2.66×10^{2}	3.89×10^{2}	5.06×10^{2}	3.91×10^{2}	3.47×10^{2}	5.52×10^{2}	7.20×10^{2}	5.21×10^{2}	3.27×10^{2}	4.33×10^{2}	$\mathbf{2.96 \times 10^{2}}$	1.04×10^{3}
	Std	1.20×10^{1}	3.30×10^{1}	7.60×10^{1}	3.70×10^{1}	3.36×10^{1}	$\mathbf{1.28 \times 10^{1}}$	6.23×10^{1}	1.31×10^{1}	1.58×10^{1}	4.96×10^{1}	1.58×10^{1}	3.00×10^{1}

Table 8 continued

Function		mRIME	RIME	WSO	TLBO	SFS	DE	GA	GSK	AMO	PSO	BBO	ACO
CEC17(F22)	Ave	4.90×10^{3}	7.40×10^{3}	2.93×10^{3}	7.62×10^{3}	$\mathbf{2.44 \times 10^{3}}$	1.32×10^{4}	1.01×10^{4}	1.10×10^{4}	5.65×10^{3}	6.02×10^{3}	4.92×10^{3}	1.38×10^{4}
	Std	2.35×10^{3}	1.06×10^{3}	1.05×10^{3}	5.62×10^{3}	2.74×10^{3}	$\mathbf{3.25 \times 10^{2}}$	7.97×10^{2}	4.85×10^{3}	3.18×10^{3}	2.58×10^{3}	8.04×10^{2}	3.66×10^{2}
CEC17(F23)	Ave	5.23×10^{2}	6.60×10^{2}	$\mathbf{2.16 \times 10^{2}}$	7.01×10^{2}	5.98×10^{2}	7.69×10^{2}	9.94×10^{2}	5.42×10^{2}	5.60×10^{2}	8.06×10^{2}	5.48×10^{2}	1.74×10^{3}
	Std	2.07×10^{1}	4.75×10^{1}	1.19×10^{2}	6.69×10^{1}	4.60×10^{1}	2.12×10^{1}	4.44×10^{1}	1.39×10^{2}	$\mathbf{1.49 \times 10^{1}}$	1.03×10^{2}	2.41×10^{1}	8.83×10^{1}
CEC17(F24)	Ave	5.80×10^{2}	7.09×10^{2}	1.45×10^{3}	7.25×10^{2}	6.91×10^{2}	8.48×10^{2}	1.27×10^{3}	6.34×10^{2}	6.13×10^{2}	8.83×10^{2}	$\mathbf{5.88 \times 10^{2}}$	1.96×10^{3}
	Std	1.98×10^{1}	4.79×10^{1}	1.36×10^{2}	5.00×10^{1}	3.95×10^{1}	$\mathbf{1.38 \times 10^{1}}$	9.86×10^{1}	1.39×10^{2}	1.50×10^{1}	8.98×10^{1}	1.77×10^{1}	1.08×10^{2}
CEC17(F25)	Ave	5.37×10^{2}	5.34×10^{2}	5.50×10^{2}	5.71×10^{2}	5.60×10^{2}	$\mathbf{4.96 \times 10^{2}}$	6.90×10^{2}	5.56×10^{2}	5.62×10^{2}	5.57×10^{2}	5.75×10^{2}	1.65×10^{4}
	Std	3.31×10^{1}	3.76×10^{1}	$\mathbf{2.32 \times 10^{1}}$	3.42×10^{1}	2.79×10^{1}	3.14×10^{1}	6.08×10^{1}	4.62×10^{1}	2.87×10^{1}	2.37×10^{1}	2.68×10^{1}	1.61×10^{3}
CEC17(F26)	Ave	2.223×10^{2}	3.40×10^{3}	1.39×10^{3}	5.28×10^{3}	1.60×10^{3}	4.40×10^{3}	5.46×10^{3}	$\mathbf{1.27 \times 10^{3}}$	2.54×10^{3}	1.94×10^{3}	2.30×10^{3}	1.57×10^{4}
	Std	2.14×10^{2}	1.12×10^{3}	1.98×10^{3}	2.03×10^{3}	2.23×10^{3}	1.82×10^{2}	5.13×10^{2}	$\mathbf{9.18 \times 10^{1}}$	2.28×10^{2}	2.23×10^{3}	1.77×10^{2}	8.19×10^{2}
CEC17(F27)	Ave	7.48×10^{2}	8.90×10^{2}	$\mathbf{4.82 \times 10^{2}}$	8.98×10^{2}	6.29×10^{2}	5.45×10^{2}	1.17×10^{3}	5.92×10^{2}	6.00×10^{2}	7.51×10^{2}	7.48×10^{2}	2.80×10^{3}
	Std	5.87×10^{1}	1.90×10^{2}	1.11×10^{2}	1.36×10^{2}	5.61×10^{1}	4.05×10^{1}	1.25×10^{2}	8.29×10^{1}	$\mathbf{2.82 \times 10^{1}}$	1.09×10^{2}	7.01×10^{1}	1.80×10^{2}
CEC17(F28)	Ave	5.11×10^{2}	5.03×10^{2}	4.85×10^{2}	5.04×10^{2}	5.08×10^{2}	$\mathbf{4.67 \times 10^{2}}$	1.40×10^{3}	4.94×10^{2}	5.02×10^{2}	5.12×10^{2}	5.26×10^{2}	1.03×10^{4}
	Std	5.06×10^{0}	3.34×10^{1}	3.29×10^{1}	2.40×10^{1}	3.05×10^{1}	1.85×10^{1}	4.65×10^{2}	2.24×10^{1}	2.01×10^{1}	3.21×10^{1}	$\mathbf{1.78 \times 10^{1}}$	6.44×10^{2}
CEC17(F29)	Ave	8.06×10^{2}	2.00×10^{3}	1.54×10^{3}	1.54×10^{3}	8.61×10^{2}	1.18×10^{3}	1.53×10^{3}	$\mathbf{3.60 \times 10^{2}}$	6.58×10^{2}	1.27×10^{3}	1.12×10^{3}	8.78×10^{3}
	Std	1.85×10^{2}	4.69×10^{2}	2.15×10^{2}	3.86×10^{2}	2.33×10^{2}	5.28×10^{2}	3.29×10^{2}	$\mathbf{2.23 \times 10^{1}}$	1.05×10^{2}	2.56×10^{2}	2.31×10^{2}	1.30×10^{3}
CEC17(F30)	Ave	9.65×10^{5}	1.57×10^{6}	6.29×10^{5}	9.35×10^{5}	9.88×10^{5}	$\mathbf{5.91 \times 10^{5}}$	3.09×10^{6}	5.96×10^{5}	8.16×10^{5}	9.06×10^{5}	1.83×10^{6}	1.31×10^{10}
	Std	1.65×10^{5}	5.82×10^{5}	1.14×10^{5}	1.49×10^{5}	1.73×10^{5}	2.40×10^{4}	3.38×10^{6}	$\mathbf{2.24 \times 10^{4}}$	5.64×10^{4}	1.24×10^{5}	3.95×10^{5}	3.91×10^{9}
Ranking (W|T|L)		0|0|29	0|0|29	5|0|24	0|0|29	3|0|26	6|0|23	0|0|29	7|0|22	2|0|27	1|0|28	5|0|24	0|0|29

Table 9 Results of 51 independent runs on the CEC2017 test functions of 100 variables with 1,000,000 function evaluations comparing mRIME to RIME and other techniques for optimization

Function		mRIME	RIME	WSO	TLBO	SFS	DE	GA	GSK	AMO	PSO	BBO	ACO
CEC17(F1)	Ave	8.10×10^{4}	1.58×10^{4}	7.72×10^{3}	7.13×10^{3}	8.57×10^{3}	1.10×10^{4}	2.03×10^{8}	5.80×10^{3}	$\mathbf{2.51 \times 10^{3}}$	1.08×10^{4}	1.28×10^{7}	4.99×10^{11}
	Std	1.36×10^{5}	1.78×10^{4}	2.13×10^{3}	8.54×10^{3}	1.21×10^{4}	1.62×10^{4}	1.62×10^{8}	4.63×10^{3}	$\mathbf{1.89 \times 10^{3}}$	1.42×10^{4}	5.83×10^{5}	1.38×10^{11}
CEC17(F3)	Ave	2.35×10^{5}	3.92×10^{2}	$\mathbf{2.84 \times 10^{3}}$	6.30×10^{4}	3.00×10^{4}	4.23×10^{5}	2.07×10^{5}	1.15×10^{5}	1.75×10^{5}	3.17×10^{4}	4.21×10^{5}	1.32×10^{11}
	Std	2.17×10^{4}	1.05×10^{3}	$\mathbf{4.36 \times 10^{3}}$	1.28×10^{4}	6.89×10^{3}	2.81×10^{4}	3.03×10^{4}	2.15×10^{4}	1.41×10^{4}	5.19×10^{3}	8.31×10^{4}	2.81×10^{11}
CEC17(F4)	Ave	2.49×10^{2}	2.37×10^{2}	2.16×10^{2}	2.70×10^{2}	2.72×10^{2}	2.16×10^{2}	4.76×10^{2}	2.05×10^{2}	$\mathbf{1.41 \times 10^{2}}$	3.03×10^{2}	2.87×10^{2}	1.28×10^{5}
	Std	4.03×10^{1}	3.30×10^{1}	5.50×10^{1}	5.22×10^{1}	3.84×10^{1}	$\mathbf{2.36 \times 10^{1}}$	6.04×10^{1}	4.57×10^{1}	6.04×10^{1}	3.91×10^{1}	4.43×10^{1}	1.09×10^{4}
CEC17(F5)	Ave	1.63×10^{2}	5.04×10^{2}	5.76×10^{2}	5.90×10^{2}	6.36×10^{2}	8.19×10^{2}	1.14×10^{3}	5.31×10^{2}	4.41×10^{2}	6.02×10^{2}	$\mathbf{2.24 \times 10^{2}}$	1.98×10^{3}
	Std	2.18×10^{1}	5.51×10^{1}	6.00×10^{1}	5.32×10^{1}	9.34×10^{1}	$\mathbf{1.77 \times 10^{1}}$	7.63×10^{1}	3.39×10^{2}	4.10×10^{1}	6.13×10^{1}	3.29×10^{1}	3.89×10^{1}
CEC17(F6)	Ave	6.98×10^{-1}	3.47×10^{1}	1.33×10^{1}	4.20×10^{1}	1.76×10^{0}	2.57×10^{-3}	4.24×10^{1}	$\mathbf{1.72 \times 10^{-3}}$	6.28×10^{-2}	3.64×10^{1}	8.79×10^{-1}	1.35×10^{2}
	Std	1.43×10^{-1}	4.29×10^{0}	4.56×10^{0}	3.90×10^{0}	1.94×10^{0}	$\mathbf{4.31 \times 10^{-3}}$	4.85×10^{0}	6.46×10^{-3}	8.93×10^{-2}	8.16×10^{0}	4.51×10^{-2}	3.70×10^{0}
CEC17(F7)	Ave	2.86×10^{2}	1.43×10^{3}	$\mathbf{2.88 \times 10^{2}}$	1.35×10^{3}	9.78×10^{2}	9.39×10^{2}	1.60×10^{3}	8.75×10^{2}	5.86×10^{2}	5.71×10^{2}	6.31×10^{2}	8.86×10^{3}
	Std	3.69×10^{1}	2.84×10^{2}	1.12×10^{2}	2.09×10^{2}	1.76×10^{2}	1.95×10^{1}	1.19×10^{2}	$\mathbf{1.83 \times 10^{1}}$	4.46×10^{1}	1.17×10^{2}	5.66×10^{1}	1.95×10^{2}
CEC17(F8)	Ave	1.60×10^{2}	4.93×10^{2}	6.63×10^{2}	6.31×10^{2}	6.27×10^{2}	8.18×10^{2}	1.08×10^{3}	4.92×10^{2}	4.33×10^{2}	6.01×10^{2}	$\mathbf{2.28 \times 10^{2}}$	2.00×10^{3}
	Std	1.89×10^{1}	7.62×10^{1}	1.28×10^{2}	5.78×10^{1}	9.44×10^{1}	$\mathbf{2.02 \times 10^{1}}$	9.95×10^{1}	3.41×10^{2}	4.24×10^{1}	7.76×10^{1}	3.48×10^{1}	2.18×10^{1}
CEC17(F9)	Ave	1.05×10^{2}	1.93×10^{4}	1.57×10^{4}	2.82×10^{4}	1.18×10^{4}	$\mathbf{3.86 \times 10^{0}}$	3.11×10^{4}	8.46×10^{0}	1.83×10^{3}	1.81×10^{4}	2.25×10^{3}	1.25×10^{5}
	Std	2.37×10^{2}	6.91×10^{3}	1.33×10^{3}	9.68×10^{3}	3.78×10^{3}	4.56×10^{0}	7.46×10^{3}	$\mathbf{3.41 \times 10^{0}}$	1.30×10^{3}	3.09×10^{3}	9.01×10^{2}	7.21×10^{3}
CEC17(F10)	Ave	1.97×10^{4}	1.47×10^{4}	$\mathbf{1.10 \times 10^{4}}$	2.30×10^{4}	1.21×10^{4}	3.01×10^{4}	2.28×10^{4}	2.95×10^{4}	1.81×10^{4}	1.32×10^{4}	1.10×10^{4}	3.01×10^{4}
	Std	5.91×10^{3}	1.52×10^{3}	1.46×10^{3}	6.19×10^{3}	1.33×10^{3}	$\mathbf{3.96 \times 10^{2}}$	1.51×10^{3}	4.44×10^{2}	7.17×10^{2}	1.28×10^{3}	9.31×10^{2}	7.44×10^{2}
CEC17(F11)	Ave	1.04×10^{3}	2.17×10^{3}	5.40×10^{2}	1.16×10^{3}	6.43×10^{2}	6.05×10^{2}	6.12×10^{4}	$\mathbf{2.78 \times 10^{2}}$	6.14×10^{2}	1.12×10^{3}	6.48×10^{4}	1.17×10^{7}

