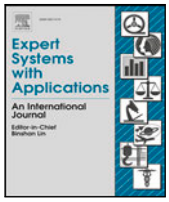
Expert Systems With Applications 246 (2024) 123214

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa



Multi-strategy differential evolution algorithm based on adaptive hash clustering and its application in wireless sensor networks ☆

Xianglong Bu, Qingke Zhang *, Hao Gao, Huaxiang Zhang

School of Information Science and Engineering, Shandong Normal University, Jinan 250358, China

ARTICLE INFO

Keywords:
Differential evolution algorithm
Population clustering
Multiple mutation strategies
Population state evaluation
Wireless sensor networks

ABSTRACT

Population-based algorithms aim to explore the entire solution space in global numerical optimization problems. However, it is important to acknowledge that the solution spaces of different problems possess distinct characteristics, and even distinct regions within the same solution space can vary significantly. Efficient exploration of these diverse regions necessitates the utilization of distinct search models. To address this challenge, this study proposes a new variant of the Differential Evolution (DE) algorithm called MHDE. The MHDE algorithm introduces hash clustering technology combined with adaptive mutation strategies. The integration of hash clustering technology enables fast population clustering, significantly enhancing clustering efficiency. Additionally, a method for evaluating the population state is designed that allows for an adaptive clustering quantity to adapt the clustering quantity to the population state. Furthermore, a novel method is devised to calculate individual improvements considering the varying levels of difficulty that individuals face in achieving improvements within a population. This method is combined with a parameter adaptive mechanism, resulting in a weighted parameter adaptive mechanism. Experiments are conducted on the CEC2017 benchmark suite to evaluate the performance of the MHDE algorithm. The experimental results demonstrate that MHDE exhibits competitive performance compared with other efficient DE variants. Moreover, the MHDE algorithm is applied to the node deployment problem in wireless sensor networks (WSNs). The MHDE algorithm demonstrates efficient performance in node deployment problems through simulation experiments in different scenarios.

1. Introduction

Differential Evolution (DE) is a population-based optimizer proposed by Storn and Price in 1997 (Storn & Price, 1997) that has received considerable attention in the past two decades owing to its efficient performance. In addition, it has been widely applied to real-world problems, such as wind speed distribution parameter estimations (Kumar & Kumar, 2021), intelligent transportation (Chen et al., 2022), power electronic circuits (Liu et al., 2022; Zhan et al., 2017), flow-shop scheduling (Zhao et al., 2020), and anomaly detection (Guendouzi & Boukra, 2022).

With the development of society and technology, the single-search model in traditional DE can no longer cope with the increasing diversity and complexity of problems (Zhang et al., 2023b). The mutation and crossover operations in DE instruct the search model of the population, whereas specific search models are only efficiently optimized for specific types of problems (Mohamed et al., 2019). Therefore, DE variants with multiple search models have been proposed to improve the comprehensive optimization capability of the DE on different types of problems. For example, Qin et al. (Qin et al., 2009) proposed the SaDE algorithm, which constructs a pool of four mutation strategies and calculates the selection probabilities of different strategies using success and failure memories of the generations. Zhang et al. proposed the JADE algorithm (Zhang & Sanderson, 2009), which extracts recent successful parameters to guide the setting of future parameters while generating F and CR parameters for each individual using normal and Cauchy distributions. Mallipeddi et al. proposed the EPSDE algorithm (Mallipeddi et al., 2011), whereby different mutation strategies and each control parameter coexist and compete during the entire operation to produce offspring. Wang et al. introduced the CoDE algorithm (Wang et al., 2011), which generates multiple offspring by randomly combining three mutation strategies with three control parameter methods and allows the optimal offspring to join the next generation population.

☆ This work is supported by the National Natural Science Foundation of China (No. 62006144), the Major Fundamental Research Project of Shandong, China (No. ZR2019ZD03), and the Taishan Scholar Project of Shandong, China (No. ts20190924).

* Corresponding author.
E-mail addresses: bullong@foxmail.com (X. Bu), tsingke@sdnu.edu.cn (Q. Zhang), Gaohao@sdnu.edu.cn (H. Gao), huaxzhang@sdnu.edu.cn (H. Zhang).

https://doi.org/10.1016/j.eswa.2024.123214
Received 22 June 2023; Received in revised form 22 December 2023; Accepted 9 January 2024
Available online 16 January 2024

Nomenclature

List of Abbreviations

ϵ	A Very Small Positive Number
$\vec{CS}$	Set of cs_c
$\vec{HS}$	Ordered Hash Value Vector
$\vec{H}$	Hash Value Vector
$\vec{LB}$	Lower bound of the solution space
$\vec{O}$	Projected Line Vector
$\vec{UB}$	Upper bound of the solution space
Bp	Basic Selection Probability of Mutation Strategies
CLU	Set of clu_c
clu_c	Index of Members in the cth cluster
CR	Crossover Probability
cs_c	The Size of the cth Cluster
D	Problem Dimension
e	Proportion of the Extra-cluster
E_G	The Entropy Value of Elite Distribution in Gth Generation
F	Scale Factor
$f(\vec{X})$	Fitness value of the $\vec{X}$
FEs	The Number of Function Evaluations
G	Generation of the Population
h	Wilcoxon test result
$I^c_{m,G}$	The Accumulative Improvement of the mth Strategy in the cth Cluster
$MaxFEs$	The Maximum Number of Function Evaluations
NC	The Number of Clusters
NC_{max}	Upper limit of the number of clusters
NC_{min}	Lower limit of the number of clusters
NE	The Number of Elites
NP	Population Size
NS	EcPop Size
$R_{i,G}$	The improvement of the ith individual in Gth Generation
$randc$	Cauchy Distribution
$randn$	Gaussian Distribution
SCR	Set of scr
scr	CR value that enables individuals to successfully evolve
SF	Set of sf
sf	F value that enables individuals to successfully evolve
TS	Time Span for the MHDE algorithm reclustering
BLSH	Balanced Locality-Sensitive Hashing
DE	Differential Evolution Algorithm
EcPop	Extra-cluster Subpopulation
f-rank	Friedman test result
FDC	Fitness-Distance Correlation
LSH	Locality-Sensitive Hashing
MHDE	Multi-strategy Differential Evolution Algorithm based on Adaptive Hash Clustering
overall f-rank	Summation of f-rank Across All Test Functions
SCE	Shuffled Complex Evolution Partitioning
std	Standard Deviation
WSNs	Wireless Sensor Networks

The aforementioned variants of the DE algorithm are modifications based on a single population. In addition to these modifications, another approach to enhance algorithm performance is the incorporation of multi-population techniques into the DE algorithm. For example, Gui et al. proposed the MPADE algorithm that was inspired by the concept of work specialization (Cui et al., 2016). The algorithm divides the entire population into three subpopulations based on fitness values, then different subpopulations perform different tasks. Wu et al. proposed the MPEDE algorithm (Wu et al., 2016), which assigns a smaller indicator subpopulation to each mutation strategy and a larger reward subpopulation to the best-performing mutation strategy after a certain number of generations to improve population search efficiency. In 2018, Wu et al. further proposed an EDEV algorithm based on MPEDE (Wu et al., 2018), which uses three excellent DE variants, JADE, CoDE, and EPSDE, to replace the three mutation strategies in MPEDE. This approach exhibited a much better overall optimization strength.

The above DE variants either do not split the subpopulations, split them randomly, or split them according to the fitness value. However, none of these methods considered the topology of the populations in the solution space. Population-based algorithms operate on the principle of exploring a solution space through a population. However, different regions within the solution space of the same problem can exhibit distinct characteristics. Therefore, an efficient search in different regions requires adopting different search patterns. To address this challenge, this study proposes the MHDE algorithm, which includes a balanced clustering technique. This technique not only achieves balanced coverage of different regions but also improves upon the time-consuming nature of traditional clustering methods. Additionally, each cluster is equipped with an independent adaptive mechanism to facilitate an adaptive search within the cluster coverage area. Furthermore, a population state evaluation method is proposed to ensure that the clustering quantity aligns with the current population state. Finally, this study introduces a new method for calculating individual improvements and combines this method with parameter mechanisms.

The four main contributions regarding the MHDE algorithm proposed in this study are summarized as follows.

1. A balanced clustering technique based on Locality-Sensitive Hashing (LSH) was designed to achieve a balanced and fast clustering of the population.
2. A population state evaluation method was introduced to regulate the clustering quantity.
3. A novel approach for calculating the individual improvement was presented to improve adaptive parameter mechanisms.
4. An external population was leveraged to accelerate the convergence speed of the algorithm.

2. Related works

This section provides a brief overview of the representative strategies relevant to this study. These strategies primarily include clustering mechanisms in the DE, population partitioning technologies, population state evaluation mechanisms, and adaptive F and CR mechanisms.

2.1. Clustering in DE

Clustering is a data grouping technique in which data within the same cluster exhibit higher similarity to each other than to other clusters. Clustering is an effective data aggregation method that has been widely applied in various domains, such as electricity consumption forecasting (Hadjout et al., 2023) and networking (Kumar et al., 2023). Additionally, clustering mechanisms have been applied to enhance the optimization capabilities of DE.

Cai et al. (Cai et al., 2011) utilized one-step k-means clustering to generate offspring, making full use of population information to

achieve a balance between exploration and exploitation during population evolution. Neri et al. (Poikolainen et al., 2015) applied clustering mechanisms to the population initialization of DE algorithms, using k-means clustering to assess dominant regions within the solution space, thereby improving the coverage of dominant regions by the initial population. Liang et al. (Liang et al., 2021) employed clustering techniques to identify neighborhood relationships and calculated the comprehensive crowding distance of individuals in the decision and objective spaces. The crowding distance was used to guide the exemplar and environmental selection of the algorithm. Zhang et al. (Zhang et al., 2017) utilized the LSH technique to achieve rapid population partitioning. This approach circumvents the high computational complexity associated with calculating the distances between individuals, thereby enabling the localized evolution of individuals.

2.2. Population partitioning

Population partitioning is the process of dividing a population into smaller subpopulations that can move towards different dominant regions, thereby maintaining population diversity and helping the algorithm escape local optima (Chaudhary & Banati, 2020). This section introduces classical population-partitioning methods.

Random partitioning (Al-Betar & Awadallah, 2018) is a technique that divides all individuals randomly and evenly into a fixed number of subpopulations. This is the most commonly used population partitioning technique. A dynamic subpopulation number strategy was proposed in Xia et al. (2018), in which the number of subpopulations was larger in the early stages of the algorithm and gradually decreased to one as the algorithm progressed. This technique provides the population with greater exploratory ability in the early stages and focuses on exploitation in the later stages. The shuffled complex evolution (SCE) partitioning technique was proposed in Banati and Chaudhary (2016). This technique sorts individuals based on their fitness values and then assigns the first individual to the first subpopulation, the second individual to the second subpopulation, and so on until one individual is added to the last subpopulation. The next individual is then assigned to the first subpopulation, and the process is repeated until the population is fully partitioned. SCE ensures that each subpopulation contains individuals at different levels, thus avoiding inefficient searches owing to some subpopulations consisting entirely of inferior individuals. Hierarchical partitioning (Toledo et al., 2013) divides a population into different subpopulations based on their fitness values, with superior individuals grouped into the same subpopulation and inferior individuals grouped into different subpopulations. This approach provides different subpopulations with various functions, with superior subpopulations focusing on exploiting dominant regions, and inferior subpopulations responsible for global exploration and maintaining population diversity. Hongru et al. (Hongru et al., 2018) designed a population partitioning technique based on k-means clustering that divided the entire population into non-overlapping subpopulations (clusters). Clustering can group individuals in the same region into the same cluster, which can help the population to better search for different dominant regions. Wang et al. (Wang et al., 2021) proposed using the LSH technique for population clustering to address large-scale optimization problems, which can significantly improve the clustering efficiency of the population.

2.3. Population state evaluation

For black-box optimization, researchers are unaware of the characteristics of the optimization problem and therefore cannot choose appropriate algorithms and strategies to address the challenges of black-box optimization. Therefore, using the population state to help the algorithm select appropriate search patterns has become an effective means of improving black-box optimization (Zhang et al., 2023a).

Yu et al. (Yu et al., 2014) proposed an indicator of the optimization state (IOS), that evaluated the population state by calculating the difference between individuals in the solution and objective spaces. Zhan et al. (Zhan et al., 2020) designed an evolution factor (f) that specifically determined the population state by calculating the Euclidean distance between optimal and median individuals. Li et al. (Li & Li, 2020) used the fitness-distance correlation (FDC) to evaluate the population state. This was accomplished by assessing the correlation between the population distribute in the objective and solution spaces. Tan et al. introduced two dynamic fitness landscape analysis methods in their study (Tan et al., 2022). The first method used a random walk algorithm combined with FDC to propose a dynamic FDC, and the second involved the dynamic ruggedness of information entropy. These two methods were used in this study to jointly evaluate the population state. Li et al. (Li et al., 2023) proposed a framework for DE that addressed issues of population prematurity and stagnation. This was achieved by combining two types of information: changes in the optimal solution of the current population and the population state. By combining these, the framework effectively alleviated the problems of premature convergence and population stagnation. Meng (Meng, 2023) designed a method for evaluating population states by detecting changes in population diversity. If the population state indicates that the current population is stagnant, a population promotion mechanism is triggered to prevent the population from continuously remaining stagnant.

2.4. Adaptive F and CR

F and CR are two important parameters that significantly affect the performance of the DE algorithm. F affects the range of population mutations, whereas CR determines the crossover ratio between the trial and original individuals. Therefore, adaptive F and CR are common practices for improving the algorithm performance.