Table 9 continued

Function		mRIME	RIME	WSO	TLBO	SFS	DE	GA	GSK	AMO	PSO	BBO	ACO
	Std	1.39×10^{2}	3.64×10^{2}	1.26×10^{2}	2.76×10^{2}	8.04×10^{1}	8.70×10^{1}	2.52×10^{4}	$\mathbf{6.85 \times 10^{1}}$	7.28×10^{1}	2.00×10^{2}	1.92×10^{4}	3.79×10^{7}
CEC17(F12)	Ave	7.24×10^{6}	4.95×10^{6}	7.11×10^{5}	1.46×10^{6}	2.11×10^{6}	2.72×10^{5}	8.67×10^{7}	$\mathbf{8.34 \times 10^{4}}$	1.34×10^{6}	1.11×10^{7}	4.10×10^{7}	3.32×10^{11}
	Std	5.73×10^{6}	4.38×10^{6}	2.98×10^{5}	9.86×10^{5}	9.27×10^{5}	1.28×10^{5}	3.86×10^{7}	$\mathbf{7.52 \times 10^{4}}$	5.48×10^{5}	5.28×10^{6}	1.51×10^{7}	3.41×10^{10}
CEC17(F13)	Ave	1.20×10^{4}	2.09×10^{4}	5.88×10^{3}	8.96×10^{3}	6.42×10^{3}	5.32×10^{3}	2.75×10^{6}	3.20×10^{3}	$\mathbf{2.37 \times 10^{3}}$	4.22×10^{3}	2.20×10^{6}	8.01×10^{10}
	Std	4.04×10^{3}	1.01×10^{4}	3.30×10^{3}	4.36×10^{3}	6.05×10^{3}	4.39×10^{3}	2.82×10^{6}	2.64×10^{3}	$\mathbf{1.03 \times 10^{3}}$	3.95×10^{3}	3.17×10^{5}	1.35×10^{10}
CEC17(F14)	Ave	4.93×10^{5}	2.82×10^{4}	1.48×10^{4}	1.70×10^{5}	$\mathbf{3.37 \times 10^{2}}$	7.71×10^{3}	9.93×10^{6}	4.64×10^{3}	8.50×10^{5}	5.24×10^{5}	1.54×10^{7}	2.74×10^{8}
	Std	2.43×10^{5}	6.80×10^{4}	4.93×10^{3}	1.96×10^{5}	$\mathbf{4.29 \times 10^{1}}$	8.97×10^{3}	5.46×10^{6}	4.47×10^{3}	3.54×10^{5}	2.42×10^{5}	7.21×10^{6}	2.30×10^{8}
CEC17(F15)	Ave	5.06×10^{3}	1.50×10^{4}	2.76×10^{3}	2.41×10^{3}	3.12×10^{3}	8.91×10^{3}	1.78×10^{6}	7.33×10^{2}	$\mathbf{5.84 \times 10^{2}}$	2.17×10^{3}	1.37×10^{6}	3.65×10^{10}
	Std	4.95×10^{3}	1.11×10^{4}	2.18×10^{3}	2.13×10^{3}	3.95×10^{3}	7.19×10^{3}	4.52×10^{5}	1.09×10^{3}	$\mathbf{3.50 \times 10^{2}}$	1.59×10^{3}	2.99×10^{5}	5.65×10^{9}
CEC17(F16)	Ave	3.10×10^{3}	4.40×10^{3}	3.41×10^{3}	3.36×10^{3}	3.35×10^{3}	7.65×10^{3}	6.22×10^{3}	$\mathbf{2.27 \times 10^{3}}$	3.28×10^{3}	3.31×10^{3}	3.44×10^{3}	1.88×10^{4}
	Std	1.53×10^{3}	8.63×10^{2}	6.04×10^{2}	6.41×10^{2}	6.00×10^{2}	3.37×10^{2}	7.26×10^{2}	2.61×10^{3}	$\mathbf{2.77 \times 10^{2}}$	5.55×10^{2}	7.75×10^{2}	1.35×10^{3}
CEC17(F17)	Ave	2.06×10^{3}	3.79×10^{3}	3.34×10^{3}	3.17×10^{3}	$\mathbf{2.30 \times 10^{3}}$	4.71×10^{3}	3.70×10^{3}	3.91×10^{3}	2.40×10^{3}	2.93×10^{3}	3.20×10^{3}	4.34×10^{6}
	Std	7.28×10^{2}	5.78×10^{2}	4.29×10^{2}	5.95×10^{2}	4.85×10^{2}	4.85×10^{2}	4.74×10^{2}	6.68×10^{2}	$\mathbf{2.29 \times 10^{2}}$	5.54×10^{2}	7.82×10^{2}	2.67×10^{6}
CEC17(F18)	Ave	1.32×10^{6}	7.55×10^{4}	3.85×10^{4}	4.27×10^{5}	$\mathbf{3.18 \times 10^{4}}$	1.21×10^{5}	8.40×10^{6}	5.73×10^{4}	1.60×10^{6}	1.92×10^{6}	6.60×10^{6}	5.51×10^{8}
	Std	1.03×10^{6}	4.62×10^{4}	$\mathbf{1.42 \times 10^{4}}$	2.51×10^{5}	2.32×10^{4}	6.33×10^{4}	4.08×10^{6}	3.63×10^{4}	4.67×10^{5}	8.57×10^{5}	3.01×10^{6}	1.57×10^{8}
CEC17(F19)	Ave	2.82×10^{3}	6.74×10^{3}	1.58×10^{3}	2.36×10^{3}	3.39×10^{3}	8.95×10^{3}	1.30×10^{6}	$\mathbf{1.00 \times 10^{3}}$	1.24×10^{3}	1.93×10^{3}	1.39×10^{6}	3.35×10^{10}
	Std	3.28×10^{3}	9.35×10^{3}	1.63×10^{3}	2.03×10^{3}	4.82×10^{3}	1.05×10^{4}	2.09×10^{5}	$\mathbf{8.30 \times 10^{2}}$	9.31×10^{2}	2.77×10^{3}	2.01×10^{5}	8.97×10^{9}
CEC17(F20)	Ave	2.79×10^{3}	3.14×10^{3}	2.39×10^{3}	2.34×10^{3}	$\mathbf{2.04 \times 10^{3}}$	4.15×10^{3}	3.92×10^{3}	4.46×10^{3}	2.30×10^{3}	2.75×10^{3}	2.74×10^{3}	5.42×10^{3}
	Std	9.45×10^{2}	3.57×10^{2}	5.65×10^{2}	8.22×10^{2}	4.66×10^{2}	8.41×10^{2}	4.42×10^{2}	2.22×10^{2}	2.63×10^{2}	4.15×10^{2}	4.91×10^{2}	$\mathbf{1.66 \times 10^{2}}$
CEC17(F21)	Ave	4.16×10^{2}	7.70×10^{2}	1.25×10^{3}	8.80×10^{2}	7.12×10^{2}	1.04×10^{3}	1.48×10^{3}	6.07×10^{2}	6.23×10^{2}	9.70×10^{2}	$\mathbf{4.78 \times 10^{2}}$	2.62×10^{3}
	Std	1.54×10^{1}	6.28×10^{1}	1.01×10^{2}	8.83×10^{1}	7.84×10^{1}	2.54×10^{1}	1.20×10^{2}	3.37×10^{2}	$\mathbf{2.31 \times 10^{1}}$	1.34×10^{2}	3.30×10^{1}	6.65×10^{1}

Table 9 continued

Function		mRIME	RIME	WSO	TLBO	SFS	DE	GA	GSK	AMO	PSO	BBO	ACO
CEC17(F22)	Ave	1.76×10^{4}	1.64×10^{4}	1.57×10^{4}	2.49×10^{4}	1.30×10^{4}	3.02×10^{4}	2.41×10^{4}	3.00×10^{4}	1.94×10^{4}	1.55×10^{4}	$\mathbf{1.24 \times 10^{4}}$	3.19×10^{4}
	Std	5.50×10^{3}	1.20×10^{3}	2.45×10^{3}	8.31×10^{3}	5.94×10^{3}	7.15×10^{2}	1.07×10^{3}	$\mathbf{4.57 \times 10^{2}}$	7.43×10^{2}	1.75×10^{3}	1.12×10^{3}	5.87×10^{2}
CEC17(F23)	Ave	8.07×10^{2}	1.19×10^{3}	7.51×10^{2}	1.35×10^{3}	9.75×10^{2}	8.60×10^{2}	1.54×10^{3}	$\mathbf{6.11 \times 10^{2}}$	8.22×10^{2}	1.49×10^{3}	7.11×10^{2}	3.91×10^{3}
	Std	4.34×10^{1}	1.48×10^{2}	2.12×10^{2}	1.05×10^{2}	7.94×10^{1}	2.99×10^{2}	1.02×10^{2}	$\mathbf{1.58 \times 10^{1}}$	2.46×10^{0}	1.49×10^{2}	3.06×10^{1}	1.67×10^{2}
CEC17(F24)	Ave	1.19×10^{3}	1.83×10^{3}	2.39×10^{3}	1.99×10^{3}	1.54×10^{3}	1.63×10^{3}	2.11×10^{3}	$\mathbf{9.32 \times 10^{2}}$	1.23×10^{3}	1.60×10^{3}	1.20×10^{3}	7.41×10^{3}
	Std	3.84×10^{1}	3.12×10^{2}	3.16×10^{2}	1.96×10^{2}	1.00×10^{2}	1.49×10^{2}	1.33×10^{2}	$\mathbf{1.70 \times 10^{1}}$	4.73×10^{1}	1.56×10^{2}	4.17×10^{1}	6.15×10^{2}
CEC17(F25)	Ave	8.07×10^{2}	7.81×10^{2}	7.90×10^{2}	8.28×10^{2}	7.90×10^{2}	$\mathbf{7.31 \times 10^{2}}$	1.63×10^{3}	8.21×10^{2}	8.31×10^{2}	8.12×10^{2}	8.03×10^{2}	4.90×10^{4}
	Std	5.21×10^{1}	5.75×10^{1}	5.87×10^{1}	6.11×10^{1}	4.99×10^{1}	5.56×10^{1}	1.63×10^{2}	$\mathbf{4.34 \times 10^{1}}$	5.21×10^{1}	7.29×10^{1}	7.33×10^{1}	5.90×10^{3}
CEC17(F26)	Ave	6.16×10^{3}	1.13×10^{4}	8.18×10^{3}	2.01×10^{4}	1.37×10^{4}	1.08×10^{4}	1.58×10^{4}	$\mathbf{3.66 \times 10^{3}}$	7.91×10^{3}	9.73×10^{3}	6.26×10^{3}	5.23×10^{4}
	Std	4.01×10^{2}	1.35×10^{3}	1.05×10^{3}	5.23×10^{3}	6.56×10^{3}	1.37×10^{3}	1.46×10^{3}	$\mathbf{1.78 \times 10^{2}}$	6.30×10^{2}	5.69×10^{3}	4.03×10^{2}	2.10×10^{3}
CEC17(F27)	Ave	8.10×10^{2}	1.07×10^{3}	8.87×10^{2}	1.42×10^{3}	8.81×10^{2}	$\mathbf{6.16 \times 10^{2}}$	1.27×10^{3}	6.57×10^{2}	8.49×10^{2}	9.31×10^{2}	8.28×10^{2}	7.92×10^{3}
	Std	4.08×10^{1}	1.41×10^{2}	1.44×10^{2}	2.02×10^{2}	7.99×10^{1}	$\mathbf{2.15 \times 10^{1}}$	1.20×10^{2}	3.07×10^{1}	5.06×10^{1}	1.14×10^{2}	6.41×10^{1}	2.58×10^{2}
CEC17(F28)	Ave	6.19×10^{2}	1.17×10^{3}	5.64×10^{2}	6.30×10^{2}	6.13×10^{2}	5.59×10^{2}	2.60×10^{3}	$\mathbf{5.53 \times 10^{2}}$	5.60×10^{2}	6.55×10^{2}	6.44×10^{2}	3.82×10^{4}
	Std	3.28×10^{1}	2.70×10^{3}	8.74×10^{1}	$\mathbf{2.91 \times 10^{1}}$	3.09×10^{1}	3.52×10^{1}	1.62×10^{3}	3.25×10^{1}	3.26×10^{1}	4.30×10^{1}	4.08×10^{1}	1.51×10^{3}
CEC17(F29)	Ave	2.65×10^{3}	4.60×10^{3}	4.20×10^{3}	4.34×10^{3}	3.09×10^{3}	4.74×10^{3}	4.37×10^{3}	$\mathbf{1.21 \times 10^{3}}$	2.74×10^{3}	3.66×10^{3}	3.29×10^{3}	1.19×10^{5}
	Std	3.97×10^{2}	7.84×10^{2}	6.76×10^{2}	6.73×10^{2}	4.26×10^{2}	1.19×10^{3}	4.85×10^{2}	$\mathbf{1.74 \times 10^{2}}$	3.04×10^{2}	5.24×10^{2}	5.09×10^{2}	5.53×10^{4}
CEC17(F30)	Ave	2.01×10^{4}	1.88×10^{5}	1.19×10^{4}	2.15×10^{4}	1.22×10^{4}	4.68×10^{3}	3.17×10^{6}	$\mathbf{2.99 \times 10^{3}}$	5.25×10^{3}	8.19×10^{3}	2.63×10^{6}	6.75×10^{10}
	Std	2.37×10^{4}	2.16×10^{5}	7.74×10^{3}	2.11×10^{4}	6.93×10^{3}	3.05×10^{3}	3.02×10^{5}	$\mathbf{2.73 \times 10^{2}}$	1.28×10^{3}	3.68×10^{3}	4.88×10^{5}	1.47×10^{10}
Ranking (W|T|L)		0|0|29	0|0|29	3|0|26	0|0|29	4|0|25	3|0|26	0|0|29	11|0|18	4|0|25	0|0|29	4|0|25	0|0|29

ExpeRIMEnts series2: applying mRIME for global optimization using CEC2011 test suit functions

Table 11 presents the results of an optimization study using the CEC2011 test functions. The study compares the performance of mRIME (a method under evaluation) with the original RIME method and several other methods across 25 independent runs. The results are reported in terms of Ave and Std of the performance metric for each method.

Based on the results presented in the table, we can make the following observations:

- For some test functions (e.g., F1, F3, F4), mRIME performs better than the original RIME method. It achieves lower average values, indicating improved optimization performance.
- In a few cases (e.g., F2, F5, F6), other methods outperform both mRIME and the original RIME method. The best-performing method varies depending on the test function.
- The standard deviations (Std) provide insights into the consistency of the method's performance across different runs. Smaller Std values suggest more stable and reliable optimization results.
- In some instances (e.g., F3, F4), the standard deviation for mRIME is exceptionally low (0.00E+00), indicating consistent and reproducible results.

Statistical test analysis

Two statistical tests were performed in this section to rank the optimization techniques and determine whether the quality differences among all techniques are statistically significant. Friedman's test is the first one run in this paper [65]. In order to get a reliable comparison using this test, it is necessary to compare more than ten functions, and to compare more than five distinct approaches [66]. This study examined over five distinct approaches in this regard, testing each one on 51 functions. Two test bunches consisting of many test functions of varied complexity are listed as follows: (1) CEC-2017, which has 29 test problems and a total of four different dimensions for each examined problem, 10, 30, 50, and 100, and (2) CEC-2011, which has 22 test functions. In order to use Friedman's test to rank the algorithms, the average ranking value must be determined. The crucial values acquired for the statistically significant level (p-value), which was established as $\alpha = 0.05$, are then compared. This implies that if Friedman's test reveals a significance level of no more than or equal to 0.05, the null assumption is disregarded, which indicates that there is no difference in the accuracy of all the techniques that were tested. According to the alternate hypothesis, there are variations in each of the compared approaches' performances. When Friedman's test is used, the algorithm that scores the lowest is the best; on the other side, the algorithm that scores the highest is the worst.

The optimal algorithm serves as a control strategy for further investigation. The mean error values of all techniques in all of the dimensions and the ranking of every technique in each dimension were considered in order to judge the statistical efficacy of the proposed mRIME algorithm and all other methods on the CEC-2017 test suite. Table 12 summarizes the average ranks of mRIME combined with other techniques using Friedman's method on CEC-2017 with a dimension of 10 for all problems in this set of problems.

Table 10 Results of mRIME optimization compared to original RIME across 25 separate runs on the CEC2011 test functions with 150,000 function evaluations

Function	Best value		Median value		Average value		Worst value		Std	
	RIME	mRIME	RIME	mRIME	RIME	mRIME	RIME	mRIME	RIME	mRIME
CEC11(F1)	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	1.65×10^{1}	$\mathbf{1.23 \times 10^{1}}$	1.61×10^{1}	$\mathbf{1.19 \times 10^{1}}$	2.41×10^{1}	$\mathbf{1.87 \times 10^{1}}$	6.14×10^{0}	$\mathbf{4.75 \times 10^{0}}$
CEC11(F2)	-1.89×10^{1}	$\mathbf{-2.76 \times 10^{1}}$	-1.21×10^{1}	$\mathbf{-2.64 \times 10^{1}}$	-1.27×10^{1}	$\mathbf{-2.54 \times 10^{1}}$	$\mathbf{-9.86 \times 10^{0}}$	-2.20×10^{1}	2.71×10^{0}	$\mathbf{2.07 \times 10^{0}}$
CEC11(F3)	$\mathbf{1.15 \times 10^{-5}}$	$\mathbf{1.15 \times 10^{-5}}$	$\mathbf{1.15 \times 10^{-5}}$	$\mathbf{1.15 \times 10^{-5}}$	$\mathbf{1.15 \times 10^{-5}}$	$\mathbf{1.15 \times 10^{-5}}$	$\mathbf{1.15 \times 10^{-5}}$	$\mathbf{1.15 \times 10^{-5}}$	$\mathbf{1.56 \times 10^{-19}}$	1.89×10^{-19}
CEC11(F4)	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$
CEC11(F5)	-3.54×10^{1}	$\mathbf{-3.68 \times 10^{1}}$	-3.15×10^{1}	$\mathbf{-3.45 \times 10^{1}}$	-3.10×10^{1}	$\mathbf{-3.37 \times 10^{1}}$	-2.52×10^{1}	$\mathbf{-2.79 \times 10^{1}}$	3.39×10^{0}	$\mathbf{2.08 \times 10^{0}}$
CEC11(F6)	$\mathbf{-2.74 \times 10^{1}}$	$\mathbf{-2.74 \times 10^{1}}$	-1.95×10^{1}	$\mathbf{-2.30 \times 10^{1}}$	-2.06×10^{1}	$\mathbf{-2.30 \times 10^{1}}$	-1.68×10^{1}	$\mathbf{-1.95 \times 10^{1}}$	$\mathbf{2.67 \times 10^{0}}$	3.13×10^{0}
CEC11(F7)	1.00×10^{0}	$\mathbf{6.02 \times 10^{-1}}$	1.39×10^{0}	$\mathbf{8.55 \times 10^{-1}}$	1.36×10^{0}	$\mathbf{8.72 \times 10^{-1}}$	1.85×10^{0}	$\mathbf{1.21 \times 10^{0}}$	2.15×10^{-1}	$\mathbf{1.60 \times 10^{-1}}$
CEC11(F8)	$\mathbf{2.20 \times 10^{2}}$	$\mathbf{2.20 \times 10^{2}}$	$\mathbf{2.20 \times 10^{2}}$	$\mathbf{2.20 \times 10^{2}}$	2.27×10^{2}	$\mathbf{2.20 \times 10^{2}}$	2.62×10^{2}	$\mathbf{2.20 \times 10^{2}}$	1.35×10^{1}	$\mathbf{0.00 \times 10^{0}}$
CEC11(F9)	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	8.88×10^{4}	$\mathbf{0.00 \times 10^{0}}$	1.13×10^{5}	$\mathbf{0.00 \times 10^{0}}$	3.10×10^{5}	$\mathbf{0.00 \times 10^{0}}$	9.48×10^{4}	$\mathbf{0.00 \times 10^{0}}$
CEC11(F10)	-1.42×10^{1}	$\mathbf{-2.14 \times 10^{1}}$	-1.01×10^{1}	$\mathbf{-1.27 \times 10^{1}}$	$\mathbf{-9.89 \times 10^{0}}$	-1.50×10^{1}	$\mathbf{-7.82 \times 10^{0}}$	-1.20×10^{1}	$\mathbf{1.35 \times 10^{0}}$	3.63×10^{0}
CEC11(F11)	7.96×10^{4}	$\mathbf{5.12 \times 10^{4}}$	3.56×10^{5}	$\mathbf{5.23 \times 10^{4}}$	3.69×10^{5}	$\mathbf{5.23 \times 10^{4}}$	5.76×10^{5}	$\mathbf{5.33 \times 10^{4}}$	1.25×10^{5}	$\mathbf{5.18 \times 10^{2}}$
CEC11(F12)	2.03×10^{6}	$\mathbf{1.07 \times 10^{6}}$	2.77×10^{6}	$\mathbf{1.08 \times 10^{6}}$	2.85×10^{6}	$\mathbf{1.09 \times 10^{6}}$	3.93×10^{6}	$\mathbf{1.17 \times 10^{6}}$	4.77×10^{5}	$\mathbf{2.09 \times 10^{4}}$

Table 10 continued

Function	Best value		Median value		Average value		Worst value		Std	
	RIME	mRIME	RIME	mRIME	RIME	mRIME	RIME	mRIME	RIME	mRIME
CEC11(F13)	1.55×10^{4}	$\mathbf{1.54 \times 10^{4}}$	1.55×10^{4}	$\mathbf{1.54 \times 10^{4}}$	1.55×10^{4}	$\mathbf{1.54 \times 10^{4}}$	1.56×10^{4}	$\mathbf{1.55 \times 10^{4}}$	2.94×10^{1}	$\mathbf{2.54 \times 10^{0}}$
CEC11(F14)	1.88×10^{4}	$\mathbf{1.82 \times 10^{4}}$	1.92×10^{4}	$\mathbf{1.86 \times 10^{4}}$	1.92×10^{4}	$\mathbf{1.86 \times 10^{4}}$	1.94×10^{4}	$\mathbf{1.89 \times 10^{4}}$	$\mathbf{1.43 \times 10^{2}}$	1.61×10^{2}
CEC11(F15)	3.28×10^{4}	$\mathbf{3.27 \times 10^{4}}$	3.32×10^{4}	$\mathbf{3.28 \times 10^{4}}$	3.31×10^{4}	$\mathbf{3.28 \times 10^{4}}$	3.36×10^{4}	$\mathbf{3.29 \times 10^{4}}$	2.11×10^{2}	$\mathbf{5.42 \times 10^{1}}$
CEC11(F16)	1.31×10^{5}	$\mathbf{1.27 \times 10^{5}}$	1.37×10^{5}	$\mathbf{1.29 \times 10^{5}}$	1.37×10^{5}	$\mathbf{1.29 \times 10^{5}}$	1.47×10^{5}	$\mathbf{1.31 \times 10^{5}}$	4.17×10^{3}	$\mathbf{9.54 \times 10^{2}}$
CEC11(F17)	1.94×10^{6}	$\mathbf{1.90 \times 10^{6}}$	1.98×10^{6}	$\mathbf{1.95 \times 10^{6}}$	2.19×10^{7}	$\mathbf{1.95 \times 10^{6}}$	2.10×10^{8}	$\mathbf{2.00 \times 10^{6}}$	6.25×10^{7}	$\mathbf{3.10 \times 10^{4}}$
CEC11(F18)	$\mathbf{9.41 \times 10^{5}}$	9.42×10^{5}	$\mathbf{9.50 \times 10^{5}}$	9.53×10^{5}	9.89×10^{5}	$\mathbf{9.53 \times 10^{5}}$	1.56×10^{6}	$\mathbf{9.60 \times 10^{5}}$	1.37×10^{5}	$\mathbf{4.46 \times 10^{3}}$
CEC11(F19)	1.02×10^{6}	$\mathbf{9.48 \times 10^{5}}$	1.43×10^{6}	$\mathbf{9.67 \times 10^{5}}$	1.48×10^{6}	$\mathbf{1.05 \times 10^{6}}$	2.26×10^{6}	$\mathbf{2.18 \times 10^{6}}$	3.50×10^{5}	$\mathbf{2.63 \times 10^{5}}$
CEC11(F20)	$\mathbf{9.42 \times 10^{5}}$	9.44×10^{5}	$\mathbf{9.50 \times 10^{5}}$	9.52×10^{5}	1.05×10^{6}	$\mathbf{9.54 \times 10^{5}}$	1.85×10^{6}	$\mathbf{9.66 \times 10^{5}}$	2.27×10^{5}	$\mathbf{6.47 \times 10^{3}}$
CEC11(F21)	1.23×10^{1}	$\mathbf{8.95 \times 10^{0}}$	2.11×10^{1}	$\mathbf{1.69 \times 10^{1}}$	2.28×10^{1}	$\mathbf{1.62 \times 10^{1}}$	3.86×10^{1}	$\mathbf{2.16 \times 10^{1}}$	6.00×10^{0}	$\mathbf{2.97 \times 10^{0}}$
CEC11(F22)	$\mathbf{1.02 \times 10^{1}}$	1.23×10^{1}	2.46×10^{1}	$\mathbf{2.13 \times 10^{1}}$	2.35×10^{1}	$\mathbf{2.09 \times 10^{1}}$	$\mathbf{3.05 \times 10^{1}}$	2.95×10^{1}	$\mathbf{5.37 \times 10^{0}}$	5.91×10^{0}
Ranking (W|T|L)	2|6|14	14|6|20	2|4|16	16|4|2	1|2|19	19|2|1	2|2|18	18|2|2	4|1|17	17|1|4

Table 11 Results of optimization using the CEC2011 test functions with 150,000 feature evaluations, comparing mRIME to the original RIME and other methods across 25 independent runs

Function		mRIME	RIME	WSO	TLBO	SFS	DE	GA	GSK	AMO	PSO	BBO	ACO
CEC11(F1)	Ave	1.61×10^{1}	1.19×10^{1}	6.74×10^{0}	6.60×10^{0}	4.69×10^{0}	4.30×10^{0}	1.95×10^{1}	3.28×10^{0}	$\mathbf{4.85 \times 10^{-1}}$	2.63×10^{1}	2.05×10^{1}	3.05×10^{1}
	Std	6.14×10^{0}	4.75×10^{0}	6.35×10^{0}	6.78×10^{0}	4.49×10^{0}	5.38×10^{0}	3.01×10^{0}	5.21×10^{0}	1.17×10^{0}	$\mathbf{0.00 \times 10^{0}}$	3.40×10^{0}	8.23×10^{-1}
CEC11(F2)	Ave	-1.27×10^{1}	-2.54×10^{1}	$\mathbf{-2.69 \times 10^{1}}$	-2.08×10^{1}	-2.66×10^{1}	-1.31×10^{1}	-9.87×10^{0}	-1.13×10^{1}	-1.98×10^{1}	-3.79×10^{0}	-1.85×10^{1}	-6.16×10^{0}
	Std	2.71×10^{0}	2.07×10^{0}	6.96×10^{-1}	2.15×10^{0}	1.28×10^{0}	4.60×10^{0}	2.20×10^{0}	1.03×10^{0}	1.03×10^{0}	$\mathbf{5.52 \times 10^{-2}}$	1.85×10^{0}	8.28×10^{-1}
CEC11(F3)	Ave	1.15×10^{-5}	1.15×10^{-5}	$\mathbf{1.15 \times 10^{-5}}$	$\mathbf{1.15 \times 10^{-5}}$	$\mathbf{1.15 \times 10^{-5}}$	$\mathbf{1.15 \times 10^{-5}}$	$\mathbf{1.15 \times 10^{-5}}$	$\mathbf{1.15 \times 10^{-5}}$	$\mathbf{1.15 \times 10^{-5}}$	$\mathbf{1.15 \times 10^{-5}}$	$\mathbf{1.15 \times 10^{-5}}$	$\mathbf{1.15 \times 10^{-5}}$
	Std	1.56×10^{-19}	1.89×10^{-19}	7.80×10^{-20}	4.08×10^{-17}	4.87×10^{-10}	1.86×10^{-15}	9.88×10^{-10}	9.12×10^{-13}	7.25×10^{-14}	1.46×10^{-12}	3.80×10^{-9}	$\mathbf{0.00 \times 10^{0}}$
CEC11(F4)	Ave	0.00×10^{0}	0.00×10^{0}	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$
	Std	0.00×10^{0}	0.00×10^{0}	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$
CEC11(F5)	Ave	-3.10×10^{1}	-3.37×10^{1}	$\mathbf{-3.42 \times 10^{1}}$	-2.83×10^{1}	-3.37×10^{1}	-1.95×10^{1}	-1.38×10^{1}	-2.06×10^{1}	-3.32×10^{1}	-1.98×10^{1}	-3.30×10^{1}	-1.48×10^{1}
	Std	3.39×10^{0}	2.08×10^{0}	2.56×10^{0}	3.70×10^{0}	1.61×10^{0}	9.34×10^{-1}	1.94×10^{0}	1.21×10^{0}	$\mathbf{7.33 \times 10^{-1}}$	9.91×10^{-1}	7.34×10^{-1}	1.21×10^{0}
CEC11(F6)	Ave	-2.06×10^{1}	-2.30×10^{1}	-2.31×10^{1}	-1.94×10^{1}	-2.74×10^{1}	-1.41×10^{1}	-3.35×10^{0}	-6.94×10^{0}	-2.70×10^{1}	-1.06×10^{1}	$\mathbf{-2.79 \times 10^{1}}$	-1.04×10^{1}
	Std	2.67×10^{0}	3.13×10^{0}	3.37×10^{0}	1.70×10^{0}	1.86×10^{0}	1.53×10^{0}	3.99×10^{0}	2.48×10^{0}	9.45×10^{-1}	1.03×10^{0}	$\mathbf{7.95 \times 10^{-1}}$	1.42×10^{0}
CEC11(F7)	Ave	1.36×10^{0}	8.72×10^{-1}	$\mathbf{9.20 \times 10^{-1}}$	1.16×10^{0}	1.36×10^{0}	1.75×10^{0}	1.13×10^{0}	1.78×10^{0}	1.37×10^{0}	1.53×10^{0}	1.44×10^{0}	1.89×10^{0}
	Std	2.15×10^{-1}	1.60×10^{-1}	1.70×10^{-1}	2.69×10^{-1}	1.39×10^{-1}	9.79×10^{-2}	1.61×10^{-1}	1.08×10^{-1}	9.38×10^{-2}	1.21×10^{-1}	$\mathbf{8.64 \times 10^{-2}}$	9.89×10^{-2}
CEC11(F8)	Ave	2.27×10^{2}	2.20×10^{2}	$\mathbf{2.20 \times 10^{2}}$	$\mathbf{2.20 \times 10^{2}}$	$\mathbf{2.20 \times 10^{2}}$	$\mathbf{2.20 \times 10^{2}}$	$\mathbf{2.20 \times 10^{2}}$	$\mathbf{2.20 \times 10^{2}}$	$\mathbf{2.20 \times 10^{2}}$	2.21×10^{2}	$\mathbf{2.20 \times 10^{2}}$	2.39×10^{3}
	Std	1.35×10^{1}	0.00×10^{0}	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	2.84×10^{0}	$\mathbf{0.00 \times 10^{0}}$	1.32×10^{3}
CEC11(F9)	Ave	1.13×10^{5}	0.00×10^{0}	2.96×10^{4}	3.99×10^{3}	2.27×10^{4}	2.37×10^{4}	1.32×10^{5}	2.11×10^{3}	$\mathbf{1.15 \times 10^{3}}$	1.85×10^{6}	7.83×10^{4}	1.03×10^{6}
	Std	9.48×10^{4}	0.00×10^{0}	6.54×10^{4}	2.88×10^{3}	6.55×10^{3}	2.35×10^{3}	3.76×10^{4}	5.02×10^{2}	$\mathbf{3.24 \times 10^{2}}$	1.54×10^{5}	2.42×10^{4}	2.42×10^{4}
CEC11(F10)	Ave	-9.89×10^{0}	-1.50×10^{1}	-2.14×10^{1}	-1.89×10^{1}	-2.15×10^{1}	-1.41×10^{1}	-9.61×10^{0}	$\mathbf{-2.16 \times 10^{1}}$	-2.13×10^{1}	-9.31×10^{0}	-1.52×10^{1}	-9.04×10^{0}
	Std	1.35×10^{0}	3.63×10^{0}	5.03×10^{-1}	2.43×10^{0}	1.31×10^{-1}	2.68×10^{0}	1.67×10^{0}	1.19×10^{-1}	$\mathbf{9.12 \times 10^{-2}}$	8.20×10^{-1}	1.78×10^{0}	1.49×10^{-1}
CEC11(F11)	Ave	3.69×10^{5}	5.23×10^{4}	5.26×10^{4}	3.79×10^{6}	4.92×10^{5}	1.26×10^{5}	5.52×10^{5}	$\mathbf{5.24 \times 10^{4}}$	5.26×10^{4}	5.36×10^{6}	5.66×10^{4}	9.25×10^{6}
	Std	1.25×10^{5}	5.18×10^{2}	5.23×10^{2}	9.41×10^{5}	2.14×10^{5}	2.43×10^{4}	8.33×10^{5}	6.88×10^{2}	$\mathbf{5.02 \times 10^{2}}$	3.85×10^{5}	8.38×10^{2}	2.73×10^{6}