In the JADE algorithm (Zhang & Sanderson, 2009), all F and CR values that lead to successful evolution of individuals in one iteration are saved in $SCR = \{scr_1, scr_2, \ldots\}$ and $SF = \{sf_1, sf_2, \ldots\}$ sets, respectively. Subsequently, new F and CR values are respectively obtained using Eqs. (1) and (2).

$$\begin{cases} mean_L(SF) = \frac{\sum_{sf \in SF} sf^2}{\sum_{sf \in SF} sf} \\ \mu F = (1-c) \cdot \mu F + c \cdot mean_L(SF) \\ F = randc(\mu F, 0.1) \end{cases} \tag{1}$$

where $mean_L(SF)$ denotes the Lehmer mean of SF, $randc$ represents the Cauchy distribution.

$$\begin{cases} mean_A(SCR) = \frac{\sum_{scr \in SCR} scr}{|SCR|} \\ \mu CR = (1-c) \cdot \mu CR + c \cdot mean_A(SCR) \\ CR = randn(\mu CR, 0.1) \end{cases} \tag{2}$$

where $mean_A(SCR)$ denotes the mean of SCR, $randn$ represents the Gaussian distribution, and $|SCR|$ denotes the size of the set SCR.

The SHADE algorithm (Tanabe & Fukunaga, 2013) proposes a new method for calculating F and CR based on the JADE parameter mechanism. In addition to saving successful CR and F values, the improvement $\Delta f_i = f\left(\vec{X}_i\right) - f\left(\vec{U}_i\right)$ of individuals is also saved in set $R = \{\Delta f_1, \Delta f_2, \ldots\}$, and used as a weight to calculate μCR and μF. Thus, the parameters that lead to greater individual improvement receive more attention. The calculation processes are expressed in Eqs. (3) and (4).

$$\begin{cases} w_k = \frac{\Delta f_k}{\sum_{k=1}^{|SF|} \Delta f_k} \\ mean_{WL}(SF) = \frac{\sum_{k=1}^{|SF|} w_k \cdot sf_k^2}{\sum_{k=1}^{|SF|} w_k \cdot sf_k} \\ \mu F = \begin{cases} mean_{WL}(SF), \text{ if } SF \neq \varnothing \\ \mu F, otherwise \end{cases} \\ F = randc(\mu F, 0.1) \end{cases} \tag{3}$$

$$\begin{cases} w_k = \frac{\Delta f_k}{\sum_{k=1}^{|SF|} \Delta f_k} \\ mean_{WA}(SCR) = \sum_{k=1}^{|SCR|} w_k \cdot scr_k \\ \mu CR = \begin{cases} mean_{WA}(SCR), \text{ if } SCR \neq \varnothing \\ \mu CR, otherwise \end{cases} \\ CR = randn(\mu CR, 0.1) \end{cases} \tag{4}$$

The PaDE_pet algorithm (Meng, 2023) proposes dimension improvement-based adaptation schemes for control parameters, which use the standard deviation(std) of good changes in individual dimensions as a weight to adjust the calculation process of μF and μCR. Therefore, in PaDE_pet, it is necessary to store the dimensional changes $\Delta loc_i = loc\left(\vec{U}_i - \vec{X}_i\right)$ of successful individuals in a set ΔLOC. The specific calculation methods for F and CR are respectively shown in Eqs. (5) and (6).

$$\begin{cases} w_i = \frac{std(\Delta loc_i)}{\sum_{i=1}^{|\Delta LOC|} std(\Delta loc_i)} \\ mean_{WL}(SF) = \frac{\sum_{i=1}^{|SF|} w_i \cdot sf_i^2}{\sum_{i=1}^{|SF|} w_i \cdot sf_i} \\ \mu F = \begin{cases} mean_{WL}(SF), \text{ if } SF \neq \varnothing \\ \mu F, otherwise \end{cases} \\ F = randc(\mu F, 0.1) \end{cases} \tag{5}$$

$$\begin{cases} w_i = \frac{std(\Delta loc_i)}{\sum_{i=1}^{|\Delta LOC|} std(\Delta loc_i)} \\ mean_{WL}(SCR) = \frac{\sum_{i=1}^{|SCR|} w_i \cdot scr_i^2}{\sum_{i=1}^{|SCR|} w_i \cdot scr_i} \\ \mu CR = \begin{cases} mean_{WL}(SCR), \text{ if } SCR \neq \varnothing \\ \mu CR, otherwise \end{cases} \\ CR = randn(\mu CR, 0.1) \end{cases} \tag{6}$$

3. Differential evolution

The DE algorithm was designed to iteratively evolve the population towards the global optimum. It is comprised of four essential components: population initialization, mutation, crossover, and selection operations.

3.1. Initialization

The population in the Gth generation $P^G = \{\vec{X}_1^G, \vec{X}_2^G, \dots, \vec{X}_{NP}^G\}$ is composed of NP individuals, where each individual is a D-dimensional decision vector. For example, the ith individual is denoted as $\vec{X}_i^G = [x_{i,1}^G, x_{i,2}^G, \dots, x_{i,D}^G]$. To ensure that the population covers the solution space, the upper and lower $\vec{UB} = [ub_1, ub_2, \dots, ub_D]$ and $\vec{LB} = [lb_1, lb_2, \dots, lb_D]$ bounds of the solution space need to be obtained before initializing the population. Then the initialized population ($G = 0$) is generated by Eq. (7).

$$x_{i,j}^0 = lb_j + rand(0,1) \cdot (ub_j - lb_j) \tag{7}$$

where $rand(0,1)$ denotes a random real number between 0 and 1.

3.2. Mutation

In the mutation phase, each target vector $\vec{X}_i^G$ (so-called individual) generates a mutant vector $\vec{V}_i^G = [v_{i,1}^G, v_{i,2}^G, \dots, v_{i,D}^G]$, and the mutation strategy is expressed in Eq. (8).

$$\vec{V}_i^G = \vec{X}_{r1}^G + F \cdot (\vec{X}_{r2}^G - \vec{X}_{r3}^G) \tag{8}$$

where $\vec{X}_{r1}^G$, $\vec{X}_{r2}^G$ and $\vec{X}_{r3}^G$ are three random individuals in the current population and $r1 \neq r2 \neq r3$, $F \in (0, 1]$ denotes the scale factor used to adjust the mutation step.

3.3. Crossover

There are two main types of crossover operations: binomial and exponential. This study used a binomial crossover, therefore, this section details the binomial crossover operations.

Each mutant vector $\vec{V}_i^G$ was blended with the corresponding target vector $\vec{X}_i^G$ to generate the trial vector $\vec{U}_i^G = [u_{i,1}^G, u_{i,2}^G, \dots, u_{i,D}^G]$. The specific crossover operation is expressed in Eq. (9).

$$u_{i,j}^G = \begin{cases} v_{i,j}^G & rand(0,1) < CR \text{ or } j = j_{rand} \\ x_{i,j}^G & otherwise \end{cases} \tag{9}$$

where $CR \in [0, 1]$ denotes the crossover probability and j_{rand} denotes a random integer between 1 and D.

The trial vectors generated by the crossover operation may move outside the search region; thus, the elements of the trial vectors that are out of bounds are regenerated using Eq. (7).

3.4. Selection

The goal of the DE algorithm is to drive the evolution of the population, that is to drive the continuous improvement of the fitness values of each individual. Therefore, the trial vectors need to calculate the fitness values. Subsequently, the fitness values of the trial vectors are compared with the corresponding fitness values of the target vectors so that the vectors with better fitness values can enter the next generation of the population. The specific selection operation is presented in Eq. (10).

$$\vec{X}_i^G = \begin{cases} \vec{U}_i^G & f(\vec{U}_i^G) < f(\vec{X}_i^G) \\ \vec{X}_i^G & otherwise \end{cases} \tag{10}$$

where $f(\vec{X})$ denotes the fitness value of the vector $\vec{X}$.

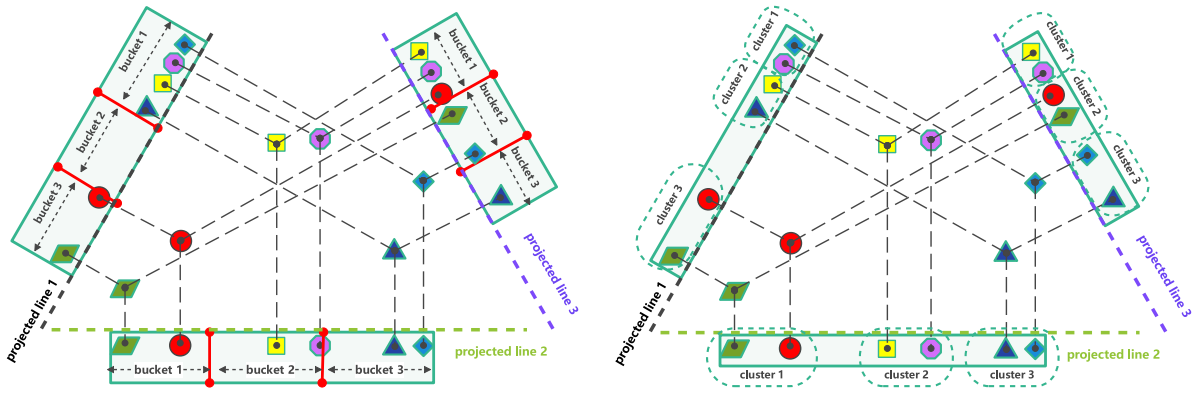
4. Locality-sensitive hashing (LSH)

LSH is an efficient clustering technique (Zhang et al., 2017), whereby individuals in the original data space that are close to each other are highly likely to remain close to each other, even after being mapped to a projected line through hashing operations. In LSH, there is a "bucket" concept where the projected line is divided into several segments of length r. Each segment is referred to as a bucket, and individuals within the same bucket are assigned to the same cluster. The specific computational process of the LSH within the context of the DE algorithm is presented in Algorithm 1.

Algorithm 1: Pseudocode for the LSH

Input: Population $P^G = \{X_1^G, \cdots, X_{NP}^G\}$; Clustering quantity control parameter nb
Output: Index vector $\vec{B}$ of the buckets

Generate a vector $\vec{O}$ with the same dimension as X_i^G
Perform dot product between $\vec{X}_i^G$ and $\vec{O}$:
for $i = 1$ *to* NP **do**
  $\vec{H} \leftarrow h_i = \vec{X}_i^G \cdot \vec{O}^T$;
end
$h_{max} = MAX(\vec{H}), h_{min} = MIN(\vec{H})$
Calculate the size r of the bucket:
$r = (h_{max} - h_{min})/nb$
Calculate the bucket number of each individual:
for $i = 1$ *to* NP **do**
  $\vec{B} \leftarrow b_i = \lfloor (h_i + b)/r \rfloor, b = rand(0, r)$
end



(a) Locality-sensitive hashing (b) Balanced locality-sensitive hashing

Fig. 1. Demonstration of LSH and BLSH in 2D space.

A simple illustration of the use of LSH to divide individuals from a 2D Euclidean space into three clusters is shown in Fig. 1(a). Here, six different graphs represent individuals distributed in 2D space. The six individuals are mapped onto three different projected lines and divided into three buckets of the same size.

5. MHDE algorithm

The proposed MHDE algorithm can be broadly divided into three main steps: population clustering, intra-cluster evolution, and extra-cluster evolution. Detailed descriptions of the three components of the MHDE algorithm are presented in the following sections.

5.1. Population clustering

Search regions for the same problem can be different, so different regions require different search models.

This study introduces a novel DE variant called MHDE that integrates clustering techniques into the DE algorithm to address efficient search requirements across different regions. Furthermore, a method for evaluating the population states was designed to adaptively adjust the number of clusters in the population.

5.1.1. Balanced locality-sensitive hashing

LSH technology significantly enhances the clustering efficiency of the population. However, the LSH cannot guarantee that each cluster will contain at least the minimum number of individuals required for a DE search, as shown in Fig. 1(a). For example, at least three individuals were required to perform the mutation operation in Eq. (8). Moreover, the efficiency of the algorithm is significantly affected if there are too many or few individuals within a cluster. Therefore, Balanced LSH (BLSH) was designed based on LSH.

BLSH replaces buckets with a sequential selection of individuals to ensure a balanced distribution of individuals across different clusters. First, the population is mapped to hash values, assuming that the population must be clustered into NC clusters, with $cs = NP/NC$ individuals in each cluster. Second, the hash values are sorted and the first cs individuals are clustered into the same cluster. The next cs individuals are clustered into the second cluster, and so on until all individuals are clustered into NC clusters. The calculation process is shown in Algorithm 2. The clustering of the six individuals into three clusters using BLSH on the projected lines is shown in Fig. 1(b).

The MHDE algorithm re-clusters after a certain number of iterations, called the time span TS, to avoid overlapping search regions between clusters owing to population evolution.

In addition to performing population clustering, LSH also has the capability to determine the convergence of the population. In Algorithm 2, the population has converged when the $flag$ is 0 for multiple

Algorithm 2: Pseudocode for the BLSH

Input: Population $P^G = \{\vec{X}_1^G, \cdots, \vec{X}_{NP}^G\}$; Clustering quantity NC_G
Output: Size of each cluster $\vec{CS}$;Cluster set CLU clustered by BLSH

Generate a vector $\vec{O}$ with the same dimension as $\vec{X}_i$
Perform dot product between $\vec{X}_i$ and $\vec{O}$:
for $i = 1$ *to* NP **do**
  $\vec{H} \leftarrow h_i = \vec{X}_i^G \cdot \vec{O}^T$;
end
$\vec{HS} = sort(\vec{H})$;
$flag = \vec{HS}[1] - \vec{HS}[NP]$;
$basicCS = \lfloor NP/NC_G \rfloor$;
$rem = NP\%NC_G$;
for $c = 1$ *to* NC_G **do**
  if $c <= rem$ **then**
    $\vec{CS} \leftarrow cs_c = basicCS + 1$
  else
    $\vec{CS} \leftarrow cs_c = basicCS$
  end
  $clu_c \leftarrow \vec{HS}[1 : cs_c]$;
  $CLU \leftarrow clu_c$;
  $\vec{HS} = \vec{HS}[cs_c + 1 : NP]$;
end

calculations with different vectors $\vec{O}$. Premature convergence of the population can lead to wasted computational resources, necessitating a population restart. Therefore, all individuals except the current best are regenerated using Eq. (11).