Table 11 continued

Function		mRIME	RIME	WSO	TLBO	SFS	DE	GA	GSK	AMO	PSO	BBO	ACO
CEC11(F12)	Ave	2.85×10^{6}	1.09×10^{6}	1.09×10^{6}	1.53×10^{6}	1.30×10^{6}	1.14×10^{6}	1.05×10^{7}	$\mathbf{1.07 \times 10^{6}}$	$\mathbf{1.07 \times 10^{6}}$	1.42×10^{7}	1.09×10^{6}	8.10×10^{6}
	Std	4.77×10^{5}	2.09×10^{4}	1.16×10^{4}	2.23×10^{5}	6.45×10^{4}	9.67×10^{3}	7.73×10^{5}	1.73×10^{3}	$\mathbf{1.27 \times 10^{3}}$	9.63×10^{5}	1.93×10^{4}	2.33×10^{5}
CEC11(F13)	Ave	1.55×10^{4}	1.54×10^{4}	$\mathbf{1.54 \times 10^{4}}$	1.55×10^{4}	$\mathbf{1.54 \times 10^{4}}$	$\mathbf{1.54 \times 10^{4}}$	1.55×10^{4}	$\mathbf{1.54 \times 10^{4}}$	$\mathbf{1.54 \times 10^{4}}$	1.56×10^{4}	1.55×10^{4}	1.29×10^{5}
	Std	2.94×10^{1}	2.54×10^{0}	$\mathbf{1.21 \times 10^{0}}$	1.24×10^{1}	1.44×10^{0}	1.18×10^{1}	2.23×10^{1}	2.44×10^{0}	2.59×10^{0}	5.13×10^{1}	2.49×10^{1}	9.70×10^{4}
CEC11(F14)	Ave	1.92×10^{4}	1.86×10^{4}	$\mathbf{1.82 \times 10^{4}}$	1.93×10^{4}	1.88×10^{4}	1.84×10^{4}	1.98×10^{4}	1.84×10^{4}	1.92×10^{4}	1.97×10^{4}	1.94×10^{4}	4.15×10^{5}
	Std	1.43×10^{2}	1.61×10^{2}	9.12×10^{1}	9.32×10^{1}	$\mathbf{8.04 \times 10^{1}}$	1.66×10^{2}	7.01×10^{2}	1.22×10^{2}	1.48×10^{2}	2.14×10^{2}	3.21×10^{2}	4.79×10^{5}
CEC17(F15)	Ave	3.31×10^{4}	3.28×10^{4}	3.29×10^{4}	3.29×10^{4}	3.30×10^{4}	3.29×10^{4}	3.30×10^{4}	$\mathbf{3.28 \times 10^{4}}$	3.30×10^{4}	1.26×10^{5}	3.31×10^{4}	4.13×10^{6}
	Std	2.11×10^{2}	5.42×10^{1}	8.25×10^{1}	7.14×10^{1}	2.35×10^{1}	3.03×10^{1}	7.93×10^{1}	$\mathbf{1.55 \times 10^{1}}$	2.09×10^{1}	5.23×10^{4}	6.90×10^{1}	3.30×10^{6}
CEC11(F16)	Ave	1.37×10^{5}	1.29×10^{5}	1.37×10^{5}	1.36×10^{5}	1.37×10^{5}	1.37×10^{5}	1.50×10^{5}	$\mathbf{1.35 \times 10^{5}}$	1.37×10^{5}	4.76×10^{6}	1.42×10^{5}	7.43×10^{7}
	Std	4.17×10^{3}	9.54×10^{2}	4.50×10^{3}	3.40×10^{3}	2.16×10^{3}	2.91×10^{3}	7.18×10^{3}	2.22×10^{3}	$\mathbf{1.70 \times 10^{3}}$	4.24×10^{6}	4.91×10^{3}	1.77×10^{7}
CEC11(F17)	Ave	2.19×10^{7}	1.95×10^{6}	2.11×10^{6}	2.05×10^{6}	2.21×10^{6}	2.26×10^{6}	8.78×10^{8}	2.09×10^{6}	$\mathbf{2.02 \times 10^{6}}$	1.24×10^{10}	2.70×10^{6}	1.78×10^{10}
	Std	6.25×10^{7}	3.10×10^{4}	2.84×10^{5}	2.21×10^{5}	2.60×10^{5}	2.34×10^{5}	1.10×10^{9}	$\mathbf{1.20 \times 10^{5}}$	1.63×10^{5}	2.39×10^{9}	1.98×10^{6}	3.18×10^{9}
CEC11(F18)	Ave	9.89×10^{5}	9.53×10^{5}	1.19×10^{6}	1.14×10^{6}	1.05×10^{6}	1.88×10^{6}	1.32×10^{6}	1.27×10^{6}	1.04×10^{6}	1.29×10^{8}	$\mathbf{9.70 \times 10^{5}}$	1.53×10^{8}
	Std	1.37×10^{5}	4.46×10^{3}	5.59×10^{5}	8.67×10^{4}	4.38×10^{4}	3.06×10^{5}	2.70×10^{5}	7.56×10^{4}	7.56×10^{4}	1.15×10^{7}	$\mathbf{1.47 \times 10^{4}}$	1.89×10^{7}
CEC11(F19)	Ave	1.48×10^{6}	1.05×10^{6}	1.50×10^{6}	$\mathbf{1.42 \times 10^{6}}$	1.51×10^{6}	2.52×10^{6}	2.24×10^{6}	2.00×10^{6}	1.56×10^{6}	1.34×10^{8}	1.53×10^{6}	1.47×10^{8}
	Std	3.50×10^{5}	2.63×10^{5}	4.54×10^{5}	1.58×10^{5}	1.93×10^{5}	2.58×10^{5}	2.48×10^{6}	1.36×10^{5}	$\mathbf{1.29 \times 10^{5}}$	1.95×10^{7}	2.56×10^{5}	2.71×10^{7}
CEC11(F20)	Ave	1.05×10^{6}	9.54×10^{5}	1.05×10^{6}	1.12×10^{6}	1.05×10^{6}	1.77×10^{6}	1.37×10^{6}	1.29×10^{6}	1.03×10^{6}	1.37×10^{8}	$\mathbf{9.75 \times 10^{5}}$	1.59×10^{8}
	Std	2.27×10^{5}	6.47×10^{3}	4.37×10^{5}	8.06×10^{4}	4.64×10^{4}	2.78×10^{5}	1.09×10^{6}	9.20×10^{4}	5.94×10^{4}	1.98×10^{7}	$\mathbf{1.54 \times 10^{4}}$	1.44×10^{7}
CEC11(F21)	Ave	2.28×10^{1}	1.62×10^{1}	1.59×10^{1}	$\mathbf{1.55 \times 10^{1}}$	1.77×10^{1}	1.77×10^{1}	3.15×10^{1}	1.70×10^{1}	1.85×10^{1}	6.11×10^{1}	2.24×10^{1}	8.61×10^{1}
	Std	6.00×10^{0}	2.97×10^{0}	2.65×10^{0}	2.02×10^{0}	3.13×10^{0}	3.62×10^{0}	6.07×10^{0}	3.11×10^{0}	$\mathbf{1.39 \times 10^{0}}$	5.34×10^{0}	3.32×10^{0}	2.33×10^{1}
CEC11(F22)	Ave	2.35×10^{1}	2.09×10^{1}	2.06×10^{1}	2.12×10^{1}	2.00×10^{1}	1.32×10^{1}	3.50×10^{1}	$\mathbf{1.29 \times 10^{1}}$	2.22×10^{1}	5.09×10^{1}	2.52×10^{1}	8.85×10^{1}
	Std	5.37×10^{0}	5.91×10^{0}	2.56×10^{0}	2.41×10^{0}	3.09×10^{0}	2.78×10^{0}	8.81×10^{0}	2.93×10^{0}	$\mathbf{1.81 \times 10^{0}}$	4.65×10^{0}	2.77×10^{0}	1.26×10^{1}
Ranking (W|T|L)		0|0|22	0|0|22	4|4|14	2|3|17	1|6|15	0|4|18	0|4|18	0|3|19	5|4|13	3|5|14	3|3|16	0|2|20

Table 12 Average ranking of mRIME in relation to other algorithms using Friedman's test based upon their results on the CEC-2017 functions, which have 10 dimensions each

Algorithm	Rank
mRIME	5.98275862
RIME	6.53448275
WSO	4.46551724
TLBO	6.18965517
SFS	4.48275862
DE	3.96551724
GA	10.55172412
GSK	4.56896551
AMO	3.60344827
PSO	7.67241379
BBO	8.79310344
ACO	11.1896551

The p-value obtained by using Friedman's procedure on CEC-2017 with dimensionality 10 is 8.75425287E−11 in Table 12. This insinuates that the competing algorithms' performances differ statistically significantly from one another. The control algorithm is AMO, on the basis of the results in Table 12. Although mRIME is rated sixth in this table, after AMO, DE, WSO, SFS, and GSK, there are not numerous distinctions between mRIME and those competing algorithms. Regarding the remaining algorithms, TLBO, RIME, PSO, BBO, GA, and ACO are presented after mRIME in that order.

Further computations by the second statistical test, Holm's method, are required to determine which algorithms perform substantially differently and which ones perform similarly to the proposed mRIME. To determine if the efficiency of mRIME statistically differs from that of various other algorithms, these computations are crucial. In order to do this, Holm's test [67], a post-hoc statistical technique, was taken into consideration. This technique is crucial for determining which approaches perform more or less well than mRIME. Holm's statistical test uses $\alpha/k - i$, wherein k is the freedom amount and i is the technique number, to compare all algorithms sorted as per their p-values. This approach dismisses the null hypothesis sequentially, beginning with the most significant p-value and continuing until $p_i < \alpha/k - i$. The process ends when the hypothesis is rejected, at which point any further theories are considered plausible. Table 13 displays the outcomes of using Holm's test on the CEC-2017 test sample with a dimension of 10 as a post-hoc technique following Friedman's test.

Based on Holm's test, hypotheses with a p-value ≤ 0.01 are disregarded in Table 13. This table divulges that mRIME performs statistically significantly better than some of the other competing strategies discussed above, with little deviation from those superior algorithms.

The ranking results of Friedman's test utilized for the mean error findings of the CEC-2017 test with dimensions 30 for each function are summarized in Table 14. Based on their performance on the CEC-2017 test functions, each with 30 dimensions, the average ranking of several techniques (mRIME, RIME, WSO, TLBO, SFS, DE, GA, GSK, AMO, PSO, BBO, and ACO) is shown in Table 14. Friedman's test, a non-parametric statistical test designed to identify consideration differences over several test attempts, serves as the

Table 13 Holm's test results for the CEC-2017 test group, where each function's dimensions is 10

i	mRIME vs.	z-value	p-value	α/i (0.05)	Hypothesis
11	ACO	8.01192744	1.12924480E−15	0.00454545	0
10	GA	7.33819718	2.16490060E−13	0.005	0
9	BBO	5.48088673	4.23199367E−08	0.00555555	0
8	PSO	4.29730654	1.72886034E−05	0.00625	0
7	RIME	3.09551742	0.0019646	0.00714285	0
6	TLBO	2.73133890	0.00630775	0.00833333	0
5	mRIME	2.51283179	0.1197664	0.01	1
4	GSK	1.01969985	0.30787082	0.0125	1
3	SFS	0.92865522	0.35306779	0.01666666	1
2	WSO	0.91044630	0.36258718	0.025	1
1	DE	0.38238744	0.70217400	0.05	1

"0" means rejected and "1" means Not rejected

Table 14 Average ranking of mRIME in relation to other algorithms using Friedman's test based upon their results on the CEC-2017 functions, which have 30 dimensions each

Algorithm	Rank
mRIME	5.34482758
RIME	6.91379310
WSO	4.34482758
TLBO	6.87931034
SFS	3.93103448
DE	4.91379310
GA	10.72413793
GSK	3.94827586
AMO	3.74137931
PSO	7.67241379
BBO	7.58620689
ACO	12.00000000

foundation for the ranking. The algorithms for every function are ranked in the test, and the ranks are then averaged.

Based on the CEC-2017 functions with a dimension of 30, the p-value obtained using Friedman's test is 9.06069663E−11. Given that the p-value suggests a statistically significant difference between the methods, the null hypothesis is disproved. The data in Table 14 clearly demonstrate that, in comparison to all other rivals, mRIME scored rather well with a rank of 5.34482758, outperforming the parent RIME algorithm, which has a rank of 6.91379310. The following algorithms are ranked in order in Table 14: AMO, SFS, GSK, WSO, DE, mRIME, TLBO, RIME, BBO, PSO, GA, and ACO is ranked last. Once again, Table 14 shows that there is a substantial disparity amongst mRIME and RIME, TLBO, BBO, PSO, GA, and ACO, but a very modest difference between mRIME and AMO, GSK, and SFS. This highlights how well the evolved mRIME performs while optimizing CEC-2017 with large-dimensionality functions. Table 15 shows the results of using Holm's test on the 30-dimension CEC-2017 test functions.

The findings of Holm's test, a statistical technique for managing the family-wise error rate during multiple comparisons, are shown in Table 15. mRIME, DE, GSK, SFS, WSO, ACO, GA, PSO, BBO, RIME, and TLBO are the competitive algorithms whose perfor-

Table 15 Holm's test results for the CEC-2017 test group, where each function's dimensions is 30

i	mRIME vs.	z-value	p-value	α/i (0.05)	Hypothesis
11	ACO	8.72207556	2.73147934E−18	0.00454545	0
10	GA	7.37461503	1.64820606E−13	0.005	0
9	PSO	4.15163513	3.30108255E−05	0.00555555	0
8	BBO	4.06059050	4.89487584E−05	0.00625	0
7	RIME	3.35044238	8.06825911E−04	0.00714285	0
6	TLBO	3.31402453	9.19634372E−04	0.00833333	0
5	mRIME	1.69343011	0.09037362	0.01	1
4	DE	1.23820696	0.21563932	0.0125	1
3	WSO	0.63731241	0.52392136	0.01666666	1
2	GSK	0.21850711	0.82703401	0.025	1
1	SFS	0.20029818	0.84124738	0.05	1

"0" means rejected and "1" means Not rejected

mance is being compared amongst others on the CEC-2017 test functions, each of which has 30 dimensions. The Holm's test procedure rejected all of the hypotheses with p-value ≤ 0.01 in Table 15. The findings in this table, which are in line with earlier ones, attest to mRIME's strong position among its main competitors. Once again, Table 15 presents the results of the contrasts between mRIME and numerous other well-known and competitive algorithms, including ACO, GA, PSO, BBO, RIME, and TLBO. The results show that there are notable differences in the efficacy of mRIME and these opposing algorithms, rejecting the null hypothesis. In contrast, the contrasts between mRIME and other algorithms, such as DE, WSO, GSK, and SFS, did not reject the null hypothesis, suggesting that there are no appreciable variations in mRIME's performance from these highly efficient algorithms. According to this average rating, mRIME's performance score lags below that of DE and SFS but outperforms several of its rivals, including PSO, GA, as well as the native RIME algorithm.

The statistical outcomes of performing Friedman's test to the performance of mRIME and other contending techniques on the CEC-2017 benchmark test collection in 50d test problems are displayed in Table 16. Stated otherwise, the performance of mRIME on the CEC-2017 test functions-all of which have 50 dimensions-is compared to that of several other algorithms, such as RIME, WSO, TLBO, SFS, DE, GA, GSK, AMO, PSO, BBO, and ACO, in this table, on average. This rating, as stated several times before, is predicated on Friedman's test, to identify examination variations between test tries. The algorithms for every function are ranked in the test, and the ranks are then averaged.