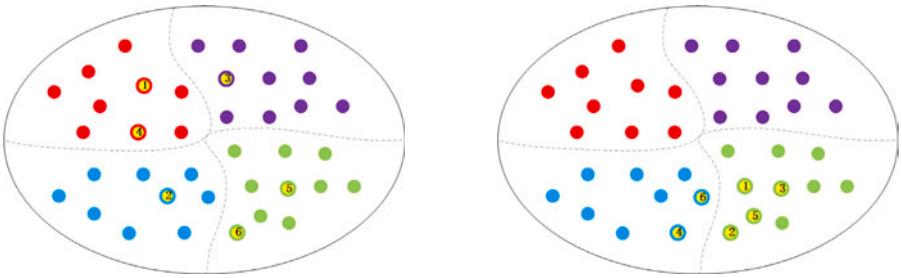
$$\vec{X}_i^G = randc\left(\vec{X}_i^G, \left(\vec{UB} - \vec{LB}\right)\left(1 - \frac{FEs}{MaxFEs}\right)\right) \tag{11}$$

where FEs and $MaxFEs$ represent the current and the maximum number of function evaluations, respectively.

5.1.2. Adaptive clustering quantity

A larger number of clusters is effective in maintaining the diversity of a population but can also increase the blindness of the population search (Yang et al., 2022) thus reducing its efficiency. A smaller number of clusters increases the convergence speed of the population but reduces the significance of population partitioning (Wang et al., 2021). Therefore, the number of clusters NC_G is a key factor that affects the performance of the algorithm.

The better the individual, the higher the probability of approaching the global optimum. Therefore, a population with a concentration of



(a) Scattered distribution of elites (b) Concentrated distribution of elites

Fig. 2. Distribution of elites in clusters.

dominant individuals indicates a concentration of dominant regions in the solution space. Consequently, the number of clusters should be reduced to enhance the exploitation of the dominant regions. Conversely, if the dominant individuals are dispersed, the number of clusters must be increased to improve the ability of the population to explore globally. For ease of presentation, the remainder of this paper refers to the top NE_G dominant individuals as elites.

Entropy expresses the degree of disorder within a system (Gray, 2011), allowing for the evaluation of the degree of disorder in the elites by calculating the entropy value of their distribution in different clusters. The entropy value is calculated as shown in Eq. (12).

$$E_G = -\sum_{n=1}^{Z} \frac{k_n^G}{NE_G} \log \frac{k_n^G}{NE_G} \tag{12}$$

where Z denotes the number of clusters containing elites, k_n^G denotes the number of elites in the nth cluster, and NE_G denotes the number of elites.

The two distributions of elites are shown in Fig. 2, where the numbered dots represent the elites of the population. The entropy values of the two elite distributions are expressed in Eq. (13) and (14), respectively.

$$E_G = -\left(\frac{2}{6}\log\frac{2}{6}\right) - \left(\frac{1}{6}\log\frac{1}{6}\right) - \left(\frac{1}{6}\log\frac{1}{6}\right) - \left(\frac{2}{6}\log\frac{2}{6}\right) = 1.918 \tag{13}$$

$$E_G = -\left(\frac{2}{6}\log\frac{2}{6}\right) - \left(\frac{4}{6}\log\frac{4}{6}\right) = 0.9183 \tag{14}$$

Eq. (13) and (14) indicate that the entropy value is higher when the elite distribution is more chaotic, and lower when the distribution is more regular. This observation inspired the design of the regulation mechanism for NC_G, which is calculated as Eq. (15).

$$NC_{G+1} = \begin{cases} NC_G + 1 & E_G > ET_G \ and \ NC_G < NC_{max} \\ NC_G - 1 & E_G \le ET_G \ and \ NC_G > NC_{min} \\ NC_G & otherwise \end{cases} \tag{15}$$

where NC_{max} and NC_{min} indicate the upper and lower limits of the number of clusters, respectively, and ET_G is the entropy value when the elites are evenly distributed within the $round(NC_G/2)$ clusters. The elites are distributed in more clusters then NT_G, if E_G is greater than ET_G. In other words, the distribution of the elite is more chaotic and the number of clusters needs to be expanded. Conversely, the number of clusters decreases when the distribution is more regular.

The entropy value calculated using Eq. (12) is influenced by the cluster quantity NC_G and number of elites NE_G. Therefore, NE_G was set to equal NC_G to achieve a dynamic balance between NC_G and NE_G.

The clustering method employed for this phase was LSH to enhance clustering efficiency.

5.2. Intra-cluster evolution

Each cluster evolves independently within a TS to efficiently search for its own coverage area. The following section introduces the details of intra-cluster evolution with the background of the cth cluster, as each cluster uses the same evolution mechanism. The superscript of the parameter is used as a marker for the clusters.

5.2.1. Calculating individual improvement

Before discussing the mutation strategies and parameter settings, a novel method for calculating individual improvements must be introduced. This method is integrated into the calculation of the mutation strategy selection probabilities and parameters.

The goal of each individual in the population is to improve the fitness value. Inferior individuals can improve themselves by learning from superior individuals, but superior individuals have fewer opportunities for such learning. Therefore, the improvement of superior individuals should receive more attention. Based on the above analysis, this study proposes a new method for calculating individual improvements, which is presented in the Eq. (16).

$$R_{i,G} = \begin{cases} \left(f(\vec{X}_i^G) - f(\vec{U}_i^G)\right) \frac{f_{\max}^G - f(\vec{X}_i^G) + \epsilon}{f_{\max}^G - f_{min}^G + \epsilon} & f(\vec{U}_i^G) < f(\vec{X}_i^G) \\ 0 & otherwise \end{cases} \tag{16}$$

where f_{max}^G and f_{min}^G denote the worst and best fitness values in the Gth population, respectively, and ϵ denotes an extremely small positive real number.

5.2.2. Multiple mutation strategies

Mutation plays a crucial role in the DE algorithm because it determines the search pattern of the population. In the MHDE algorithm, individuals participating in the same mutation originate from the same cluster. This prevents the search areas of each cluster from overlapping with those of the other clusters, thereby improving the search efficiency of the entire population.

For each cluster, the mutation pool is comprised of four mutation strategies assuming different functions. These mutation strategies are expressed as:

- DE/current-to-best/1

$$\vec{V}_{i,G}^c = \vec{X}_{i,G}^c + F_{i,G}^c \cdot \left(\vec{X}_{b,G}^c - \vec{X}_{i,G}^c\right) + F_{i,G}^c \cdot \left(\vec{X}_{r1,G}^c - \vec{X}_{r2,G}^c\right)$$

- DE/rand-to-best/1

$$\vec{V}_{i,G}^c = \vec{X}_{r1,G}^c + F_{i,G}^c \cdot \left(\vec{X}_{b,G}^c - \vec{X}_{r1,G}^c\right) + F_{i,G}^c \cdot \left(\vec{X}_{r2,G}^c - \vec{X}_{r3,G}^c\right)$$

- DE/current-to-rand/1

$$\vec{V}_{i,G}^c = \vec{X}_{r1,G}^c + F_{i,G}^c \cdot \left(\vec{X}_{r1,G}^c - \vec{X}_{i,G}^c\right) + F_{i,G}^c \cdot \left(\vec{X}_{r2,G}^c - \vec{X}_{r3,G}^c\right)$$

- DE/rand/1

$$\vec{V}_{i,G}^c = \vec{X}_{r1,G}^c + F_{i,G}^c \cdot \left(\vec{X}_{r2,G}^c - \vec{X}_{r3,G}^c\right)$$

where $\vec{X}_{r1,G}^c$, $\vec{X}_{r2,G}^c$, and $\vec{X}_{r3,G}^c$ denote random members of the cth cluster and $r1 \neq r2 \neq r3 \neq i$, $\vec{X}_{b,G}^c$ denotes the best member of the cth cluster.

In the mutation pool, DE/current-to-best/1 and DE/rand-to-best/1 are used to help the population converge quickly towards the dominant region and perform extremely well when dealing with unimodal

Table 1
Individual improvement in cth cluster.

m \ i	1	2	⋯	cs_c-1	cs_c
DE/current-to-best/1	0	$R^c_{2,1,G}$	⋯	0	0
DE/rand-to-best/1	$R^c_{1,2,G}$	0	⋯	0	0
DE/current-to-rand/1	0	0	⋯	0	$R^c_{cs_c,3,G}$
DE/rand/1	0	0	⋯	$R^c_{cs_c-1,4,G}$	0

functions. DE/current-to-rand/1 and DE/rand/1 provide the population with more power to explore. As such, they are used to address complex multimodal problems.

Each mutation strategy within the current cluster is assigned a selection probability p, and the mutation strategy for each individual is selected using a roulette-wheel selection mechanism based on these selection probabilities. Thus, the effectiveness of a cluster search heavily depends on selection probabilities.

The cumulative improvement and times that the strategy is selected are used to efficiently set the selection probability. Setting selection probabilities based on individual improvements is an efficient approach. The distribution structure of the individual improvements within the cth cluster is provided in Table 1. Here, m and i represent the identifiers of the mutation strategies and individuals, respectively, and cs_c represents the number of individuals in the cth cluster.

The selection probability $p^c_{m,G}$ of the mth mutation strategy is expressed in Eqs. (17).

$$\begin{cases} I^c_{m,G} = \sum_{i=1}^{cs_c} R^c_{i,m,G} \\ M^c_{m,G} = \frac{I^c_{m,G-1}}{N^c_{m,G-1}+\epsilon} \\ p^c_{m,G} = \frac{Bp}{4} + (1-Bp)\frac{M^c_{m,G}+\epsilon}{\sum_{m=1}^{4}\left(M^c_{m,G}+\epsilon\right)} \end{cases} \tag{17}$$

where $I^c_{m,G}$ denotes the cumulative improvement of the mth strategy in the cth cluster, Bp denotes the basic selection probability of the mutation strategies, $N^c_{m,G}$ denotes the number of times the mth mutation strategy is used in the Gth generation, and ϵ denotes a very small positive number used to ensure that the denominator is not zero.

5.2.3. Adaptive parameter setting

The JADE algorithm (Zhang & Sanderson, 2009) provides an excellent adaptive mechanism for parameters F and CR, and many DE variants follow or modify this mechanism, such as the DEET (Li & Li, 2020), JADE_sort (Zhou et al., 2017) and SHADE (Tanabe & Fukunaga, 2013) algorithms. The parameter mechanism of the MHDE algorithm is also a modified version of JADE.

All successful parameter values in JADE produce the same effect on the Gaussian and Cauchy expectations. However, a parameter-adaptive mechanism was proposed by combining the JADE mechanism with a novel individual improvement, assuming that the parameter values that produce a greater contribution should provide more guidance for later parameter settings, as calculated in Eq. (16).

The crossover probabilities CR and scale factor F of success obtained using the mth mutation strategy are denoted as $SCR^c_{m,G} = \{scr^c_{1,m,G}, scr^c_{2,m,G}, \ldots\}$ and $SF^c_{m,G} = \{sf^c_{1,m,G}, sf^c_{2,m,G}, \ldots\}$, respectively.

The corresponding crossover probability CR^c_G is generated assuming that an individual is mutated using the mutation strategy, which is expressed in Eqs. (18).

$$\begin{cases} R^c_{i,m,G} = \max\left(\left(f(\vec{X}^c_{i,G}) - f(\vec{U}^c_{i,G})\right)\frac{f^G_{\max}-f(\vec{X}^c_{i,G})+\epsilon}{f^G_{\max}-f^G_{min}+\epsilon}, 0\right) \\ wmean^c_{m,G} = \frac{\sum_{i=1}^{|SCR^c_{m,G}|}\left(scr^c_{i,m,G}\cdot R^c_{i,m,G}\right)}{\sum_{i=1}^{|SCR^c_{m,G}|}\left(R^c_{i,m,G}\right)} \\ \mu CR^c_{m,G} = (1-w)\cdot \mu CR^c_{m,G-1} + w\cdot wmean^c_{m,G} \\ CR^c_G = randn(\mu CR^c_{m,G}, 0.1) \end{cases} \tag{18}$$

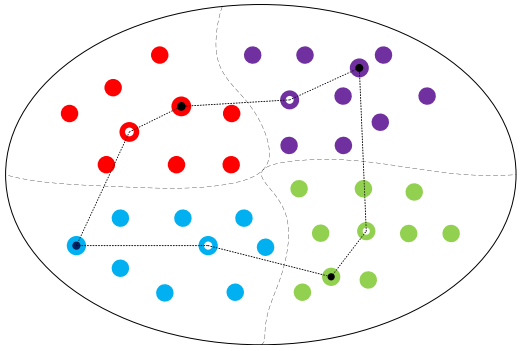


Fig. 3. Distribution of an extra-cluster subpopulation.

where $wmean^c_{m,G}$ denotes the weighted mean value obtained by calculating the successful crossover probability.