Based on the outcomes shown in Table 16, it is determined that 1.07684638E−10 is the p-value that Friedman's test yields for the mean error estimates of the CEC-2017 problems with dimensions of 50. These results signify that there are scientifically significant differences between the competing algorithms, rejecting the null hypothesis. Table 16 shows that AMO performed non-significantly better than the proposed mRIME method. This table also shows that the mean score of mRIME is 4.56896551, which is not much behind the average rankings of WSO, GSK, and SFS, but better than the average rankings of several other algorithms, including DE, TLBO, RIME, PSO, BBO, GA, and ACO. Once more, Table 16 presents average ranking results that are consistent with the previously discussed findings for various dimensions. The findings of Holm's test, a statistical tech-

Table 16 Average ranking of mRIME in relation to other algorithms using Friedman's test based upon their results on the CEC-2017 functions, which have 50 dimensions each

Algorithm	Rank
mRIME	4.56896551
RIME	6.93103448
WSO	4.46551724
TLBO	6.77586206
SFS	4.55172413
DE	5.39655172
GA	10.34482758
GSK	4.50000000
AMO	4.17241379
PSO	7.10344827
BBO	7.18965517
ACO	12.00000000

Table 17 Holm's test results for the CEC-2017 test group, where each function's dimensions is 50

i	mRIME vs.	z-value	p-value	α/i (0.05)	Hypothesis
11	ACO	8.26685241	1.37542058E−16	0.00454545	Rejected
10	GA	6.51879551	7.08741942E−11	0.005	0
9	BBO	3.18656205	0.00143974	0.00555555	0
8	PSO	3.09551742	0.00196469	0.00625	0
7	RIME	2.91342816	0.00357484	0.00714285	0
6	TLBO	2.74954782	0.00596775	0.00833333	0
5	DE	1.29283374	0.19606856	0.01	1
4	mRIME	0.41880529	0.67535843	0.0125	1
3	SFS	0.40059637	0.68871731	0.01666666	1
2	GSK	0.34596959	0.72936556	0.025	1
1	WSO	0.30955174	0.75690185	0.05	1

"0" means rejected and "1" means Not rejected

nique used to regulate the family wise error rate while conducting multiple comparison tests, are shown in Table 17. In this instance, the test is being used to assess how well the considered mRIME algorithm operates in comparison to other appropriate algorithms, such as WSO, ACO, GA, BBO, PSO, RIME, TLBO, DE, SFS, and GSK, on the CEC-2017 test functions, each of which has 50 dimensions.

Holm's test procedure was used to reject the claims with p-value ≤ 0.01 in Table 17. According to the outcomes in this table, mRIME is a statistically effective algorithm that performs on par with the best algorithms, such as AMO, WSO, GSK, and SFS. Stated differently, Table 17 shows that, when mRIME is compared to the methods ACO, GA, BBO, PSO, RIME, and TLBO, the null hypothesis was rejected, implying that the efficacy of the proposed mRIME algorithm differs significantly from these competitor algorithms. For the evaluations between mRIME and other top-performing algorithms, such as DE, SFS, GSK, and WSO, the assumption of a null was not, however, refuted. This indicates that the effectiveness of mRIME and various efficient algorithms that have the highest performance levels documented in the literature do not differ significantly from one another. The average ranking of the developed mRIME method against other

Table 18 Average ranking of mRIME in relation to other algorithms using Friedman's test based upon their results on the CEC-2017 functions, which have 100 dimensions each

Algorithm	Rank
mRIME	4.79310344
RIME	7.13793103
WSO	4.94827586
TLBO	7.27586206
SFS	5.08620689
DE	6.41379310
GA	10.20689655
GSK	3.48275862
AMO	4.03448275
PSO	6.27586206
BBO	6.36206896
ACO	11.98275862

competing algorithms is shown in Table 18, which is based on how well each algorithm behaved on the CEC-2017 test functions, which have 100 dimensions.

The findings shown in Table 18 indicate a significant difference, as determined by Friedman's test at $\alpha = 0.05$. Specifically, 7.49404982E−11 is the p-value that Friedman's test yielded for the CEC-2017 findings in 100d. Table 18 makes it evident that, in these test situations with the taken into consideration dimension, GSK performs better than the other competing algorithms. To be more precise, Table 18 shows that the proposed mRIME algorithm is among the best optimization algorithms, outperforming WSO, SFS, PSO, BBO, DE, RIME, TLBO, GA, and ACO with an efficient degree of performance and a very respectable rank of 4.79310344.

It is important to note that there is a noticeable difference between the parent RIME algorithm and the derived mRIME, denoting that the melioration conducted on the native RIME algorithm is efficient. Though AMO is among the preferable and solid optimization algorithms mentioned in the literature, mRIME's rank is slightly lesser than AMO's rank, and the gap between mRIME, AMO, and GSK is not substantial. The algorithms are, in brief, ranked from best to worst as follows: GSK, AMO, mRIME, WSO, SFS, BBO, PSO, DE, RIME, TLBO, GA, and lastly, ACO. Following Friedman's test, Holm's test method was implemented using 100d as a follow-up testing method on the CEC-2017 test suite. The statistical findings of Holm's test on the CEC-2017 test functions with 100 dimensions are shown in Table 19.

Holm's test technique rejected each hypothesis with a p-value ≤ 0.0125 in Table 19. When mRIME is put up against the most potential algorithms that have achieved the greatest performance in the literature, such AMO and GSK, the findings in this table demonstrate that mRIME is a prospective optimization method for addressing high-dimensional problems. As Table 19 illustrates, there is not a substantial distinction between mRIME and GSK, AMO, WSO, and SFS, and Holm's test procedure did not reject the hypotheses. However, there is a major distinction between mRIME in comparison to other competitors, including PSO, BBO, DE, RIME, TLBO, GA, and ACO, where Holm's test procedure rejected the hypotheses.

Table 19 Holm's test results for the CEC-2017 test group, where each function's dimensions is 100

i	mRIME vs.	*z*-value	*p*-value	α/i (0.05)	Hypothesis
11	ACO	8.97700052	2.78248558E−19	0.00454545	0
10	GA	7.101481147	1.23426850E−12	0.005	0
9	TLBO	4.00596372	6.17651286E−5	0.00555555	0
8	RIME	3.86029231	1.13251462E−4	0.00625	0
7	DE	3.09551742	0.00196469	0.00714285	0
6	BBO	3.04089064	0.00235879	0.00833333	0
5	PSO	2.94984601	0.00317932	0.01	0
4	SFS	1.69343011	0.09037362	0.0125	1
3	WSO	1.54775871	0.12168040	0.01666666	1
2	mRIME	1.383878	0.16639569	0.025	1
1	AMO	0.58268563	0.56010494	0.05	1

"0" means rejected and "1" means Not rejected

Table 20 Average ranking of mRIME in relation to other algorithms using Friedman's test based upon their results on the CEC-2011 test group

Algorithm	Rank
mRIME	3.24999999
RIME	7.22727272
WSO	4.13636363
TLBO	5.47727272
SFS	4.86363636
DE	6.47727272
GA	8.97727272
GSK	4.95454545
AMO	4.63636363
PSO	10.36363636
BBO	6.454545454
ACO	11.18181818

Lastly, Table 20 applies Friedman's test with $\alpha = 0.05$ to the mean error results shown in Table 11 which belong to the results of the comparative algorithms on CEC-2011, summarizing the ranking outcomes of mRIME in comparison to other competing approaches.

Table 20 yields a *p*-value of 6.94070356E−11, as determined by Friedman's test. This supports the notion that the performances of the assessed algorithms varied statistically significantly. As per the results in Table 20, the control technique is the proposed mRIME algorithm that yielded the best rank, with WSO, AMO, SFS, GSK, TLBO, BBO, DE, RIME, GA, PSO, and ACO following in order of preference. AMO and SFS came in third and fourth position, respectively, while WSO got the second rank out of all algorithms, as Table 20 makes evident. According on its stated performance degree, which considerably surpassed that of several other algorithms including PSO, BBO, and GA, mRIME appears to be able to outperform promising optimization methods.

Holm's test was then used after Friedman's test to make sure that the variances between mRIME and the others in Table 20 are statistically significant; the statistical findings are shown in Table 21. In this instance, the test is being utilized to evaluate mRIME's performance level on the CEC-2011 test functions against that of alternative algorithms, including WSO, ACO, PSO, GA, RIME, DE, BBO, TLBO, GSK, SFS, and AMO.

Table 21 Holm's test results for the CEC-2011 test group

i	mRIME vs.	*z*-value	*p*-value	α/i (0.05)	Hypothesis
11	ACO	7.29621153	2.95983429E−13	0.00454545	0
10	PSO	6.54359372	6.00578145E−11	0.005	0
9	GA	5.26832466	1.37674455E−7	0.00555555	0
8	RIME	3.65855879	2.53637558E−4	0.00625	0
7	DE	2.96865913	0.00299102	0.00714285	0
6	BBO	2.94775308	0.00320092	0.00833333	0
5	TLBO	2.04879292	0.04048236	0.01	0
4	GSK	1.56795376	0.11689192	0.0125	1
3	SFS	1.48432956	0.13772150	0.01666666	1
2	AMO	1.27526906	0.20221402	0.025	1
1	WSO	0.81533595	0.41488003	0.05	1

"0" means rejected and "1" means Not rejected

Using Holm's technique, the hypotheses with p-value ≤ 0.01 in Table 21 are rejected. The results of this test show that mRIME can deliver competitive performance comparable to other promising algorithms stated in the literature. It is evident from Table 21 that when mRIME is compared to other algorithms like TLBO, BBO, DE, RIME, GA, PSO, and ACO, the null hypothesis is rejected. This points out that the performance of these algorithms and mRIME differs significantly. In contrast, the assessments between mRIME and the remaining algorithms-WSO, GSK, SFS, and AMO-did not result in the null hypothesis being rejected. This indicates that the performance level of the mRIME algorithm and that of various competing methods do not differ much.

One important conclusion drawn from the statistically significant results discussed above is that, on average, mRIME outperformed several of the robust state-of-the-art techniques mentioned in the literature, including BBO, GA, and ACO. This highlights the effective performance of mRIME and confirms that the search space may be effectively explored by this developed optimization method, regardless of the number of optima-one, two, or many-or the dimensionality of the optimization problems. Furthermore, the average ranking shows that, although the effectiveness of mRIME, AMO, and GSK is far from that of all other rivals, including GA, BBO, RIME, and ACO, the performance score of mRIME is not lag behind that of AMO, GSK, and SFS. It may be inferred that the remarkable performance of the mRIME algorithm on CEC-2011 and CEC-2017 is mostly due to its well-designed and functional mathematical model. Overall, this statistical analysis's findings show that mRIME is a dependable and efficient optimization approach with well calibrated exploration and exploitation characteristics that preserve a balance between the variety of local and global optimization features. These conclusions provide motivation to use this technique to tackle more challenging real-world optimization problems.

Evaluation FS results

Table 23 compares the average classification accuracy between the basic RIME and proposed mRIME utilizing the four TFs mentioned earlier. Using the X-shaped and U-shaped TFs, the proposed mRIME outperforms the standard RIME in 8 datasets (CARCINOM, CLL_SUB_111, Colon, GLI_85, GLIOMA, LUNG_DISCRETE, LYMPHOMA, and nci9) as shown in Table 22. In 5 datasets, the mRIME outperforms the RIME with the V-shaped TF, while in 2 datasets, the performance is equal. In contrast, with the S-shaped

Table 22 A brief description of the datasets utilized in this study

Dataset	No. features	No. instances	No. classes
ALLAML	7129	72	2
Carcinom	9182	174	11
CLL_SUB_111	11340	111	3
Colon	2000	62	2
GLI_85	22283	85	2
GLIOMA	4434	50	4
Lung_discrete	325	73	7
Lung	3312	203	5
Lymphoma	4026	96	9
nci9	9712	60	9

TF, mRIME performs equally well in 6 datasets and better in the ALLAML, Colon, and LYMPHOMA datasets. According to the results, the RIME modifications have a superior influence on accuracy measures since the mRIME surpasses the basic RIME in 6 datasets and is equal in three datasets. The searchability of the covered positions is strengthened by employing varying search strategies, especially with the high-dimension datasets. This proves that data reduction using these modifications has directly improved the algorithm's performance.

The comparison between the average fitness values obtained by mRIME and the basic RIME utilizing various TFs is shown in Table 24. In 90% of the datasets using the X-shaped TF, it can be observed that the mRIME outperforms the RIME and yields the fittest values when compared with RIME in 8 datasets. The mRIME_S and mRIME_V placed second and third, respectively, with 60% and 50% of the most fitting values across the examined datasets. The mRIME_U came in second with 80% of the most fitting values. As mentioned earlier, the mRIME used techniques to enhance the soft and hard RIMEs behavior, with the aid of the strength of the X-shaped TF, by discovering both directions of the search space and with the rollback operator proves the obtained results using this TF.

Sensitivity or recall measure is a crucial measurement for biomedical datasets. A high sensitivity indicates that the model accurately identifies the most positive findings. In contrast, low sensitivity means that the model is missing a significant portion of the positive results. Table 25 illustrating the sensitivity results of the mRIME and the basic RIME algorithms, the mRIME shows a superior performance by achieving the highest sensitivity results over all the tested datasets using the S-shaped Tf and 70%, 60% using the U and V-shaped TFs. These findings prove the effect of the modifications in enhancing the ML model's efficacy for biomedical datasets.

Table 26 compares the proposed mRIME with basic RIME for the different TFs in terms of specificity for several datasets. The mRIME outperforms the basic RIME in 6 datasets, whereas the mRIME-S achieves better in terms of AV in 4 datasets, LUNG_DISCRETE, LUNG, LYMPHOMA, and nci9. The mRIME, utilizing various TFs, outperforms the different versions of the basic RIME in CARCINOM, GLIOMA, LUNG_DISCRETE, LUNG, LYMPHOMA, and nci9 datasets. In addition to the AV measure, mRIME gets the lowest SD values in 7 datasets (70%), proving the proposed algorithm's stability. Based on the results presented above, the mRIME indicates a considerable improvement in the majority of studied scenarios.

Table 23 Comparison results between the proposed mRIME and the basic RIME for different TFs methods in terms of average classification accuracy

Dataset	Measure	mRIME_S	mRIME_U	mRIME_V	mRIME_X	RIME_S	RIME_U	RIME_V	RIME_X
ALLAML	AV	0.95516	0.94838	0.94631	0.94661	0.95428	**0.95605**	0.94867	**0.95605**
	SD	0.00323	0.004696	0.00398	0.001616	0.003354	0.001616	0.005401	0.00283
CARCINOM	AV	0.87879	0.90794	0.91106	**0.92155**	0.87879	0.87879	0.90287	0.87879
	SD	0	0.019449	0.031599	0.03534	0	0	0.017475	0
CLL_SUB_111	AV	0.73485	**0.87424**	0.87273	0.85303	0.73485	0.73939	0.83333	0.74394
	SD	0.01723	0.040802	0.048341	0.048757	0.01723	0.020444	0.043588	0.022279
Colon	AV	0.93889	**1.0000**	0.99167	0.99167	0.93056	0.93611	**1.0000**	0.93333
	SD	0.037481	0	0.025427	0.025427	0.031587	0.035849	0	0.033903
GLI_85	AV	0.94706	**1.0000**	**1.0000**	**1.0000**	0.95098	0.95098	**1.0000**	0.95882
	SD	0.017949	0	0	0	0.022297	0.022297	0	0.027417
GLIOMA	AV	0.9	**0.99667**	0.98667	0.97667	0.9	0.9	0.97667	0.9
	SD	0	0.018257	0.034575	0.043018	0	0	0.043018	0
LUNG_DISCRETE	AV	0.92857	0.94782	0.93924	**0.95916**	0.92857	0.92857	0.95288	0.92857
	SD	0	0.035028	0.041386	0.045002	0	0	0.036823	0
LUNG	AV	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
	SD	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
LYMPHOMA	AV	0.85088	0.92797	**0.9579**	0.95614	0.84912	0.85789	0.92983	0.85439
	SD	0.01995	0.035142	0.037599	0.048046	0.018197	0.024531	0.034784	0.022641
nci9	AV	0.73273	0.96471	**0.99364**	0.96302	0.74242	0.74545	0.96006	0.74061
	SD	0.020895	0.047788	0.024247	0.060094	0.030975	0.033763	0.054696	0.030549
Ranking (W|T|L)		0|1|9	2|3|5	2|2|6	1|2|7	0|1|9	1|1|8	0|3|7	1|1|8

Another important measure is the number of selected features, which indicates the reduction rate of the original datasets. The mRIME shows incredible findings compared with the basic RIME in all tested TFs. The mRIME selects a less relevant number of features in all datasets with an exciting variance. Table 27 demonstrates the average number of features chosen utilizing various TFs for both mRIME and basic RIME algorithms. The mRIME_V achieves the lowest reduction rate in 9 of 10 datasets. For example, in the LUNG dataset, the mRIME_V chooses 68.046 features with a 98% reduction rate, while the basic RIME in all versions chooses more than 1500 features since the total number of features is 3312. Similarly, the Carcinom dataset has a 97% reduction rate using mRIME_V. These findings prove the proposed modifications' efficiency in enhancing the original algorithm and the improved exploration and exploitation phases in the mRIME.

The modifications in the proposed mRIME force the algorithm to discover more positions in the search space, which increases the computation time of the mRIME. The TF also plays an essential role in the searching process. As clearly shown in Table 28, the basic RIME consumes less time for optimization. On the other hand, the mRIME findings prove the overall performance of the used modification instead of the CPU time. The mRIME_U takes less average time in computation in the LUNG_DISCRETE and LYMPHOMA datasets, and the mRIME_V achieves better CPU time in the Colon dataset, but both did not get the lowest standard deviations.

Convergence curves

The convergence curves for the proposed mRIME and the basic RIME over all the tested datasets are drawn in Fig. 3. This figure, combined with the conducted findings in the

Table 24 Comparison results between the proposed mRIME and the basic RIME for different TFs methods in terms of average fitness values

Dataset	Measure	mRIME_S	mRIME_U	mRIME_V	mRIME_X	RIME_S	RIME_U	RIME_V	RIME_X
ALLAML	AV	0.048981	0.051913	0.054355	0.053794	0.049631	0.047685	0.051674	**0.047599**
	SD	0.003053	0.004495	0.00401	0.001718	0.003047	0.001789	0.00513	0.002617
CARCINOM	AV	0.12601	0.091267	0.088139	**0.077764**	0.12604	0.12592	0.096319	0.12612
	SD	0.000368	0.01926	0.031291	0.035005	0.000485	0.000352	0.017311	0.000225
CLL_SUB_111	AV	0.2685	**0.12451**	0.12600	0.14551	0.26863	0.26381	0.16501	0.25967
	SD	0.016974	0.040391	0.047857	0.048269	0.016949	0.020098	0.043148	0.02187
Colon	AV	0.06577	**1E−05**	0.008261	0.008264	0.074059	0.068537	1.3E−05	0.071323
	SD	0.036688	2E−06	0.025171	0.025171	0.030923	0.035017	6E−06	0.033175
GLI_85	AV	0.058196	**1E−06**	**1E−06**	2E−06	0.054499	0.054178	1E−06	0.046693
	SD	0.017585	0	0	1E−06	0.021956	0.021793	1E−06	0.026872
GLIOMA	AV	0.10385	**0.003308**	0.013208	0.023112	0.10385	0.10386	0.023111	0.10386
	SD	3.9E−05	0.018074	0.034228	0.042586	4.4E−05	4.8E−05	0.042585	4.3E−05
LUNG_DISCRETE	AV	0.076188	0.051914	0.060489	**0.040696**	0.076433	0.076171	0.046911	0.076354
	SD	0.00041	0.034649	0.041022	0.04457	0.000271	0.000297	0.036507	0.000196
LUNG	AV	0.004854	7.6E−05	7.1E−05	**4.5E−05**	0.004848	0.004827	0.000102	0.004844
	SD	4.3E−05	3.9E−05	7.5E−05	2E−05	4.9E−05	4.1E−05	8.3E−05	5E−05
LYMPHOMA	AV	0.15307	0.071375	**0.04172**	0.043466	0.15491	0.14628	0.069535	0.1497
	SD	0.019413	0.034781	0.037219	0.047573	**0.017753**	0.02394	0.034428	0.022028
nci9	AV	0.27059	0.035004	**0.006318**	0.036626	0.26102	0.25796	0.039561	0.26279
	SD	**0.020561**	0.047357	0.024016	0.059502	0.030533	0.033182	0.054166	0.030141
Ranking (W|T|L)		1|0|9	4|0|6	3|0|7	3|0|7	1|0|9	0|0|10	0|0|10	0|0|10

Table 25 Comparison results between the proposed mRIME and the basic RIME for different TFs methods in terms of average sensitivity results

Dataset	Measure	mRIME_S	mRIME_U	mRIME_V	mRIME_X	RIME_S	RIME_U	RIME_V	RIME_X
ALLAML	AV	**0.90683**	0.89288	0.90281	0.89531	0.90298	0.90012	0.89366	0.89441
	SD	0.046374	0.049585	0.052191	0.043554	0.037556	0.048914	0.05503	0.050005
CARCINOM	AV	**1**	**1**	**1**	**1**	0.99444	**1**	**1**	**1**
	SD	0	0	0	0	0.030429	0	0	0
CLL_SUB_111	AV	0.64899	**0.7295**	0.71574	0.69833	0.62894	0.64056	0.68505	0.54602
	SD	0.37029	0.31123	0.392	0.31895	0.38164	0.3677	0.40547	0.33583
Colon	AV	0.60667	0.57849	**0.62373**	0.54675	0.57913	0.55702	0.56349	0.54405
	SD	0.21773	0.2433	0.2405	0.20446	0.29318	0.33933	0.29194	0.27052
GLI_85	AV	0.73992	0.75687	0.56544	0.71995	0.42098	0.59242	0.50982	**0.78849**
	SD	0.20196	0.20282	0.26045	0.16578	0.26891	0.25138	0.24271	0.17492
GLIOMA	AV	**0.76722**	0.74111	0.63063	0.62833	0.60389	0.74833	0.71429	0.71095
	SD	0.32898	0.25441	0.33682	0.30236	0.33235	0.3346	0.28137	0.27893
LUNG_DISCRETE	AV	0.90556	0.85833	0.86729	0.83889	0.71111	0.63889	0.71944	**0.91111**
	SD	0.23441	0.22118	0.21847	0.35147	0.41276	0.44814	0.43134	0.2263
LUNG	AV	0.97911	0.96865	0.96436	0.98314	0.94814	0.95838	0.96213	**0.98477**
	SD	0.028902	0.032605	0.025568	0.024262	0.046342	0.033537	0.033789	0.020166
LYMPHOMA	AV	0.99163	0.98632	0.93652	0.99373	0.91995	0.95368	0.97654	**0.99667**
	SD	0.025612	0.035992	0.080725	0.024272	0.090252	0.080676	0.0438	0.018257
nci9	AV	0.61530	0.62222	0.60833	0.61778	0.48889	0.46667	**0.67500**	0.70278
	SD	0.37120	0.36602	0.42725	0.37557	0.45208	0.4557	0.41312	0.31233
Ranking (W|T|L)		2|1|7	1|1|8	1|1|8	0|1|9	0|0|10	0|1|9	1|1|8	4|1|5

Table 26 Comparison results between the proposed mRIME and the basic RIME for different TFs methods in terms of specificity results

Dataset	Measure	mRIME_S	mRIME_U	mRIME_V	mRIME_X	RIME_S	RIME_U	RIME_V	RIME_X
ALLAML	AV	0.95366	0.96516	0.96253	0.96666	0.96574	0.96233	**0.96809**	0.95814
	SD	0.023086	0.024873	0.022472	0.020807	0.019925	0.022937	0.022485	0.025931
CARCINOM	AV	0.81914	**0.84400**	0.80954	0.80573	0.73157	0.74669	0.72036	0.71383
	SD	0.067203	0.066742	0.059442	0.075893	0.099354	0.091248	0.080811	0.081679
CLL_SUB_111	AV	0.51202	0.56819	0.51946	0.51812	0.62755	0.60003	**0.64101**	0.6182
	SD	0.12802	0.1248	0.087506	0.1131	0.13169	0.11087	0.13566	0.12976
Colon	AV	0.85743	0.82562	0.86316	0.83386	0.78802	**0.86829**	0.78845	0.79382
	SD	0.11704	0.12602	0.12235	0.15759	0.23494	0.18809	0.17382	0.15644
GLI_85	AV	0.87164	0.86207	0.86141	0.85239	**0.89733**	0.88269	0.8815	0.88212
	SD	0.12702	0.097707	0.11217	0.10329	0.092631	0.10674	0.098013	0.093336
GLIOMA	AV	0.82204	0.80273	0.81911	**0.82693**	0.5772	0.58632	0.56517	0.59452
	SD	0.14996	0.15907	0.14342	0.15037	0.2307	0.25763	0.17503	0.22201
LUNG_DISCRETE	AV	**0.85903**	0.83571	0.84975	0.84102	0.67473	0.68855	0.67547	0.66366
	SD	0.089455	0.10451	0.10765	0.10671	0.14847	0.15924	0.14835	0.14896
LUNG	AV	**0.8722**	0.84784	0.86273	0.85774	0.77855	0.75725	0.74487	0.76277
	SD	0.093717	0.11274	0.11542	0.11095	0.11929	0.10705	0.13626	0.10523
LYMPHOMA	AV	**0.78963**	0.74909	0.78441	0.77758	0.61271	0.58618	0.55215	0.54429
	SD	0.16047	0.13749	0.15923	0.10549	0.19374	0.16094	0.19692	0.21866
nci9	AV	**0.44324**	0.38444	0.37028	0.3528	0.20816	0.30185	0.2897	0.20247
	SD	0.16943	0.1362	0.165	0.14823	0.18206	0.19168	0.17502	0.14636
Ranking (W|T|L)		4|0|6	1|0|9	0|0|10	1|0|9	1|0|9	1|0|9	2|0|8	0|0|10

Table 27 Comparison results between the proposed mRIME and the basic RIME for different TFs methods in terms of average number of selected features

Dataset	Measure	mRIME_S	mRIME_U	mRIME_V	mRIME_X	RIME_S	RIME_U	RIME_V	RIME_X
ALLAML	AV	3.414	2.9874	3.0292	**2.7094**	13.533	12.933	14.233	12.667
	SD	3.846	2.6986	1.5326	0.893	1.7953	1.596	1.8323	1.4933
CARCINOM	AV	470.14	393.53	281.52	**243.25**	5545.7	5437.5	5518.4	5618.5
	SD	342.89	292.7	330.61	175.48	445.15	323.14	337.61	206.88
CLL_SUB_111	AV	296.65	323.08	**84.246**	388.78	6955.1	6583.9	6802	6993.8
	SD	190.75	511.48	222.16	257.19	384.69	481.44	455.29	399.36
Colon	AV	59.259	60.043	**15.784**	43.308	1061.9	1057.3	1054	1064.6
	SD	50.855	69.382	60.318	32.979	107.27	105.66	104.78	119.34
GLI_85	AV	494.32	658.38	**363.14**	443.64	13302	12587	12888	13211
	SD	374.87	845	735.79	331.81	996.4	1059.1	1009	1193.9
GLIOMA	AV	113.46	148.41	**61.635**	102	2150.4	2156.2	2149.1	2152.6
	SD	90.112	159.62	151.55	67.593	19.42	21.471	17.231	19.22
LUNG_DISCRETE	AV	19.043	17.565	**10.622**	13.528	185.87	177.33	177.9	183.3
	SD	28.921	12.829	10.768	9.6108	8.8033	9.6681	13.337	6.3688
LUNG	AV	144.8	99.596	**68.046**	108.39	1605.8	1598.8	1607.5	1604.4
	SD	91.35	69.151	87.295	82.247	16.357	13.595	14.183	16.473
LYMPHOMA	AV	194.28	126.75	**80.147**	126.85	2229.3	2254.3	2190.8	2229.9
	SD	113.91	119.29	94.568	94.66	227.26	196.88	208.97	247.34
nci9	AV	390.8	320.25	**280.07**	358.89	5846.5	5792.4	5813	5817
	SD	197.72	335.21	389.24	262.72	409.94	353.57	288.93	398.18
Ranking (W|T|L)		0|0|10	0|0|10	8|0|2	2|0|8	0|0|10	0|0|10	0|0|10	0|0|10

Table 28 Comparison results between the proposed mRIME and the basic RIME for different TFs methods in terms of average CPU times

Dataset	Measure	mRIME_S	mRIME_U	mRIME_V	mRIME_X	RIME_S	RIME_U	RIME_V	RIME_X
ALLAML	AV	18.25	10.897	10.13	50.58	16.588	**9.296**	9.9329	53.722
	SD	5.5629	3.3024	3.1303	4.5975	1.2454	1.9342	2.7134	4.2663
CARCINOM	AV	218.55	35.404	41.41	724.4	229.01	**27.561**	21.688	525.4
	SD	14.391	3.303	1.8417	17.258	101.15	4.1603	1.3627	38.852
CLL_SUB_111	AV	154.98	30.496	30.619	448.52	101.76	**24.056**	24.513	307.19
	SD	3.4536	4.7964	5.2988	9.0529	2.5691	1.1506	2.0252	14.574
Colon	AV	22.947	12.436	**11.339**	99.01	36.77	22.558	26.087	107.6
	SD	0.3569	3.289	2.5868	13.153	1.1156	2.7071	4.7389	3.6912
GLI_85	AV	198.53	42.015	39.182	495.39	145.62	33.698	**33.538**	464.44
	SD	4.0219	2.5465	4.1051	80.951	2.855	1.7316	1.4875	82.825
GLIOMA	AV	43.626	27.645	28.044	88.371	22.994	**14.158**	15.148	65.05
	SD	1.1147	2.9157	3.5264	32.598	1.9958	1.5476	1.1396	4.7871
LUNG_DISCRETE	AV	13.723	**11.695**	14.741	46.243	15.88	14.372	13.224	45.254
	SD	0.97115	1.9522	1.9145	4.0051	1.5998	1.721	1.3365	3.8309
LUNG	AV	66.512	23.409	15.625	263.57	66.415	16.348	**15.146**	301.38
	SD	4.8894	4.461	1.0184	56.844	4.6848	1.0991	1.02	23.172
LYMPHOMA	AV	52.064	**23.16**	26.235	152.52	54.103	28.041	27.509	155.6
	SD	3.0348	3.3912	1.8364	8.8118	3.1799	1.9251	1.9774	8.7665
nci9	AV	60.534	25.557	27.77	175.75	2354.2	22.649	**22.37**	124.71
	SD	3.4965	1.8841	2.4645	9.3992	5951	2.6	3.917	4.8825
Ranking (W|T|L)		0|0|10	2|0|8	1|0|9	0|0|10	0|0|10	4|0|6	3|0|7	0|0|10

best fitness results in Table 24 shows the convergence behavior of the tested FS optimizers. The optimization algorithm that exhibits rapid convergence is recommended in specific convergence patterns. In other words, we favor the optimizer that settles on a shallow fitness value in the fewest possible iterations. Various TFs for the basic and the proposed methods and the nature of the tested datasets make a variety of the convergence behavior of the algorithms since it directly affects the exploration and exploitation capabilities. In 8 datasets, including CARCINOM, CLL_SUB_111, Colon, GLIOMA, LUNG_DISCRETE, LUNG, LYMPHOMA, and nci9, the proposed mRIME surpasses the basic RIME by obtaining the lowest fitness values. This demonstrates the mRIME's search capability by using the proposed modifications and avoiding being trapped in local minima. Additionally, the mRIME_U converged faster for the ALLAML dataset by reaching superior fitness values in the early iterations, which received the lowest values between 20 and 50 iterations. In the colon dataset, the mRIME_X and mRIME_U demonstrate faster convergence by obtaining the lowest fitness after 10 iterations. Similarly, before entering the 10th iteration on the GLI_85 dataset, the mRIME_S, mRIME_X, and mRIME_V were rapidly converged. By noticing spots of light in the LUNG dataset, the mRIME_X and the RIME_V compete to obtain the lowest fitness value in the initial iterations.

In conclusion, it is evident from the results presented in Table 24 and illustrated in Fig. 3 that the modifications to the RIME algorithm strengthen the algorithm's searchability, balance the exploration and exploitation phases, and avoid local optima while exploring the search space.