Each individual using the mth mutation strategy obtains scale factor F^c_G using Eq. (19).

$$\begin{cases} R^c_{i,m,G} = \max\left(\left(f(\vec{X}^c_{i,G}) - f(\vec{U}^c_{i,G})\right)\frac{f^G_{\max}-f(\vec{X}^c_{i,G})+\epsilon}{f^G_{\max}-f^G_{min}+\epsilon}, 0\right) \\ wmeanl^c_{m,G} = \frac{\sum_{i=1}^{|SF^c_{m,G}|}\left(\left(sf^c_{i,m,G}\right)^2\cdot R^c_{i,m,G}\right)}{\sum_{i=1}^{|SF^c_{m,G}|}\left(sf^c_{i,m,G}\cdot R^c_{i,m,G}\right)} \\ \mu F^c_{m,G} = (1-w)\cdot \mu F^c_{m,G-1} + w\cdot wmeanl^c_{m,G} \\ F^c_G = randc(\mu F^c_{m,G}, 0.1) \end{cases} \tag{19}$$

where $wmeanl^c_{m,G}$ denotes a weighted Lehmer mean.

While the $\mu CR^c_{m,G}$ and $\mu F^c_{m,G}$ of the mth strategy within the cth cluster evolve independently, MHDE simultaneously upgrades $\mu ACR_{m,G}$ and $\mu AF_{m,G}$ using information from the mth mutation strategy of all clusters. In addition, the $\mu ACR_{m,G}$ and $\mu AF_{m,G}$ are associated with the mth mutation strategy and are updated in the same way as that of $\mu CR^c_{m,G}$ and $\mu F^c_{m,G}$. At the beginning of each TS, $\mu CR^c_{m,G}$ and $\mu F^c_{m,G}$ were respectively set as:

$$\mu CR^c_{m,G} = \mu ACR_{m,G}, \quad m = 1, 2, \ldots, NC_G \tag{20}$$

$$\mu F^c_{m,G} = \mu AF_{m,G}, \quad m = 1, 2, \ldots, NC_G \tag{21}$$

5.3. Extra-cluster evolution

The MHDE algorithm constructs an extra-cluster subpopulation (EcPop) to enhance the overall convergence of the population and information interaction between clusters. The capacity of EcPop is expressed in terms of $NS = round(e{\times}NP)$. In addition, EcPop membership consists of two parts: the best individual in each cluster and individuals cyclically selected from different clusters until the EcPop is filled. The distributions of the clusters and EcPop are shown in Fig. 3, where the white and black centered dots represent the optimal and random individuals within the clusters, respectively.

The members of EcPop come from different clusters, thus EcPop carries information about all the clusters. EcPop can be considered as a refined version of the current population. Therefore, it can help the population to search the entire solution space while saving the

Algorithm 3: Pseudocode for the MHDE algorithm

Input: NP; $MaxFEs$; $\vec{X}^l$; $\vec{X}^u$; D; NC_{max}; NC_{min};
Output: The best solution
/* **Initialization** */
Produce the initial population by (7); Compute fitness values for the initial population; Initialize parameters $G = 0, FEs = 0$;
while $FEs < MaxFEs$ **do**
  /* **Clustering phase** */
  if *G%TS=0* **then**
    Cluster population by LSH; Compute the entropy value by (12);
    Compute NC_G by (15); Cluster population into NC_G clusters by BLSH;
    for $c = 1$ *to* NC_G **do**
      $p^c_{m,G} = Bp/4$; Set $\mu CR^c_{m,G}$ by (20); Set $\mu F^c_{m,G}$ by (21); $m = 1, 2, 3, 4$;
    end
  end
  /* **Intra-cluster evolution phase** */
  for $c = 1$ *to* NC_G **do**
    Compute selection probabilities by (17);
    Get the current cluster size cs_c;
    for $i = 1$ *to* cs_c **do**
      sm = Select a mutation strategy using roulette wheel;
      Compute $CR^c_{i,G}$ and $F^c_{i,G}$ by (18) and (19);
      Generate the $\vec{V}^c_{i,G}$;
      Generate the $\vec{U}^c_{i,G}$;
      Handle out-of-bounds elements ;
      Compute the $f(\vec{U}^c_{i,G})$;
      $FEs = FEs + 1$;
      if $f(\vec{U}^c_{i,G}) < f(\vec{X}^c_{i,G})$ **then**
        $P^{G+1} \leftarrow \vec{U}^c_{i,G}$; $A \leftarrow \vec{X}^c_{i,G}$;
        Compute the $R^c_{i,sm,G}$ and $I^c_{sm,G}$ by (16) and (17);
        $SCR^c_{sm,G} \leftarrow CR^c_{i,G}$; $SF^c_{sm,G} \leftarrow F^c_{i,G}$;
      else
        $P^{G+1} \leftarrow X^c_{i,G}$;
      end
    end
    Update $\mu CR^c_{m,G}$, $\mu F^c_{m,G}, m = 1, 2, 3, 4$;
  end
  Update $\mu ACR_{m,G}$, $\mu AF_{m,G}$ $m = 1, 2, 3, 4$;
  /* **Extra-cluster evolution phase** */
  Construct the EcPop;
  Compute selection probabilities $p^e_{m,G}$ for the mutation strategies of the EcPop;
  for $i = 1$ *to* NE_G **do**
    sem =Select a mutation strategy using roulette wheel;
    Generate $CR^e_{i,G}$ and $F^e_{i,G}$;
    Generate the $\vec{V}^e_{i,G}$;
    Generate the $\vec{U}^e_{i,G}$;
    Handle out-of-bounds elements ;
    Compute the $f(\vec{U}^e_{i,G})$;
    $FEs = FEs + 1$;
    if $f(\vec{U}^e_{i,G}) < f(\vec{X}^e_{i,G})$ **then**
      Compute the $R^e_{i,sem,G}$ and $I^e_{sem,G}$;
      $SCR^e_{sem,G} \leftarrow CR^e_{i,G}$; $SF^e_{sem,G} \leftarrow F^e_{i,G}$;
    end
    lc = Locate the cluster closest to the $\vec{U}^e_{i,G}$;
    Replace the first individual in the cluster lc worse than the $\vec{U}^e_{i,G}$ with the $\vec{U}^e_{i,G}$;
  end
  Upgrade $\mu CR^e_{m,G}$ and $\mu F^e_{m,G}$ $m = 1, 2$;
  $G = G + 1$;
end

number of function evaluations. Each element in EcPop is marked using the superscript e to distinguish between intra-cluster and extra-cluster elements, for example $CR^e_{i,G}$ and $F^e_{i,G}$.

MHDE uses two efficient mutation strategies to form a mutation pool and ensure a balance between exploration and exploitation in EcPop, which are listed below.

- DE/current-to-pbest/1

$$\vec{V}^e_{i,G} = \vec{X}^e_{i,G} + F^e_{i,G} \cdot \left(\vec{X}^e_{p,G} - \vec{X}^e_{i,G}\right) + F^e_{i,G} \cdot \left(\vec{X}^e_{r1,G} - \hat{\vec{X}}^e_{r3,G}\right)$$

- DE/rand-to-pbest/1

$$\vec{V}^e_{i,G} = \vec{X}^e_{r1,G} + F^e_{i,G} \cdot \left(\vec{X}^e_{p,G} - \vec{X}^e_{r1,G}\right) + F^e_{i,G} \cdot \left(\vec{X}^e_{r2,G} - \hat{\vec{X}}^e_{r3,G}\right)$$

where $\vec{X}^e_{p,G}$ is randomly selected from the top $p\%$ of individuals in EcPop, and $\vec{X}^e_{r1,G}$ and $\vec{X}^e_{r2,G}$ are mutually dissimilar random individuals

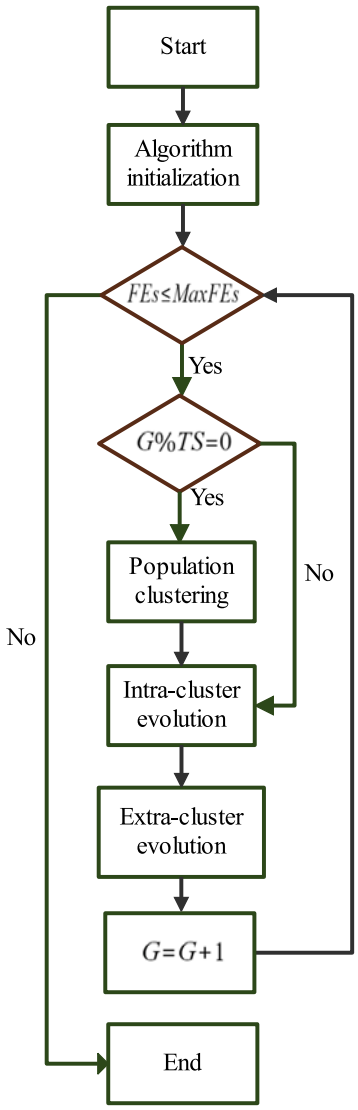


Fig. 4. Flowchart of the MHDE algorithm.

Table 2
Parameters settings.

Algorithm	Parameter settings
SDE	$F_i = F_{i4} + randn(0, 0.5) \cdot (F_{i5} - F_{i6})$; $CR_i = randn(0.5, 0.15)$
SaDE	$LP = 50$; $F_{i,G} = randn(0.5, 0.3)$
JADE	$p = 0.05$; $c = 0.1$
AGDE	$p = 0.1$
EDEV	$\lambda_1 = \lambda_2 = \lambda_3 = 0.1, ng = 20$
DEET	$p = 0.05$; $c_1 = 0.1$
DCDE	$m = 1$; $n = 3$
EHDE	$\lambda = 0.5$
MHDE	$w = 0.1$; $TS = 10$; $Bp = 0.2$; $NC_{max} = 20$; $NC_{min} = 5$; $e = 0.8$

in EcPop. $\hat{\vec{X}}^{e}_{r3,G}$ represents a random individual obtained from EcPop $\cup$ A. A represents an external archive population that is initially empty where inferior individuals are added as the population evolves. If A becomes full, the incoming individuals replace the random individuals within A. The capacity of A is equal to population size NP.

In addition, the parameter mechanism in EcPop is the same as that in intra-cluster evolution.

To minimize overlapping searches between clusters, each trial vector $\vec{U}^{e}_{i,G}$ in EcPop was added to the closest cluster to replace the first individual that was worse than the trial vector. The distance between the trial vector and clusters is calculated using Eq. (22).

$$Dis_{i,j} = \|\vec{U}^{e}_{i,G} - Mean(c_j)\|_2 \tag{22}$$

where $Mean(c_j)$ represents the mean of the individuals within cluster c_j.

Algorithm 3 presents the pseudocode of the MHDE algorithm to better illustrate the operational process and algorithm structure of the MHDE. The main operational structure of the MHDE algorithm is shown in Fig. 4.

6. Experimental results and analysis

The MHDE algorithm was experimentally compared with classical and efficient DE variants to test its optimization capability.

6.1. Experimental settings

The compared algorithms included the well-known DE variants SDE (Omran et al., 2005), JADE (Zhang & Sanderson, 2009) and SaDE (Qin et al., 2009), and recently proposed excellent DE variants AGDE (Mohamed & Mohamed, 2019), DEET (Li & Li, 2020), EDEV (Wu et al., 2018), DCDE (Deng et al., 2022) and EHDE (Zhong & Cheng, 2021). The CEC2017 (Awad et al., 2016) benchmark suite was employed for the comparative experiments.

The test functions from CEC2017 can be broadly divided into two categories: unimodal (F1-F3) and multimodal (F4-F30) functions. The multimodal functions can be further classified into three types: simple multimodal (F4-F10), hybrid (F11-F20), and composition (F21-F30) functions.

Each algorithm was run independently 30 times to ensure the fairness of the experiment, the maximum number of function evaluations was $MaxFEs = 10000 \times D$, and the population size NP was set to 100.

The mean and std were used as the performance evaluation criteria. The Wilcoxon test was employed as a non-parametric test to assess the comparative performance of the algorithms. Specifically, the Wilcoxon test was utilized to determine the statistical significance of the differences between the comparators and the MHDE algorithm at a significance level of 0.05. The test results were summarized using three assessments: "+", "≈", and "-", indicating that MHDE was significantly better than the comparator, there was no significant difference, and the comparator was significantly better than the MHDE, respectively. h represents the results of the Wilcoxon test. Additionally, the Friedman test was introduced to rank the algorithms, and the rankings are denoted as 'f-rank'.

The operating system of the experimental environment was Windows 10, the software was MATLAB 2020b, and the processor was an Intel(R) Core(TM) i5-9300H CPU @ 2.40 GHz 2.40 GHz with 8G RAM.

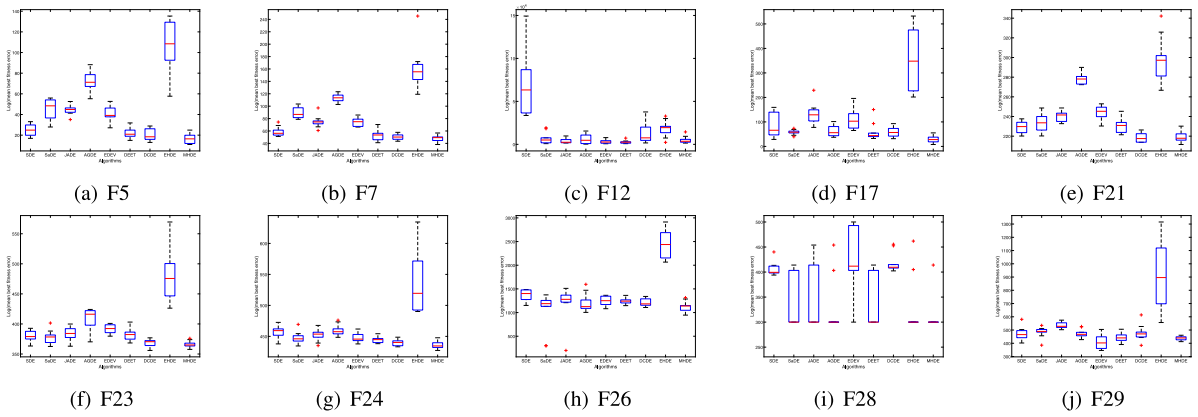
The parameter settings from our previous work were used for each of the participating algorithms. The parameter settings for the MHDE and comparison algorithms are presented in Table 2. Moreover, the vectors $\vec{O}$ used in LSH and BLSH were generated from Gaussian distributions. Gaussian distribution is one of the most well-known p-stable distributions, which has been shown to exhibit good performance in LSH techniques (Datar et al., 2004).