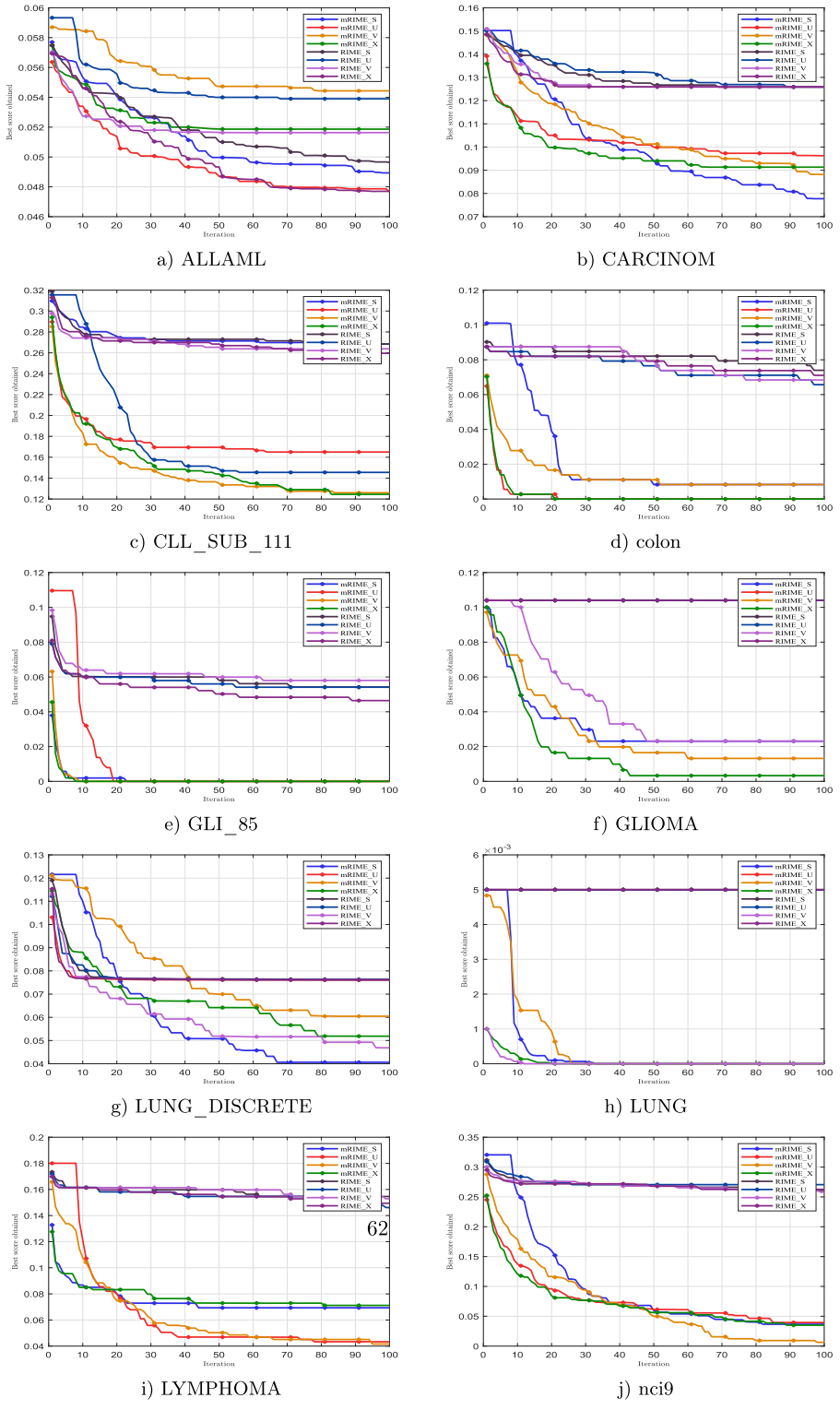

Fig. 3 Convergence curves of the proposed and basic algorithms for different TFs for all examined datasets

Statistical test

The above subsections employ several performance measures to evaluate the proposed modification with the basic RIME with various TFs. The average and standard deviation of the outcomes (i.e., classification accuracy and fitness values) found thus far in 30 independent runs serve as the statistical measures used in these comparisons. These metrics give a broad understanding of how well the suggested algorithms handle FS problems. The first measure depicts the average behavior of the proposed algorithms, while the second measure highlights how stable the proposed algorithms are across all 30 independent runs. Although these statistical metrics could show the suggested approaches' overall dependability and robustness, they cannot compare each of the 30 independent runs separately. In other words, they have demonstrated that the proposed FS approaches benefit from high levels of exploitation and exploration but have not been able to demonstrate how effective they are. The testing methods used by Friedman and Holm in this subsection show the importance of the findings and demonstrate that they were not the result of chance. In order to determine whether there is a fundamental difference in the outputs of the various algorithms, Friedman's test is predicated on the null hypothesis that there is no difference in the accuracy of any of the comparison algorithms.

For these types of tests, it is essential to find the p-value that Friedman's test will return; if the p-value is less than or equal to the degree of significance (0.05), then Disprove the null hypothesis. Which means there were statistically significant differences in the effectiveness of the evaluated algorithms. Following this statistical test, a post-hoc test technique is carried out, and Holm's approach is then used to investigate pairwise comparison of the rival algorithm. Usually used as a control approach for post-hoc analysis, the algorithm with the lowest rank is obtained by Friedman's test.

The findings have been evaluated using Friedman's and Holm's test techniques in order to prove the robustness of the suggested mRIME algorithm utilizing different TFs to solve FS problems. A summary of the statistical test results of Friedman's test, based on the results shown in Tables 23, 24, 25, 26, 27, 28, with various measures, is presented in Table 29, which shows the ranking of each version; hence, the lowest value for rank means best performed one and vice versa.

According to the accuracy results, the obtained p-value using Friedman's test is equal to 2.5619E−4; thus, The null hypothesis of equivalent performance degree was rejected with evidence of a statistically significant difference between the accuracy of the contending algorithms. The proposed mRIME with a U-shaped TF ranked as the best-performed algorithm compared with others with a 2.85 rank value. The mRIME_V and mRIME_X, with rankings of 3.05 and 3.1, respectively, came in second and third place. Similarly, the mRIME_U surpasses competitors with a 2.63 rank in the fitness values findings with a 4.8642E−6 p-value, followed by the mRIME_V and mRIME_X with 2.7 and 3.0 ranks.

On the other hand, mRIME_S ranked as the first competitor with a 2.9 rank using the sensitivity results; the p-value for Friedman's test over the sensitivity results is equal to 0.0201, which is less than the significant level of 5%, thus rejecting the null hypothesis also. In contrast, the null hypothesis was not rejected on the specificity results since the p-value is equal to 0.0762, greater than the significant level of 5% since the nature of the datasets plays an essential role in this measure. Instead, the mRIME_S outperforms others

Table 29 Average rankings of all competitor algorithms using Friedman's test

Algorithm	Accuracy	Fitness	Sensitivity	Specificity	Features	Run-time
mRIME_S	6.25	6.35	2.90	3.20	3.50	5.20
mRIME_U	2.85	2.60	3.70	3.90	2.90	2.90
mRIME_V	3.05	2.70	4.30	3.50	1.30	3.30
mRIME_X	3.10	3.00	4.30	3.90	2.30	7.50
RIME_S	6.40	6.85	6.60	5.10	7.10	5.50
RIME_U	5.40	5.25	5.60	4.70	5.80	2.30
RIME_V	3.40	3.30	5.00	5.50	6.10	2.00
RIME_X	5.55	5.95	3.60	6.20	7.00	7.30
p-value	2.5619E−4	4.8642E−6	0.0201	0.0762	1.9589E−10	1.1436E−9

with a 3.2 rank, followed by mRIME_V with a 3.5 rank, and the mRIME_U and mRIME_X are placed in the third order with the same rank of 3.9.

In the same manner, and for the number of selected features and CPU times, there is a statistically significant difference with p equal to 1.9589E−10 and 1.1436E−9 values, respectively. The mRIME_V was placed in the first order with a 1.3 rank in the results of the number of selected features. Meanwhile, RIME_V gained the lowest rank, 2.0, and the mRIME_U placed in the third rank, 2.9, ordered after the RIME_U with a 2.3 rank.

Then, Holm's test is used as a post-test technique to see whether there are statistically notable differences between the other comparison algorithms and the control binary method with the lowest rank. Based on the FS results for different measures and the statistical results of Holm's test method, Table 30 illustrates Holm's test results. $R - 0$ denotes Friedman's rank assigned to the control method, Friedman's rank allocated to algorithm i is denoted by R_i, and the ES denotes the effect size of the control binary technique on the algorithm. Together, these variables represent the statistical difference between the two algorithms. The difference between the two competing algorithms is noticeable at the significance level.

According to the classification accuracy and the best fitness values results in Table 23, mRIME_U was compared to other competing FS algorithms using Holm's test technique, disqualifying hypotheses with p-values less than 0.0055. As can be seen from the results in the first two sub tables of Table 30, mRIME_U outperforms other efficient and effective FS approaches in producing encouraging outcomes for the FS problems being studied. The p-values increase the mRIME_U's performance score on FS tasks and highlight the significance of this method. By realizing high classification accuracy rates and best fitness values compared to those reported by other existing FS methods, the results show that the proposed mRIME with U, V, and X-shaped TFs successfully avoided optimum local solutions while exploring and exploiting the search space and are statistically significant FS methods.

In contrast, for the sensitivity and specificity results, mRIME_S shows noticeable results for such optimization problems. It is clearly demonstrated that the proposed mRIME with various TFs successfully avoids local optimal solutions and is statistically significant. While the mRIME_V is the control algorithm for the number of selected features and the CPU running time results, the control algorithm outperforms others for both findings, with mRIME_U and mRIME_X being statistically significant methods in the number of

Table 30 Holm's test results between the control algorithm and all other comparative methods

i	Algorithm	$z^i = \frac{(R_0 - R^i)}{SE^i}$	p-value	$\alpha \div i$	Hypothesis
Classification accuracy (mRIME_U is the control algorithm)					
7	RIME_S	3.24069	0.00119	0.0071	0
6	mRIME_S	3.10376	0.00191	0.00833	0
5	RIME_X	2.46475	0.01371	0.01	0
4	RIME_U	2.32782	0.01992	0.0125	0
3	RIME_V	0.50207	0.61561	0.01666	1
2	mRIME_X	0.22821	0.81947	0.025	1
1	mRIME_V	0.18257	0.85513	0.05	1
Fitness (mRIME_U is the control algorithm)					
7	RIME_S	3.87970	1.04584E−4	0.0071	0
6	mRIME_S	3.42326	6.18735E−4	0.00833	0
5	RIME_X	3.05811	0.00222	0.01	0
4	RIME_U	2.41910	0.01555	0.0125	0
3	RIME_V	0.63900	0.52281	0.01666	1
2	mRIME_X	0.36514	0.71500	0.025	1
1	mRIME_V	0.09128	0.92726	0.05	1
Sensitivity (mRIME_S is the control algorithm)					
7	RIME_S	3.37762	7.31153E−4	0.0071	0
6	RIME_U	2.46475	0.01371	0.00833	0
5	RIME_V	1.91702	0.05523	0.01	1
4	mRIME_X	1.27801	0.20124	0.0125	1
3	mRIME_V	1.27801	0.20124	0.01666	1
2	mRIME_U	0.73029	0.46520	0.025	1
1	RIME_X	0.63900	0.52281	0.05	1
Specificity (mRIME_S is the control algorithm)					
7	RIME_X	2.73861	0.00616	0.0071	0
6	RIME_V	2.09960	0.03576	0.00833	0
5	RIME_S	1.73445	0.08283	0.01	1
4	RIME_U	1.36930	0.17090	0.0125	1
3	mRIME_U	0.63900	0.52281	0.01666	1
2	mRIME_X	0.63900	0.52281	0.025	1
1	mRIME_V	0.27386	0.78419	0.05	1
Features (mRIME_V is the control algorithm)					
7	RIME_S	5.29465	1.19243E−07	0.0071	0
6	RIME_X	5.20336	1.95712E−07	0.00833	0
5	RIME_V	4.38178	1.17713E−05	0.01	0
4	RIME_U	4.10791	3.99239E−05	0.0125	0
3	mRIME_S	2.00831	0.04460	0.01666	0
2	mRIME_U	1.46059	0.14412	0.025	1
1	mRIME_X	0.91287	0.36131	0.05	1
Run-time (RIME_V is the control algorithm)					
7	mRIME_X	5.02079	5.14593E−07	0.0071	0
6	RIME_X	4.83821	1.31009E−06	0.00833	0
5	RIME_S	3.19504	0.00139	0.01	0
4	mRIME_S	2.92118	0.00348	0.0125	0
3	mRIME_V	1.18673	0.23533	0.01666	1
2	mRIME_U	0.82158	0.41131	0.025	1
1	RIME_U	0.27386	0.78419	0.05	1

selected feature results. Meanwhile, mRIME_V and mRIME-U are statistically significant for the CPU time results.

To put it briefly, the statistical results of Friedman and Holm's test methods presented in Tables 29 and 30 highlight that the performance of the suggested algorithms is statistically significant and support their satisfactory performance and dependability. These results support the assertion that the proposed binary mRIME and its binary improved extensions can handle several well-known FS datasets rather effectively.

Table 31 Benchmarking functions

Function	Name	*Opt*
Unimodal category		
CEC17(f1)	Shifted and Rotated Bent Cigar Function	100
CEC17(f2)	Shifted and Rotated Sum of Different Power Function	200
CEC17(f3)	Shifted and Rotated Zakharov Function	300
Multimodal category		
CEC17(f4)	Shifted and Rotated Rosenbrock's Function	400
CEC17(f5)	Shifted and Rotated Rastrigin's Function	500
CEC17(f6)	Shifted and Rotated Expanded Scaffer's F6 Function	600
CEC17(f7)	Shifted and Rotated Lunacek Bi-Rastrigin Function	700
CEC17(f8)	Shifted and Rotated Non-Continuous Rastrigin's Function	800
CEC17(f9)	Shifted and Rotated Lévy Function	900
CEC17(f10)	Shifted and Rotated Schwefel's Function	1000
Hybrid category		
CEC17(f11)	Hybrid F1 (N=3)	1100
CEC17(f12)	Hybrid F2 (N=3)	1200
CEC17(f13)	Hybrid F3 (N=3)	1300
CEC17(f14)	Hybrid F4 (N=4)	1400
CEC17(f15)	Hybrid F5 (N=4)	1500
CEC17(f16)	Hybrid F6 (N=4)	1600
CEC17(f17)	Hybrid F6 (N=5)	1700
CEC17(f18)	Hybrid F6 (N=5)	1800
CEC17(f19)	Hybrid F6 (N=5)	1900
CEC17(f20)	Hybrid F6 (N=6)	2000
Composition category		
CEC17(f21)	Composite F1 (N = 3)	2100
CEC17(f22)	Composite F2 (N=3)	2200
CEC17(f23)	Composite F3 (N=4)	2300
CEC17(f24)	Composite F4 (N=4)	2400
CEC17(f25)	Composite F5 (N=5)	2500
CEC17(f26)	Composite F6 (N=5)	2600
CEC17(f27)	Composite F7 (N=6)	2700
CEC17(f28)	Composite F8 (N=6)	2800
CEC17(f29)	Composite F9 (N=3)	2900
CEC17(f30)	Composite F10 (N=3)	3000
Search scope:[-100, 100]	Dimension: = 30	

Final remarks and upcoming projects

Inspired by the natural phenomenon of RIME-ice production, this paper proposes a new method for global optimization and FS through the development of the sophisticated RIME algorithm. By introducing Binary mRIME, the article expands the use of the RIME algorithm to FS even further. To facilitate the transformation of a continuous search space into a binary one, four distinct types of TFs from the S-shaped, V-shaped, U-shaped, and X-shaped families were chosen for FS problems. With regard to these various TFs, the effectiveness of RIME is examined for global optimization through the use of CEC2011, CEC2017, and FS tasks in relation to applications for disease diagnosis. Ten of the most dependable optimization algorithms in the pertinent field of research have been used to test the proposed mRIME's outcomes on the CEC 2017 and CEC 2011 test suites and ten medical datasets. Several standard measurements and various statistical values have

been used for the evaluation of the algorithms. In order to illustrate the durability of the suggested mRIME, the outcomes have been evaluated utilizing the test techniques put out by Friedman and Holm. Evaluations show that the sophisticated RIME architecture performs better in FS and global optimization tasks, offering a fresh approach to challenging optimization issues across a range of industries. The paper provides scholars and practitioners in the optimization and financial science domains with a thorough introduction to comprehending and applying the sophisticated RIME algorithm and its binary counterpart. In the future, we intend to propose mRIME to solve different optimization problems in various applications such as intrusion detection, bioinformatics, and the Internet of Things. Furthermore, mRIME may be compared with other new binary transformation methods such as cosine similarity (Table 31).