6.2. Analysis of experimental results

The test results of the algorithms on 30D CEC2017 are listed in Table 3. The MHDE algorithm demonstrated excellent performance for unimodal functions F1-F3. Specifically, the MHDE successfully located the global optimum in F1. Moreover, the mean and std indicated that the solutions obtained by the MHDE algorithm were satisfactory for F2 and F3. The MHDE algorithm preformed best for the seven simple multimodal functions on F5, F6, F7, F8, and F10. For the hybrid functions, the MHDE algorithm performed best for F17 and F18, and second-best for F11, F15, F16 and F19. Finally, for the composition

Table 3
The comparison results of MHDE with other algorithms on 30D CEC2017.

Functions	Item	SDE	SaDE	JADE	AGDE	EDEV	DEET	DCDE	EHDE	MHDE
F1	mean± std	4.41E+03 ± 4.79E+03	0.00E+00 ± 0.00E+00	3.79E−25 ± 5.51E−25	7.04E−23 ± 1.21E−22	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	3.59E−20 ± 1.13E−19	2.82E−01 ± 8.62E−01	0.00E+00 ± 0.00E+00
	f-rank h	9 +	3.15 ≈	4.95 +	5.45 +	3.15 ≈	3.15 ≈	5 +	8 +	3.15
F2	mean± std	5.39E+16 ± 1.69E+17	1.11E−10 ± 3.34E−10	8.36E−13 ± 1.43E−12	9.11E−06 ± 2.64E−05	3.34E−14 ± 5.16E−14	7.32E+10 ± 2.32E+11	3.68E+09 ± 7.07E+09	4.75E+00 ± 1.01E+01	1.79E−11 ± 2.81E−11
	f-rank h	8.9 +	3.6 ≈	2.6 −	5.7 +	1.5 −	3.8 ≈	8 +	6.9 +	4
F3	mean± std	4.64E+04 ± 1.08E+04	8.68E−20 ± 1.47E−19	1.50E+04 ± 2.04E+04	1.34E−22 ± 3.15E−22	2.29E−03 ± 7.19E−03	4.07E+03 ± 6.11E+03	1.36E+01 ± 1.27E+01	1.27E−03 ± 1.10E−03	3.99E−19 ± 9.48E−19
	f-rank h	9 +	3.4 ≈	6.1 +	1.8 −	5.1 +	3.6 ≈	7.2 +	6.1 +	2.7
F4	mean± std	8.71E+01 ± 2.07E−01	1.64E+01 ± 2.83E+01	1.41E+01 ± 2.35E+01	1.36E+01 ± 2.66E+01	9.20E−01 ± 1.64E+00	5.38E+01 ± 1.90E+01	8.64E+01 ± 2.77E−01	2.57E+01 ± 3.23E+01	2.63E+01 ± 3.23E+01
	f-rank h	9 +	3.7 ≈	3.8 ≈	3.9 ≈	2.3 ≈	5.5 ≈	8 +	4.9 ≈	3.9
F5	mean± std	2.46E+01 ± 5.69E+00	4.53E+01 ± 1.00E+01	4.47E+01 ± 5.02E+00	7.25E+01 ± 9.28E+00	4.09E+01 ± 7.10E+00	2.18E+01 ± 4.74E+00	2.01E+01 ± 5.85E+00	1.07E+02 ± 2.48E+01	1.69E+01 ± 5.05E+00
	f-rank h	3.2 +	6.1 +	6.1 +	8.1 +	5.6 +	2.6 ≈	2.6 ≈	8.9 +	1.8
F6	mean± std	2.77E−09 ± 6.69E−09	1.19E−02 ± 3.77E−02	1.48E−05 ± 5.36E−06	0.00E+00 ± 0.00E+00	1.44E−10 ± 2.78E−10	0.00E+00 ± 0.00E+00	2.94E−07 ± 4.69E−07	5.98E+00 ± 4.81E+00	0.00E+00 ± 0.00E+00
	f-rank h	4.05 ≈	3.9 ≈	7.9 +	2.95 ≈	5.9 +	2.95 ≈	5.4 +	9 +	2.95
F7	mean± std	5.90E+01 ± 7.51E+00	8.91E+01 ± 8.97E+00	7.51E+01 ± 9.44E+00	1.14E+02 ± 6.47E+00	7.46E+01 ± 6.30E+00	5.31E+01 ± 8.51E+00	5.04E+01 ± 4.45E+00	1.61E+02 ± 3.32E+01	4.76E+01 ± 6.15E+00
	f-rank h	3.6 +	6.7 +	5.4 +	8 +	5.7 +	2.7 ≈	2.1 ≈	9 +	1.8
F8	mean± std	2.86E+01 ± 8.95E+00	4.38E+01 ± 1.03E+01	4.23E+01 ± 4.89E+00	7.59E+01 ± 8.92E+00	4.31E+01 ± 4.49E+00	2.41E+01 ± 4.67E+00	1.72E+01 ± 4.60E+00	9.34E+01 ± 1.41E+01	1.75E+01 ± 4.12E+00
	f-rank h	3.6 +	6 +	5.9 +	8.2 +	5.7 +	3.2 +	1.8 ≈	8.8 +	1.8
F9	mean± std	6.33E−02 ± 1.48E−01	1.34E+00 ± 1.16E+00	8.95E−03 ± 2.83E−02	1.09E−01 ± 2.09E−01	5.44E−02 ± 1.72E−01	7.01E−29 ± 2.09E−31	7.40E−29 ± 3.61E−30	7.41E+02 ± 4.82E+02	4.54E−02 ± 1.44E−01
	f-rank h	4.15 ≈	8 +	3.15 ≈	5.95 +	3.55 ≈	2.85 ≈	5.45 −	9 +	2.9
F10	mean± std	5.26E+03 ± 9.43E+02	4.15E+03 ± 2.23E+02	3.35E+03 ± 2.42E+02	4.12E+03 ± 2.62E+02	3.58E+03 ± 6.66E+02	1.95E+03 ± 7.22E+02	1.91E+03 ± 4.78E+02	3.40E+03 ± 4.47E+02	1.77E+03 ± 6.31E+02
	f-rank h	8.6 +	7.4 +	5.3 +	7.1 +	5.8 +	2.1 ≈	2.2 ≈	4.8 +	1.7
F11	mean± std	5.01E+01 ± 2.84E+01	5.31E+01 ± 2.88E+01	4.07E+01 ± 2.71E+01	1.91E+01 ± 1.83E+01	2.57E+01 ± 1.64E+01	2.08E+01 ± 1.74E+01	5.89E+01 ± 2.66E+01	1.19E+02 ± 4.68E+01	2.79E+01 ± 2.40E+01
	f-rank h	5.2 ≈	6 +	5.1 ≈	2.9 ≈	4.1 ≈	3.4 ≈	6.2 +	8.7 +	3.4
F12	mean± std	6.83E+04 ± 3.68E+04	6.96E+03 ± 6.63E+03	3.62E+03 ± 2.67E+03	5.76E+03 ± 5.34E+03	3.18E+03 ± 2.16E+03	2.68E+03 ± 1.87E+03	1.31E+04 ± 1.29E+04	1.83E+04 ± 9.19E+03	5.12E+03 ± 4.03E+03
	f-rank h	9 +	4.1 ≈	3.8 ≈	4.1 ≈	3.5 ≈	2.8 ≈	6.2 ≈	6.8 +	4.7
F13	mean± std	8.89E+03 ± 6.19E+03	3.97E+01 ± 2.05E+01	1.59E+03 ± 4.87E+03	6.61E+01 ± 7.72E+01	8.49E+01 ± 9.19E+01	2.03E+01 ± 1.20E+01	4.96E+03 ± 6.10E+03	1.24E+04 ± 1.71E+04	3.63E+01 ± 1.61E+01
	f-rank h	7.8 +	3 ≈	4.5 ≈	4 ≈	4.9 ≈	1.4 −	7.4 +	8.2 +	3.8
F14	mean± std	4.06E+01 ± 1.01E+01	2.86E+01 ± 1.54E+01	8.41E+03 ± 9.94E+03	1.18E+01 ± 6.35E+00	4.10E+01 ± 4.89E+00	3.59E+03 ± 4.32E+03	3.30E+01 ± 5.68E+00	1.07E+02 ± 5.52E+01	3.23E+01 ± 7.75E+00
	f-rank h	5.3 +	3.3 ≈	7 +	1.3 −	5.8 +	6.9 +	3.9 ≈	7.8 +	3.7
F15	mean± std	3.05E+02 ± 3.75E+02	5.13E+01 ± 4.82E+01	3.69E+03 ± 4.98E+03	1.33E+01 ± 8.57E+00	1.98E+01 ± 7.99E+00	9.10E+02 ± 1.85E+03	3.37E+03 ± 7.80E+03	8.81E+02 ± 1.70E+03	1.19E+01 ± 6.46E+00
	f-rank h	6.7 +	5.5 +	5.4 ≈	2.3 ≈	3.6 ≈	4.4 ≈	7 +	7.6 +	2.5
F16	mean± std	2.92E+02 ± 1.34E+02	3.76E+02 ± 1.17E+02	6.11E+02 ± 1.18E+02	3.62E+02 ± 1.19E+02	5.79E+02 ± 1.29E+02	2.61E+02 ± 1.19E+02	1.28E+02 ± 1.34E+02	9.13E+02 ± 3.25E+02	2.04E+02 ± 1.62E+02
	f-rank h	3.9 ≈	4.6 +	7.5 +	4.5 +	7.5 +	3.2 ≈	2.3 ≈	8.7 +	2.8
F17	mean± std	8.37E+01 ± 4.99E+01	5.79E+01 ± 1.01E+01	1.33E+02 ± 4.14E+01	6.14E+01 ± 2.31E+01	1.13E+02 ± 4.38E+01	5.91E+01 ± 3.63E+01	5.91E+01 ± 1.88E+01	3.53E+02 ± 1.21E+02	2.83E+01 ± 1.47E+01
	f-rank h	4.9 +	4 +	7.2 +	4.1 +	6.5 +	3.6 +	4.1 +	9 +	1.6
F18	mean± std	6.28E+04 ± 5.67E+04	3.98E+01 ± 1.40E+01	5.74E+04 ± 1.00E+05	1.82E+02 ± 2.02E+02	3.47E+01 ± 4.46E+00	1.13E+04 ± 2.62E+04	5.98E+03 ± 5.67E+03	4.98E+03 ± 3.90E+03	2.74E+01 ± 1.88E+00
	f-rank h	8.4 +	3.5 +	5.3 +	5.3 +	3.1 +	3.8 ≈	7.2 +	6.9 +	1.5
F19	mean± std	5.96E+02 ± 1.65E+03	2.58E+01 ± 1.67E+01	2.68E+03 ± 5.92E+03	6.73E+00 ± 4.08E+00	1.64E+01 ± 2.66E+00	1.24E+03 ± 3.81E+03	3.54E+02 ± 7.79E+02	8.51E+02 ± 1.95E+03	7.71E+00 ± 2.90E+00
	f-rank h	7.3 +	5.5 +	4.8 +	1.7 ≈	5.6 +	5.2 +	4.7 +	8.1 +	2.1
F20	mean± std	8.34E+01 ± 7.10E+01	7.43E+01 ± 4.75E+01	1.63E+02 ± 6.88E+01	1.01E+02 ± 4.44E+01	1.16E+02 ± 6.00E+01	8.50E+01 ± 6.52E+01	6.08E+01 ± 5.02E+01	3.17E+02 ± 1.43E+02	1.15E+02 ± 1.11E+02
	f-rank h	3.5 ≈	4 ≈	6.9 ≈	5.4 ≈	5.4 ≈	3.9 ≈	2.9 ≈	8.8 +	4.2
F21	mean± std	2.29E+02 ± 6.40E+00	2.33E+02 ± 9.29E+00	2.40E+02 ± 5.60E+00	2.78E+02 ± 5.30E+00	2.44E+02 ± 7.50E+00	2.31E+02 ± 7.46E+00	2.19E+02 ± 4.86E+00	2.98E+02 ± 2.26E+01	2.19E+02 ± 5.51E+00
	f-rank h	3.7 +	4.7 +	5.8 +	8.2 +	6.5 +	4 +	1.5 ≈	8.8 +	1.8
F22	mean± std	7.22E+02 ± 1.97E+03	1.00E+02 ± 0.00E+00	1.00E+02 ± 8.20E−15	1.00E+02 ± 1.34E−14	1.00E+02 ± 0.00E+00	2.76E+02 ± 5.56E+02	2.12E+03 ± 1.23E+03	1.01E+02 ± 1.77E+00	1.00E+02 ± 0.00E+00
	f-rank h	4.35 ≈	3.75 ≈	4.7 ≈	4.5 ≈	3.75 ≈	4.3 ≈	7.9 +	8 +	3.75
F23	mean± std	3.80E+02 ± 9.02E+00	3.78E+02 ± 1.12E+01	3.85E+02 ± 1.11E+01	4.07E+02 ± 2.03E+01	3.92E+02 ± 7.57E+00	3.82E+02 ± 9.49E+00	3.69E+02 ± 6.25E+00	4.81E+02 ± 4.57E+01	3.66E+02 ± 5.40E+00
	f-rank h	4.1 +	4 +	5.3 +	6.8 +	6.7 +	5 +	2.5 ≈	9 +	1.6
F24	mean± std	4.58E+02 ± 1.01E+01	4.48E+02 ± 9.36E+00	4.52E+02 ± 9.58E+00	4.60E+02 ± 8.81E+00	4.48E+02 ± 7.50E+00	4.44E+02 ± 4.84E+00	4.40E+02 ± 5.03E+00	5.38E+02 ± 5.29E+01	4.35E+02 ± 6.18E+00
	f-rank h	6.5 +	4.3 +	5.7 +	7.1 +	4.7 +	3.5 +	2.7 ≈	9 +	1.5
F25	mean± std	3.88E+02 ± 7.67E−01	3.89E+02 ± 7.76E+00	3.87E+02 ± 1.98E−01	3.87E+02 ± 1.16E+00	3.79E+02 ± 1.38E+00	3.87E+02 ± 1.48E−01	3.87E+02 ± 3.65E−01	3.92E+02 ± 8.42E+00	3.87E+02 ± 3.30E−01
	f-rank h	8 +	5.6 ≈	3.9 ≈	4.7 ≈	1 −	3.7 ≈	5.7 ≈	7.9 +	4.5
F26	mean± std	1.36E+03 ± 1.30E+02	1.04E+03 ± 3.98E+02	1.20E+03 ± 3.65E+02	1.20E+03 ± 1.92E+02	1.25E+03 ± 9.97E+01	1.24E+03 ± 6.19E+01	1.21E+03 ± 8.43E+01	2.44E+03 ± 2.93E+02	1.13E+03 ± 1.12E+02
	f-rank h	6.2 +	4.2 ≈	5.9 +	3.4 ≈	5 +	5 +	4.1 +	9 +	2.2
F27	mean± std	5.06E+02 ± 3.33E+00	5.08E+02 ± 7.89E+00	5.09E+02 ± 2.84E+00	5.03E+02 ± 6.91E+00	5.00E+02 ± 1.97E−04	5.03E+02 ± 6.78E+00	5.04E+02 ± 5.78E+00	5.37E+02 ± 1.17E+01	5.02E+02 ± 4.24E+00
	f-rank h	5.6 ≈	5.5 ≈	6.3 +	4.2 ≈	2.3 −	3.6 ≈	4.7 ≈	9 +	3.8
F28	mean± std	4.06E+02 ± 1.36E+01	3.42E+02 ± 5.48E+01	3.49E+02 ± 6.39E+01	3.26E+02 ± 5.55E+01	4.14E+02 ± 7.17E+01	3.33E+02 ± 5.34E+01	4.18E+02 ± 1.94E+01	3.27E+02 ± 5.78E+01	3.11E+02 ± 3.60E+01
	f-rank h	6.4 +	3.4 ≈	4.65 ≈	3.95 ≈	6.75 +	3.55 ≈	7.5 +	5.4 +	3.4
F29	mean± std	4.74E+02 ± 4.90E+01	4.84E+02 ± 4.02E+01	5.35E+02 ± 2.24E+01	4.74E+02 ± 3.12E+01	4.08E+02 ± 5.25E+01	4.43E+02 ± 3.24E+01	4.79E+02 ± 6.00E+01	9.06E+02 ± 2.47E+02	4.38E+02 ± 1.52E+01
	f-rank h	5.2 +	5.7 +	7.5 +	4.9 +	2.1 ≈	3.2 ≈	5.1 +	8.8 +	2.5
F30	mean± std	4.43E+03 ± 1.67E+03	2.29E+03 ± 1.71E+02	3.84E+03 ± 5.08E+03	3.66E+03 ± 2.35E+03	2.78E+02 ± 8.60E+01	2.09E+03 ± 1.36E+02	3.95E+03 ± 1.82E+03	5.93E+03 ± 2.76E+03	2.07E+03 ± 6.09E+01
	f-rank h	7.6 +	4.6 +	5 +	6.1 +	1 −	2.7 ≈	7 +	8.1 +	2.9
	+/≈/−	23/7/0	16/14/0	19/10/1	15/13/2	16/10/4	8/21/1	16/13/1	29/1/0	
	average f-rank	6.09	4.71	5.45	4.88	4.47	3.65	4.95	7.97	2.83