Author contributions
RK: methodology, conceptualization, implementation, and supervision; MB: methodology and implementation. AZ: implementation and analysis. RD: writing KHD: writing and editing: BS: reviewing and editing. All authors read and approved the final manuscript.

Funding
Open Access funding enabled and organized by CAUL and its Member Institutions.

Code availability
Will be provided upon request.

Declarations

Ethics approval and consent to participate
Not applicable

Consent for publication
Not applicable

Competing interests
The authors declare no competing interests.

Received: 27 January 2024 Accepted: 7 May 2024
Published online: 18 June 2024

References

1. Arora JS, Elwakeil OA, Chahande AI, Hsieh CC. Global optimization methods for engineering applications: a review. Struct Optim. 1995;9(3–4):137–59. https://doi.org/10.1007/BF01743964.
2. Stork J, Eiben AE, Bartz-Beielstein T. A new taxonomy of global optimization algorithms. Nat Comput. 2022;21(2):219–42. https://doi.org/10.1007/s11047-020-09820-4.
3. Korani W, Mouhoub M. Review on nature-inspired algorithms. Oper Res Forum. 2021;2(3):36. https://doi.org/10.1007/s43069-021-00068-x.
4. Slowik A, Kwasnicka H. Evolutionary algorithms and their applications to engineering problems. Neural Comput Appl. 2020;32(16):12363–79. https://doi.org/10.1007/s00521-020-04832-8.
5. Brezocnik L, Fister I, Podgorelec V. Swarm intelligence algorithms for feature selection: a review. Appl Sci (Switzerland). 2018;8(9):1521. https://doi.org/10.3390/app8091521.
6. Tang J, Liu G, Pan Q. A review on representative swarm intelligence algorithms for solving optimization problems: applications and trends. IEEE/CAA J Autom Sin. 2021;8(10):1627–43. https://doi.org/10.1109/JAS.2021.1004129.
7. Jaszcz A, Polap D, Damaševicius R. Lung x-ray image segmentation using heuristic red fox optimization algorithm. Sci Program. 2022. https://doi.org/10.1155/2022/4494139.
8. Helmi AM, Al-Qaness MAA, Dahou A, Damaševicius R, Krilavicius T, Elaziz MA. A novel hybrid gradient-based optimizer and grey wolf optimizer feature selection method for human activity recognition using smartphone sensors. Entropy. 2021;23(8):1065. https://doi.org/10.3390/e23081065.
9. Khurma RA, Alsawalqah H, Aljarah I, Elaziz MA, Damaševicius R. An enhanced evolutionary software defect prediction method using island moth flame optimization. Mathematics. 2021. https://doi.org/10.3390/math9151722.
10. Savanovic N, Toskovic A, Petrovic A, Zivkovic M, Damaševicius R, Jovanovic L, Bacanin N, Nikolic B. Intrusion detection in healthcare 4.0 internet of things systems via metaheuristics optimized machine learning. Sustainability (Switzerland). 2023. https://doi.org/10.3390/su151612563.
11. Makhadmeh SN, Al-Betar MA, Awadallah MA, Abasi AK, Alyasseri ZAA, Doush IA, Alomari OA, Damaševicius R, Zajanckauskas A, Mohammed MA. A modified coronavirus herd immunity optimizer for the power scheduling problem. Mathematics. 2022. https://doi.org/10.3390/math10030315.

12. Khurma RA, Aljarah I, Sharieh A, Elaziz MA, Damaševicius R, Krilavicius T. A review of the modification strategies of the nature inspired algorithms for feature selection problem. Mathematics. 2022. https://doi.org/10.3390/math10030464.
13. Ikotun AM, Almutari MS, Ezugwu AE. K-means-based nature-inspired metaheuristic algorithms for automatic data clustering problems: recent advances and future directions. Appl Sci (Switzerland). 2021. https://doi.org/10.3390/app112311246.
14. Yaqoob A, Aziz RM, Verma NK, Lalwani P, Makrariya A, Kumar P. A review on nature-inspired algorithms for cancer disease prediction and classification. Mathematics. 2023. https://doi.org/10.3390/math11051081.
15. Su H, Zhao D, Heidari AA, Liu L, Zhang X, Mafarja M, Chen H. Rime: a physics-based optimization. Neurocomputing. 2023;532:183–214. https://doi.org/10.1016/j.neucom.2023.02.010.
16. Salimi H. Stochastic fractal search: a powerful metaheuristic algorithm. Knowl Based Sys. 2015;75:1–18.
17. Storn R, Price K. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim. 1997;11(4):341–59.
18. Simon D. Biogeography-based optimization. IEEE Trans Evol comput. 2008;12(6):702–13.
19. Holland JH. Genetic algorithms. Sci Am. 1992;267(1):66–73.
20. Kennedy J, Eberhart R. Particle swarm optimization. In: Kennedy J, editor. Proceedings of ICNN'95-International conference on neural networks, vol. 20. Perth: IEEE; 1995. p. 1942–8.
21. Dorigo M, Maniezzo V, Colorni A. Ant system: optimization by a colony of cooperating agents. IEEE Transact Syst Man Cybern Part B (Cybern). 1996;26(1):29–41.
22. Li X, Zhang J, Yin M. Animal migration optimization: an optimization algorithm inspired by animal migration behavior. Neural Comput Appl. 2014;24(7):1867–77.
23. Rao RV, Savsani VJ, Vakharia D. Teaching-learning-based optimization: an optimization method for continuous non-linear large scale problems. Inform Sci. 2012;183(1):1–15.
24. Mohamed AW, Hadi AA, Mohamed AK. Gaining-sharing knowledge based algorithm for solving optimization problems: a novel nature-inspired algorithm. Int J Mach Learn Cybern. 2020;11(7):1501–29.
25. Braik M, Hammouri A, Atwan J, Al-Betar MA, Awadallah MA. White shark optimizer: a novel bio-inspired meta-heuristic algorithm for global optimization problems. Knowl Based Syst. 2022;243: 108457.
26. Mohapatra S, Mohapatra P. An improved golden jackal optimization algorithm using opposition-based learning for global optimization and engineering problems. Int J Comput Intell Syst. 2023. https://doi.org/10.1007/s44196-023-00320-8.
27. Houssein EH, Oliva D, Samee NA, Mahmoud NF, Emam MM. Liver cancer algorithm: a novel bio-inspired optimizer. Comput Biol Med. 2023. https://doi.org/10.1016/j.compbiomed.2023.107389.
28. Abdel-Basset M, Mohamed R, Zidan M, Jameel M, Abouhawwash M. Mantis search algorithm: a novel bio-inspired algorithm for global optimization and engineering design problems. Comput Methods Appl Mech Eng. 2023. https://doi.org/10.1016/j.cma.2023.116200.
29. Liu Q, Li N, Jia H, Qi Q, Abualigah L. A chimp-inspired remora optimization algorithm for multilevel thresholding image segmentation using cross entropy. Artif Intell Rev. 2023;56(Suppl 1):159–216.
30. Liu Q, Li N, Jia H, Qi Q, Abualigah L. Modified remora optimization algorithm for global optimization and multilevel thresholding image segmentation. Mathematics. 2022;10(7):1014.
31. Liu Q, Li N, Jia H, Qi Q, Abualigah L, Liu Y. A hybrid arithmetic optimization and golden sine algorithm for solving industrial engineering design problems. Mathematics. 2022;10(9):1567.
32. Liu Q, Qi Q, Li N. Federated opposite learning based arithmetic optimization algorithm for image segmentation using multilevel thresholding. In: Liu Q, editor. 2023 26th International conference on computer supported cooperative work in design (CSCWD). Rio de Janeiro: IEEE; 2023.
33. Liu Q, Qi Q, Li N. Sleo: an efficient equilibrium optimizer for numerical optimization. In: Liu Q, editor. 2022 IEEE smartworld, ubiquitous intelligence and computing, scalable computing and communications, digital twin, privacy computing, metaverse, autonomous and trusted vehicles (SmartWorld/UIC/ScalCom/DigitalTwin/PriComp/Meta). Haikou: IEEE; 2022. p. 1696–701.
34. Nadimi-Shahraki MH, Zamani H, Asghari Varzaneh Z, Mirjalili S. A systematic review of the whale optimization algorithm: theoretical foundation, improvements, and hybridizations. Arch Comput Methods Eng. 2023;30(7):4113–59.
35. Hassan MH, Daqaq F, Selim A, Domínguez-García JL, Kamel S. Moimpa: multi-objective improved marine predators algorithm for solving multi-objective optimization problems. Soft Comput. 2023;27(21):15719–40. https://doi.org/10.1007/s00500-023-08812-7.
36. Mehmood K, Chaudhary NI, Khan ZA, Cheema KM, Raja MAZ, Shu C-M. Novel knacks of chaotic maps with archimedes optimization paradigm for nonlinear ARX model identification with key term separation. Chaos Solitons Fractals. 2023. https://doi.org/10.1016/j.chaos.2023.114028.
37. Zhou X, Chen Y, Wu Z, Heidari AA, Chen H, Alabdulkreem E, Escorcia-Gutierrez J, Wang X. Boosted local dimensional mutation and all-dimensional neighborhood slime mould algorithm for feature selection. Neurocomputing. 2023. https://doi.org/10.1016/j.neucom.2023.126467.
38. Painuli D, Bhardwaj S, et al. Recent advancement in cancer diagnosis using machine learning and deep learning techniques: a comprehensive review. Comput Biol Med. 2022;146: 105580.
39. Yu X, Qin W, Lin X, Shan Z, Huang L, Shao Q, Wang L, Chen M. Synergizing the enhanced rime with fuzzy k-nearest neighbor for diagnose of pulmonary hypertension. Comput Biol Med. 2023. https://doi.org/10.1016/j.compbiomed.2023.107408.
40. Emam MM, Samee NA, Jamjoom MM, Houssein EH. Optimized deep learning architecture for brain tumor classification using improved hunger games search algorithm. Comput Biol Med. 2023. https://doi.org/10.1016/j.compbiomed.2023.106966.
41. Chen J, Cai Z, Heidari AA, Liu L, Chen H, Pan J. Dynamic mechanism-assisted artificial bee colony optimization for image segmentation of COVID-19 chest X-ray. Displays. 2023. https://doi.org/10.1016/j.displa.2023.102485.
42. Deng L, Liu S. Snow ablation optimizer: a novel metaheuristic technique for numerical optimization and engineering design. Expert Syst Appl. 2023. https://doi.org/10.1016/j.eswa.2023.120069.

43. Dong R, Sun L, Ma L, Heidari AA, Zhou X, Chen H. Boosting kernel search optimizer with slime mould foraging behavior for combined economic emission dispatch problems. J Bionic Eng. 2023. https://doi.org/10.1007/s42235-023-00408-z.
44. Fatahi A, Nadimi-Shahraki MH, Zamani H. An improved binary quantum-based avian navigation optimizer algorithm to select effective feature subset from medical data: a COVID-19 case study. J Bionic Eng. 2024;21(1):426–46.
45. Nadimi-Shahraki MH, Asghari Varzaneh Z, Zamani H, Mirjalili S. Binary starling murmuration optimizer algorithm to select effective features from medical data. Appl Sci. 2022;13(1):564.
46. Kennedy J, Eberhart RC. A discrete binary version of the particle swarm algorithm. In: Kennedy J, editor. 1997 IEEE international conference on systems, man, and cybernetics. Computational cybernetics and simulation, vol. 5. Orlando: IEEE; 1997. p. 4104–8.
47. Mirjalili S, Lewis A. S-shaped versus V-shaped transfer functions for binary particle swarm optimization. Swarm Evol Comput. 2013;9:1–14.
48. Zhang X, Wu C, Li J, Wang X, Yang Z, Lee J-M, Jung K-H. Binary artificial algae algorithm for multidimensional knapsack problems. Appl Soft Comput. 2016;43:583–95.
49. Mirjalili S, Zhang H, Mirjalili S, Chalup S, Noman N. A novel u-shaped transfer function for binary particle swarm optimisation. In: Nagar A, Deep K, Bansal J, Das K, editors. Soft Computing for Problem Solving 2019: Proceedings of SocProS 2019, vol. 1. Singapore: Springer; 2020. p. 241–59.
50. Ghosh KK, Singh PK, Hong J, Geem ZW, Sarkar R. Binary social mimic optimization algorithm with x-shaped transfer function for feature selection. IEEE Access. 2020;8:97890–906.
51. Rashedi E, Nezamabadi-Pour H, Saryazdi S. Bgsa: binary gravitational search algorithm. Natural Comput. 2010;9:727–45.
52. Syswerda G. Simulated crossover in genetic algorithms. In: Whitley LD, editor. Foundations of genetic algorithms, vol. 2. Amsterdam: Elsevier; 1993. p. 239–55.
53. Keller JM, Gray MR, Givens JA. A fuzzy k-nearest neighbor algorithm. IEEE Trans Syst Man Cybern. 1985;4:580–5.
54. Bishop CM, et al. Neural networks for pattern recognition. Oxford: Oxford University Press; 1995.
55. Russell SJ, Norvig P. Instructor's solution manual for artificial intelligence: a modern approach. London: Pearson Education Company; 2003.
56. Ding S, Zhang X, An Y, Xue Y. Weighted linear loss multiple birth support vector machine based on information granulation for multi-class classification. Pattern Recogn. 2017;67:32–46.
57. Rokach L. Decision forest: twenty years of research. Inform Fusion. 2016;27:111–25.
58. Denoeux T. A k-nearest neighbor classification rule based on dempster-shafer theory. Classic works of the Dempster-Shafer theory of belief functions. IEEE Transact Syst Man Cybern. 2008;737–760.
59. Qin Z, Wang AT, Zhang C, Zhang S. Cost-sensitive classification with k-nearest neighbors. In: Wang M, editor. Knowledge Science, Engineering and Management: 6th International Conference, KSEM 2013, Dalian, China, August 10-12, 2013. Proceedings, vol. 6. Berlin: Springer; 2013. p. 112–31.
60. Shakhnarovich G, Darrell T, Indyk P. Nearest-neighbor methods in learning and vision. IEEE Trans Neural Netw. 2008;19(2):377.
61. Allam M, Nandhini M. Optimal feature selection using binary teaching learning based optimization algorithm. J King Saud Univ Comput Inform Sci. 2022;34(2):329–41.
62. Braik M. Enhanced Ali baba and the forty thieves algorithm for feature selection. Neural Comput Appl. 2022. https://doi.org/10.1007/s00521-022-08015-5.
63. Khurma RA, Albashish D, Braik M, Alzaqebah A, Qasem A, Adwan O. An augmented snake optimizer for diseases and COVID-19 diagnosis. Biomed Signal Process Control. 2023;84: 104718.
64. Wu G, Mallipeddi R, Suganthan PN. Problem definitions and evaluation criteria for the cec 2017 competition on constrained real-parameter optimization. National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore, Technical Report 2017.
65. Pereira DG, Afonso A, Medeiros FM. Overview of Friedman's test and post-hoc analysis. Commun Statistics Simul Comput. 2015;44(10):2636–53.
66. Demšar J. Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res. 2006;7:1–30.
67. Holm S. A simple sequentially rejective multiple test procedure. Scand J Stat. 1979;6:65–70.

Publisher's Note