(a) F5 (b) F7 (c) F12 (d) F17 (e) F21

(f) F23 (g) F24 (h) F26 (i) F28 (j) F29

Fig. 5. The boxplots of the comparison of MHDE with other algorithms.

functions, the MHDE algorithm ranked first on five of the ten functions, including F22, F23, F24, F26, and F28. The average f-rank showed that the proposed MHDE algorithm performed best at 2.83, and the +/≈/- showed that the MHDE algorithm significantly outperformed its competitors on 23, 16, 19, 15, 16, 8, 16, and 29 functions, respectively. In conclusion, the MHDE demonstrated excellent optimization performance on unimodal and multimodal problems in 30D CEC2017.

The test results of the algorithms for 50D CEC2017 are listed in Table 4. For unimodal functions, the MHDE and DEET algorithms achieved the same accuracy of 10^{-22} in F1. Additionally, MHDE performed third best in F2 after JADE and EDEV, and second best in F3 after AGDE. Overall, the performance of the MHDE algorithm for unimodal functions remained impressive. For simple multimodal functions, the MHDE algorithm achieved the second-best performance on

Table 4
The comparison results of MHDE with other algorithms on 50D CEC2017.

Functions	Item	SDE	SaDE	JADE	AGDE	EDEV	DEET	DCDE	EHDE	MHDE
F1	mean± std	1.63E+05 ± 3.35E+05	7.99E−11 ± 2.09E−10	2.61E−19 ± 4.60E−19	4.11E+03 ± 4.00E+03	4.86E−20 ± 1.02E−19	1.43E−22 ± 1.30E−22	2.66E−03 ± 3.19E−03	5.57E+03 ± 5.07E+03	1.93E−22 ± 1.12E−22
	f-rank h	8.4 +	5 +	3.8 +	7.7 +	2.5 ≈	1.7 ≈	6 +	7.9 +	2
F2	mean± std	4.06E+34 ± 1.28E+35	8.17E+01 ± 7.00E+01	1.46E−09 ± 3.07E−09	1.31E+01 ± 2.59E+01	1.64E−10 ± 1.97E−10	1.61E+16 ± 5.08E+16	5.39E+23 ± 1.70E+24	5.13E+04 ± 1.24E+05	5.91E−08 ± 1.55E−07
	f-rank h	9 +	5.6 +	2.2 ≈	5.1 +	2.1 ≈	3.3 ≈	8 +	6.9 +	2.8
F3	mean± std	1.39E+05 ± 2.06E+04	1.05E−05 ± 1.81E−05	1.87E+04 ± 4.06E+04	7.20E−10 ± 1.53E−09	2.97E+01 ± 5.27E+01	1.42E+04 ± 2.45E+04	5.20E+02 ± 3.36E+02	1.03E+00 ± 8.78E−01	2.25E−03 ± 7.11E−03
	f-rank h	9 +	4.3 −	5 +	1.7 −	5.7 +	3 ≈	7.4 +	5.9 +	3
F4	mean± std	1.38E+02 ± 3.70E+01	4.19E+01 ± 5.23E+01	3.82E+01 ± 5.00E+01	5.91E+01 ± 4.62E+01	2.66E+01 ± 3.02E+01	3.86E+01 ± 3.75E+01	1.41E+02 ± 3.62E+01	8.30E+01 ± 3.64E+01	4.28E+01 ± 5.07E+01
	f-rank h	7.6 +	3.75 ≈	3.7 ≈	5.2 ≈	3.4 ≈	3.35 ≈	8.2 +	5.9 ≈	3.9
F5	mean± std	6.10E+01 ± 2.04E+01	9.02E+01 ± 2.88E+01	8.63E+01 ± 9.19E+00	2.17E+02 ± 1.42E+01	6.69E+01 ± 1.66E+01	5.27E+01 ± 1.41E+01	3.30E+01 ± 6.68E+00	2.64E+02 ± 4.22E+01	4.35E+01 ± 8.10E+00
	f-rank h	4.1 +	6 +	6.2 +	8.2 +	4.7 +	3.3 ≈	1.4 −	8.8 +	2.3
F6	mean± std	9.72E−07 ± 2.70E−06	1.10E−01 ± 1.95E−01	3.82E−06 ± 4.51E−06	0.00E+00 ± 0.00E+00	1.25E−15 ± 2.54E−15	0.00E+00 ± 0.00E+00	5.93E−07 ± 7.28E−07	2.28E+01 ± 6.94E+00	2.24E−05 ± 3.23E−05
	f-rank h	5 −	5.75 ≈	5.8 −	2.05 −	2.75 −	2.05 −	5.3 −	9 +	7.3
F7	mean± std	1.00E+02 ± 6.34E+00	2.04E+02 ± 4.52E+01	1.38E+02 ± 1.17E+01	2.75E+02 ± 1.00E+01	1.32E+02 ± 1.68E+01	9.81E+01 ± 1.35E+01	8.28E+01 ± 1.04E+01	4.41E+02 ± 6.08E+01	8.56E+01 ± 5.31E+00
	f-rank h	3.4 +	6.8 +	5.7 +	8 +	5.4 +	3.2 ≈	1.6 ≈	9 +	1.9
F8	mean± std	6.07E+01 ± 1.75E+01	8.95E+01 ± 2.47E+01	9.18E+01 ± 1.17E+01	2.11E+02 ± 1.70E+01	7.60E+01 ± 1.55E+01	5.07E+01 ± 1.05E+01	3.28E+01 ± 7.06E+00	2.47E+02 ± 4.01E+01	4.49E+01 ± 8.21E+00
	f-rank h	3.7 +	5.7 +	6.6 +	8.1 +	5.1 +	3.2 ≈	1.1 −	8.9 +	2.6
F9	mean± std	7.51E−01 ± 5.69E−01	1.56E+01 ± 2.70E+01	4.98E−01 ± 4.59E−01	8.70E−01 ± 5.42E−01	1.61E+00 ± 1.26E+00	7.17E−01 ± 9.71E−01	3.11E−28 ± 7.05E−30	5.02E+03 ± 1.04E+03	6.47E+00 ± 4.97E+00
	f-rank h	4.2 −	7.2 ≈	3.25 −	4.8 −	5.35 −	3.1 −	1.4 −	9 +	6.7
F10	mean± std	8.91E+03 ± 3.24E+03	8.33E+03 ± 3.25E+02	5.73E+03 ± 2.94E+02	8.66E+03 ± 4.49E+02	6.32E+03 ± 7.76E+02	3.84E+03 ± 6.69E+02	4.61E+03 ± 7.31E+02	6.15E+03 ± 6.57E+02	4.03E+03 ± 6.34E+02
	f-rank h	7.2 +	7.7 +	4.7 +	8.1 +	5.6 +	1.7 ≈	2.8 +	5.4 +	1.8
F11	mean± std	4.37E+01 ± 7.21E+00	1.21E+02 ± 3.59E+01	1.68E+02 ± 6.47E+01	5.01E+01 ± 1.30E+01	6.47E+01 ± 2.19E+01	1.02E+02 ± 2.87E+01	5.22E+01 ± 1.71E+01	1.92E+02 ± 5.71E+01	8.46E+01 ± 3.60E+01
	f-rank h	2 −	7.1 +	7.5 +	2.5 −	3.7 ≈	6.1 ≈	2.6 −	8.4 +	5.1
F12	mean± std	1.16E+06 ± 7.00E+05	6.83E+04 ± 4.50E+04	1.42E+04 ± 1.40E+04	3.31E+04 ± 2.01E+04	8.78E+03 ± 7.29E+03	7.63E+03 ± 6.84E+03	1.10E+05 ± 7.13E+04	1.22E+05 ± 9.88E+04	7.20E+03 ± 5.64E+03
	f-rank h	9 +	6.1 +	3.3 ≈	4.7 +	2.4 ≈	2.5 ≈	7.3 +	7 +	2.7
F13	mean± std	4.34E+03 ± 6.86E+03	1.46E+03 ± 1.47E+03	2.84E+02 ± 3.05E+02	2.43E+03 ± 4.08E+03	1.27E+02 ± 2.90E+01	1.15E+03 ± 2.09E+03	8.34E+03 ± 1.00E+04	7.19E+03 ± 7.97E+03	2.41E+02 ± 4.43E+02
	f-rank h	6.5 +	5.6 +	2.9 ≈	6 +	2.1 ≈	4.2 +	7.9 +	7.7 +	2.1
F14	mean± std	7.99E+03 ± 7.18E+03	1.53E+02 ± 4.87E+01	1.40E+02 ± 4.87E+01	3.11E+02 ± 3.71E+02	6.10E+01 ± 1.72E+01	2.56E+04 ± 6.04E+04	2.82E+02 ± 4.39E+02	6.16E+02 ± 4.38E+02	1.20E+02 ± 4.75E+01
	f-rank h	8.8 +	4.4 ≈	4.4 ≈	4.6 ≈	1.3 −	6.3 +	3.8 ≈	7.5 +	3.9
F15	mean± std	6.93E+03 ± 5.14E+03	2.54E+02 ± 8.17E+01	1.58E+02 ± 6.74E+01	3.50E+03 ± 4.22E+03	2.68E+02 ± 3.18E+02	1.17E+02 ± 7.60E+01	1.45E+04 ± 9.88E+03	9.21E+03 ± 7.68E+03	5.90E+01 ± 1.79E+01
	f-rank h	7.6 +	4.3 +	3.1 +	6.9 +	4 +	2.7 ≈	7.7 +	7.4 +	1.3
F16	mean± std	8.38E+02 ± 2.86E+02	5.90E+02 ± 1.97E+02	1.07E+03 ± 1.08E+02	8.77E+02 ± 2.24E+02	7.05E+02 ± 3.02E+02	7.86E+02 ± 3.26E+02	7.28E+02 ± 2.69E+02	1.65E+03 ± 4.70E+02	7.65E+02 ± 2.77E+02
	f-rank h	4.4 ≈	2.9 ≈	7.1 +	5.5 ≈	3.8 ≈	4.3 ≈	4 ≈	8.6 +	4.4
F17	mean± std	4.09E+02 ± 2.17E+02	4.67E+02 ± 1.27E+02	7.49E+02 ± 1.47E+02	6.25E+02 ± 1.04E+02	6.45E+02 ± 1.91E+02	5.80E+02 ± 2.37E+02	2.18E+02 ± 1.58E+02	1.29E+03 ± 3.04E+02	5.20E+02 ± 2.79E+02
	f-rank h	3.6 ≈	3.9 ≈	7.1 +	5.4 ≈	5.5 ≈	4.9 ≈	1.6 −	8.8 +	4.2
F18	mean± std	5.13E+05 ± 2.89E+05	1.19E+03 ± 1.42E+03	1.97E+02 ± 9.93E+01	4.77E+03 ± 2.97E+03	1.64E+02 ± 1.46E+02	9.91E+01 ± 7.97E+01	1.96E+04 ± 8.96E+03	1.73E+04 ± 1.05E+04	9.61E+02 ± 1.07E+03
	f-rank h	9 +	4.4 ≈	3 ≈	6 +	2.4 −	1.6 −	7.6 +	7.3 +	3.7
F19	mean± std	1.37E+04 ± 7.94E+03	5.91E+01 ± 3.16E+01	5.43E+01 ± 1.79E+01	8.58E+02 ± 2.07E+03	8.16E+01 ± 4.57E+01	3.78E+02 ± 8.91E+02	2.65E+03 ± 4.21E+03	3.28E+03 ± 5.67E+03	4.23E+01 ± 2.30E+01
	f-rank h	8.7 +	3 ≈	3 +	5 ≈	4 +	5.2 +	7.3 +	6.7 +	2.1
F20	mean± std	4.02E+02 ± 1.65E+02	3.75E+02 ± 1.24E+02	6.78E+02 ± 1.94E+02	5.26E+02 ± 1.31E+02	3.07E+02 ± 1.35E+02	4.24E+02 ± 2.08E+02	2.62E+02 ± 1.52E+02	9.15E+02 ± 1.85E+02	4.44E+02 ± 2.46E+02
	f-rank h	4.1 ≈	3.9 ≈	7.3 +	6.2 ≈	3.4 ≈	4.5 ≈	2.4 ≈	8.7 +	4.5
F21	mean± std	2.58E+02 ± 1.26E+01	2.70E+02 ± 1.38E+01	2.84E+02 ± 1.21E+01	4.11E+02 ± 1.82E+01	2.64E+02 ± 2.68E+01	2.53E+02 ± 1.24E+01	2.34E+02 ± 7.36E+00	3.96E+02 ± 3.67E+01	2.38E+02 ± 8.25E+00
	f-rank h	4.1 +	5.3 +	6.5 +	8.7 +	4.8 +	3.9 +	1.5 ≈	8.3 +	1.9
F22	mean± std	6.59E+03 ± 2.02E+03	2.74E+03 ± 4.26E+03	4.04E+03 ± 3.24E+03	7.58E+03 ± 3.59E+03	6.92E+03 ± 2.97E+03	4.65E+03 ± 4.97E+02	4.45E+03 ± 1.17E+03	7.27E+03 ± 8.52E+02	4.18E+03 ± 6.66E+02
	f-rank h	6 +	3.1 ≈	4.3 ≈	7.05 +	6.65 +	4.1 ≈	3.5 ≈	7.3 +	3
F23	mean± std	4.90E+02 ± 1.21E+01	4.94E+02 ± 1.45E+01	5.05E+02 ± 1.27E+01	5.94E+02 ± 6.38E+01	4.94E+02 ± 1.38E+01	4.75E+02 ± 1.76E+01	4.59E+02 ± 1.82E+01	6.72E+02 ± 3.26E+01	4.62E+02 ± 1.54E+01
	f-rank h	4.7 +	5.2 +	6.1 +	7.4 +	5.3 +	3.3 ≈	1.8 ≈	9 +	2.2
F24	mean± std	5.68E+02 ± 1.31E+01	5.74E+02 ± 1.87E+01	5.66E+02 ± 1.26E+01	5.74E+02 ± 1.88E+01	5.50E+02 ± 1.54E+01	5.40E+02 ± 1.32E+01	5.40E+02 ± 1.24E+01	7.75E+02 ± 6.03E+01	5.28E+02 ± 5.42E+00
	f-rank h	6 +	6.5 +	5.8 +	6.6 +	4.3 +	2.7 +	2.6 +	9 +	1.5
F25	mean± std	5.32E+02 ± 3.89E+01	5.54E+02 ± 3.90E+01	5.38E+02 ± 2.92E+01	5.29E+02 ± 3.98E+01	4.60E+02 ± 2.03E+01	5.35E+02 ± 3.44E+01	4.80E+02 ± 2.84E−02	5.18E+02 ± 4.11E+01	5.57E+02 ± 2.49E+01
	f-rank h	5.6 ≈	6.8 ≈	5.7 ≈	5.3 ≈	1.2 −	5.6 ≈	3 −	4.9 ≈	6.9
F26	mean± std	1.96E+03 ± 1.02E+02	2.25E+03 ± 2.56E+02	1.90E+03 ± 1.60E+02	1.81E+03 ± 4.41E+02	1.76E+03 ± 1.93E+02	1.59E+03 ± 1.31E+02	1.59E+03 ± 1.35E+02	4.26E+03 ± 2.49E+03	1.53E+03 ± 9.33E+01
	f-rank h	6.4 +	8 +	5.9 +	5 +	4.6 +	2.6 ≈	2.9 ≈	7.4 +	2.2
F27	mean± std	6.21E+02 ± 3.47E+01	6.18E+02 ± 3.87E+01	5.97E+02 ± 1.56E+01	5.73E+02 ± 3.32E+01	5.00E+02 ± 2.21E−04	5.62E+02 ± 4.01E+01	5.46E+02 ± 1.67E+01	8.91E+02 ± 9.12E+01	5.64E+02 ± 3.81E+01
	f-rank h	6.8 +	6.8 +	5.8 +	4.6 ≈	1 −	3.7 ≈	3 ≈	9 +	4.3
F28	mean± std	5.08E+02 ± 1.46E+01	4.89E+02 ± 1.92E+01	4.97E+02 ± 1.58E+01	4.97E+02 ± 2.05E+01	4.47E+02 ± 9.76E+00	4.83E+02 ± 2.28E+01	4.88E+02 ± 2.52E+01	4.92E+02 ± 2.00E+01	4.95E+02 ± 1.26E+01
	f-rank h	8.2 +	4.3 ≈	5.1 ≈	6.2 ≈	1.3 −	3.7 ≈	5.6 ≈	5.7 ≈	4.9
F29	mean± std	4.63E+02 ± 1.10E+02	4.93E+02 ± 1.20E+02	6.12E+02 ± 6.23E+01	4.82E+02 ± 1.16E+02	4.51E+02 ± 1.04E+02	3.80E+02 ± 4.06E+01	4.56E+02 ± 6.48E+01	1.60E+03 ± 5.27E+02	4.59E+02 ± 1.34E+02
	f-rank h	4.5 ≈	4.7 ≈	7.4 +	4.7 ≈	4 ≈	2.4 ≈	4.4 ≈	9 +	3.9
F30	mean± std	8.79E+05 ± 1.70E+05	6.80E+05 ± 6.58E+04	7.08E+05 ± 1.35E+05	7.62E+05 ± 7.79E+04	5.73E+02 ± 1.99E+02	6.63E+05 ± 1.06E+05	6.85E+05 ± 6.92E+04	7.91E+05 ± 1.81E+05	5.95E+05 ± 1.83E+04
	f-rank h	7.3 +	5 +	5.6 +	6.9 +	1 −	4.2 ≈	5.4 +	6.9 +	2.7
	+/≈/−	22/5/3	16/13/1	19/9/2	16/10/4	12/10/8	5/22/3	12/11/7	27/3/0	
	average f-rank	6.16	5.3	5.13	5.81	3.65	3.55	4.3	7.71	3.39

F5, F7, F8, and F10 after only the DCDE or DEET algorithms. Finally, for complex hybrid and composition functions, MHDE performed best for F13, F15, F19, F22, F24, and F26, and was closely matched by the top performers on F12, F21, F23, and F29. From the final statistical results, it can be seen that the MHDE algorithm performed the best overall in the Friedman comprehensive test with an average f-rank of 3.39. Additionally, the Wilcoxon test showed that the MHDE algorithm outperformed the compared algorithms on 22, 16, 19, 16, 12, 5, 12, and 27 test functions.

In addition, comparative experiments were conducted on 100D CEC2017 to explore the performance of MHDE on high-dimensional problems, and the results are presented in Table 5.

The performance of MHDE on 100D problems was not as good as those on 30D and 50D problems. However, it still achieved an average f-rank score of 3.65, second only to the DEET algorithm. Moreover, the Wilcoxon test showed that the MHDE algorithm significantly outperformed the compared algorithms on 23, 16, 19, 17, 18, 7, 3, 8, and 27 test functions.

6.3. Algorithm stability and convergence analysis

Stability and convergence ability are important indicators for evaluating algorithm performance. Therefore, various characteristics were comprehensively analyzed during the execution of the MHDE algorithm to explore its stability and convergence ability.

6.3.1. Scalability analysis of MHDE

The statistical boxplots of the algorithms for ten test functions are shown in Fig. 5. The plots show that the MHDE algorithm exhibited a relatively stable optimization performance.

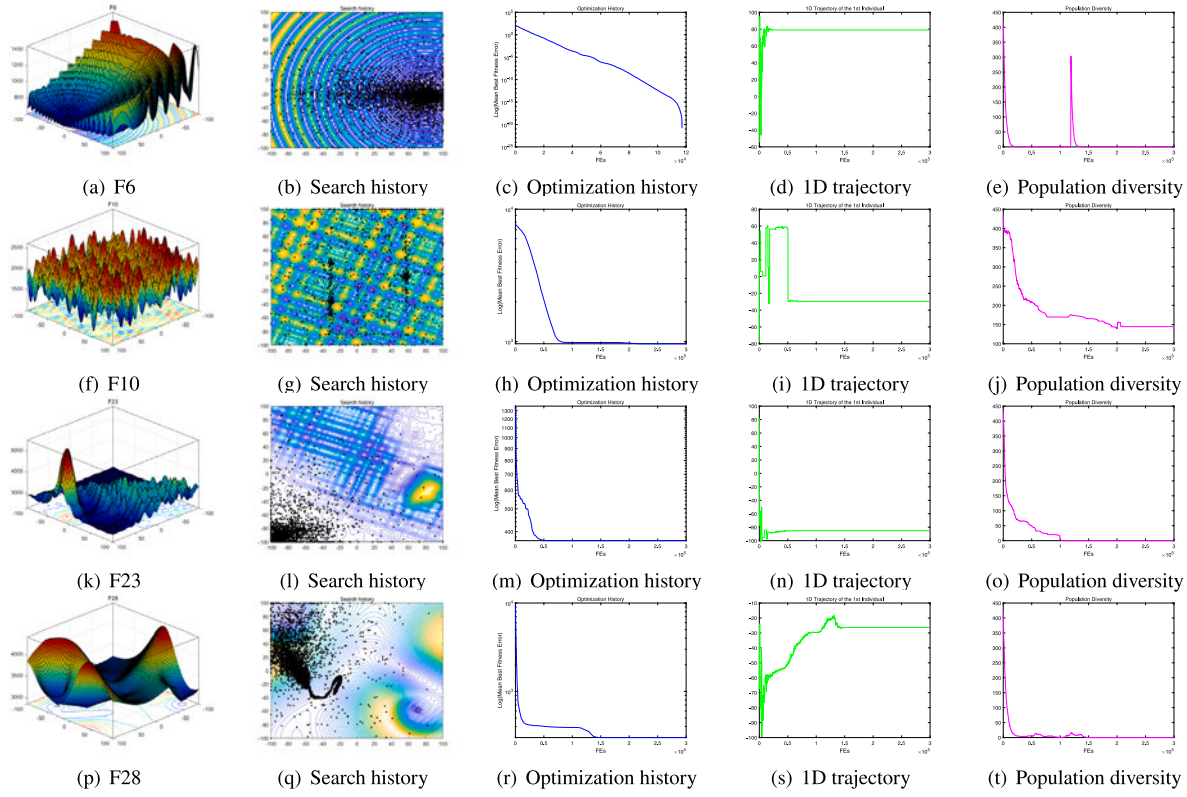
6.3.2. Comparative convergence analysis

Four qualitative metrics were used to evaluate the convergence behavior of the algorithms, as shown in Fig. 6. These metrics were the search history of the population, optimization history curve of the optimal solution of the population, 1D trajectory of an individual, and diversity curve of the population.

The population of the MHDE algorithm demonstrated rapid convergence towards promising regions in all four tested functions. The 1D trajectories of the individuals showed a trend of transitioning from significant fluctuations to gradual stability. In addition, the optimization history and population diversity curves showed continuous downward trends, which further demonstrated the excellent global convergence ability of the MHDE algorithm.

The average convergence curves of all tested algorithms on ten functions were analyzed to compare the convergence ability of the MHDE algorithm with that of the other algorithms, as shown in Fig. 7.

First, the MHDE algorithm initially converged slower than some algorithms, but eventually successfully found the global optimum, as shown in Fig. 7(a). Second, the MHDE exhibited a slower initial convergence speed than those of the SaDE and AGDE algorithms, but a



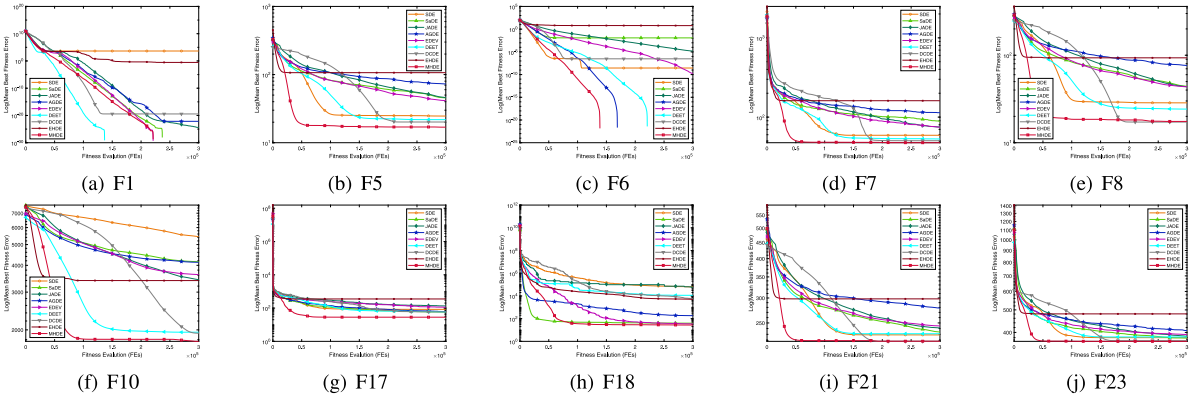
(a) F6 (b) Search history (c) Optimization history (d) 1D trajectory (e) Population diversity

(f) F10 (g) Search history (h) Optimization history (i) 1D trajectory (j) Population diversity

(k) F23 (l) Search history (m) Optimization history (n) 1D trajectory (o) Population diversity

(p) F28 (q) Search history (r) Optimization history (s) 1D trajectory (t) Population diversity

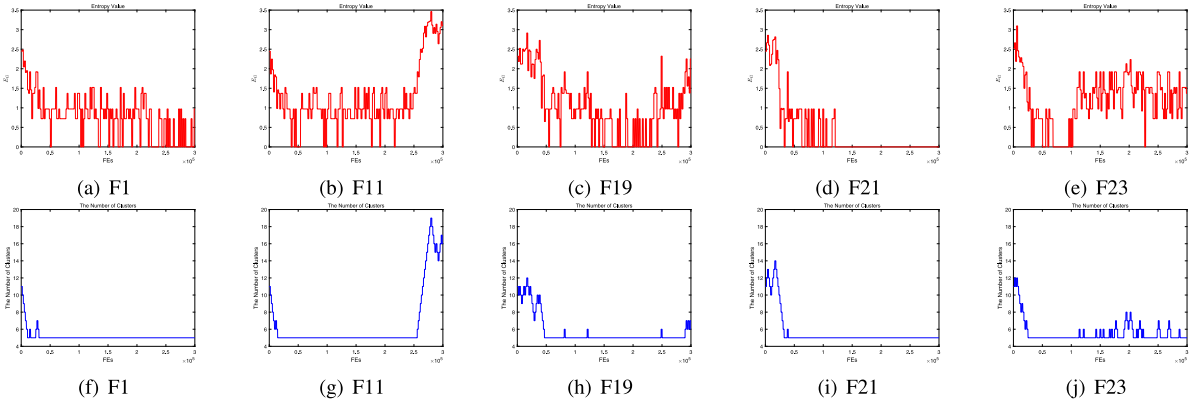
Fig. 6. Qualitative metrics:search history, optimization history, trajectory in 1st dimension and population diversity.



(a) F1 (b) F5 (c) F6 (d) F7 (e) F8

(f) F10 (g) F17 (h) F18 (i) F21 (j) F23

Fig. 7. The convergence curves of the comparison of MHDE with other algorithms.



(a) F1 (b) F11 (c) F19 (d) F21 (e) F23

(f) F1 (g) F11 (h) F19 (i) F21 (j) F23

Fig. 8. The entropy value and clustering quantity.

Table 5
The comparison results of MHDE with other algorithms on 100D CEC2017.

Functions	Item	SDE	SaDE	JADE	AGDE	EDEV	DEET	DCDE	EHDE	MHDE
F1	mean±std	1.40E+07 ± 1.97E+07	7.82E−02 ± 1.57E−01	7.93E−11 ± 2.24E−10	8.28E+03 ± 8.90E+03	3.57E−07 ± 9.80E−07	8.57E−17 ± 1.29E−16	4.69E−03 ± 5.74E−03	8.81E+03 ± 1.79E+04	7.98E−15 ± 1.20E−14
	f-rank h	9 +	5.9 +	2.9 +	7.6 +	4 +	1.1 -	5.1 +	7.4 +	2
F2	mean±std	4.12E+92 ± 1.13E+93	1.29E+24 ± 4.08E+24	1.51E+55 ± 4.79E+55	1.08E+11 ± 3.42E+11	1.04E+08 ± 2.85E+08	3.82E+20 ± 1.20E+21	1.52E+38 ± 3.18E+38	7.20E+34 ± 1.55E+35	9.61E+07 ± 3.00E+08
	f-rank h	9 +	5.6 +	3.8 ≈	4.3 +	2.7 ≈	3.1 +	7.6 +	7 +	1.9
F3	mean±std	5.94E+05 ± 5.36E+04	2.41E+03 ± 1.04E+03	1.19E+05 ± 1.54E+05	3.00E+01 ± 4.46E+01	9.78E+02 ± 1.12E+03	1.41E+05 ± 1.84E+05	7.31E+04 ± 1.97E+04	1.70E+03 ± 8.77E+02	7.46E+02 ± 6.89E+02
	f-rank h	8.1 +	5.2 +	5.4 +	1.7 ≈	3.5 ≈	6.2 +	7.2 +	5.2 +	2.5
F4	mean±std	2.54E+02 ± 2.48E+01	1.97E+02 ± 5.24E+01	8.12E+01 ± 5.36E+01	1.87E+02 ± 5.36E+01	9.19E+01 ± 7.68E+01	7.93E+01 ± 5.97E+01	2.17E+02 ± 7.23E+00	2.19E+02 ± 3.70E+01	1.38E+02 ± 4.11E+01
	f-rank h	8.4 +	6 +	2.3 -	5.3 ≈	2.8 ≈	2.2 -	6.8 +	7.2 +	4
F5	mean±std	1.54E+02 ± 2.33E+01	3.18E+02 ± 4.19E+01	2.33E+02 ± 2.04E+01	6.67E+02 ± 3.02E+01	1.47E+02 ± 3.36E+01	1.41E+02 ± 2.44E+01	6.65E+01 ± 1.77E+01	6.63E+02 ± 5.47E+01	1.51E+02 ± 1.73E+01
	f-rank h	3.6 ≈	6.9 +	6.1 +	8.4 +	3.6 ≈	3.3 ≈	1 -	8.6 +	3.5
F6	mean±std	3.00E−03 ± 3.22E−03	2.49E−02 ± 4.85E−02	6.04E−04 ± 1.91E−03	1.17E−09 ± 3.71E−09	6.04E−04 ± 1.91E−03	3.03E−03 ± 2.97E−03	6.19E−06 ± 5.06E−06	4.09E+01 ± 5.26E+00	2.01E−01 ± 1.82E−01
	f-rank h	5.8 -	5.55 -	3.4 -	1.8 -	2.05 -	5.1 -	4.4 -	9 +	7.9
F7	mean±std	2.54E+02 ± 2.36E+01	4.98E+02 ± 5.75E+01	3.66E+02 ± 3.42E+01	8.35E+02 ± 3.56E+01	2.92E+02 ± 1.89E+01	2.50E+02 ± 2.35E+01	1.64E+02 ± 8.41E+00	1.43E+03 ± 2.07E+02	3.15E+02 ± 4.17E+01
	f-rank h	2.8 -	6.6 +	5.6 +	8 +	4.3 ≈	2.6 -	1.6 -	9 +	4.5
F8	mean±std	1.58E+02 ± 2.00E+01	2.87E+02 ± 3.04E+01	2.33E+02 ± 2.66E+01	6.22E+02 ± 1.52E+02	1.81E+02 ± 2.60E+01	1.36E+02 ± 1.12E+01	6.58E+01 ± 1.32E+01	6.40E+02 ± 7.24E+01	1.52E+02 ± 3.05E+01
	f-rank h	3.7 ≈	7.1 +	6 +	8.5 +	4.5 ≈	2.7 ≈	1 -	8.3 +	3.2
F9	mean±std	2.61E+01 ± 3.92E+01	3.83E+03 ± 2.62E+03	1.97E+02 ± 1.46E+02	1.18E+02 ± 1.41E+02	1.36E+02 ± 1.23E+02	6.62E+01 ± 4.86E+01	5.44E−02 ± 1.43E−01	1.71E+04 ± 1.57E+03	4.39E+02 ± 2.26E+02
	f-rank h	2.5 -	7.5 +	5.4 -	4.4 -	4.7 -	3.6 -	1 -	9 +	6.9
F10	mean±std	1.34E+04 ± 1.17E+03	2.13E+04 ± 7.23E+02	1.52E+04 ± 7.87E+02	2.47E+04 ± 5.18E+02	1.30E+04 ± 1.70E+03	1.03E+04 ± 1.13E+03	1.34E+04 ± 3.87E+03	1.43E+04 ± 1.09E+03	1.15E+04 ± 1.50E+03
	f-rank h	4.5 +	7.9 +	6.3 +	9 +	4 +	1.6 ≈	3.4 ≈	5.6 +	2.7
F11	mean±std	6.07E+02 ± 3.76E+02	3.54E+02 ± 9.77E+01	3.57E+03 ± 3.84E+03	1.95E+02 ± 5.11E+01	6.39E+02 ± 3.16E+02	6.07E+03 ± 1.24E+03	7.95E+02 ± 1.67E+02	1.08E+03 ± 1.75E+02	6.30E+02 ± 2.83E+02
	f-rank h	4.5 ≈	2.6 -	6.4 ≈	1.5 -	4.3 ≈	7.4 ≈	5.9 ≈	7.5 +	4.9
F12	mean±std	2.94E+06 ± 1.42E+06	2.81E+05 ± 1.18E+05	6.37E+04 ± 5.34E+04	2.97E+05 ± 9.66E+04	5.98E+04 ± 1.81E+04	7.71E+04 ± 7.51E+03	3.53E+05 ± 1.70E+05	9.67E+05 ± 9.40E+05	6.33E+04 ± 3.37E+04
	f-rank h	8.6 +	5.8 +	2.9 ≈	6 +	2.1 ≈	3.1 ≈	6.3 +	7.7 +	2.5
F13	mean±std	3.87E+03 ± 3.75E+03	3.38E+03 ± 3.38E+03	3.40E+03 ± 3.10E+03	1.35E+03 ± 9.52E+02	3.29E+03 ± 2.63E+03	2.02E+03 ± 1.76E+03	3.36E+03 ± 2.94E+03	9.24E+03 ± 3.43E+03	3.53E+03 ± 3.52E+03
	f-rank h	5.1 ≈	4.7 ≈	4.7 ≈	3.2 ≈	5.5 ≈	3.9 ≈	5 ≈	8.5 +	4.4
F14	mean±std	4.65E+05 ± 2.10E+05	5.70E+03 ± 8.34E+03	5.68E+05 ± 1.79E+06	1.58E+04 ± 6.86E+03	3.86E+03 ± 9.78E+03	3.57E+02 ± 1.19E+02	2.16E+04 ± 1.39E+04	1.47E+04 ± 1.00E+04	6.94E+02 ± 4.07E+02
	f-rank h	8.9 +	5 +	3.3 +	6.8 +	3.2 +	1.6 -	7.1 +	6.3 +	2.8
F15	mean±std	3.16E+03 ± 4.85E+03	1.21E+03 ± 1.78E+03	6.54E+02 ± 7.45E+02	2.27E+03 ± 2.54E+03	6.54E+02 ± 2.89E+02	3.97E+02 ± 2.83E+02	1.83E+03 ± 2.60E+03	2.26E+03 ± 2.22E+03	9.57E+02 ± 9.67E+02
	f-rank h	6.3 ≈	4.2 ≈	3.9 ≈	5.2 ≈	5 ≈	3.3 ≈	5.8 ≈	6.7 ≈	4.6
F16	mean±std	2.95E+03 ± 3.26E+02	2.59E+03 ± 3.97E+02	3.06E+03 ± 3.53E+02	2.37E+03 ± 6.21E+02	2.64E+03 ± 4.41E+02	2.40E+03 ± 2.89E+02	2.35E+03 ± 7.11E+02	4.46E+03 ± 7.16E+02	2.32E+03 ± 2.45E+02
	f-rank h	6.2 +	3.2 ≈	6.6 +	3.8 ≈	5 ≈	3.7 ≈	4.4 ≈	9 +	3.1
F17	mean±std	1.94E+03 ± 2.73E+02	1.64E+03 ± 2.82E+02	2.43E+03 ± 1.73E+02	2.28E+03 ± 2.30E+02	1.80E+03 ± 3.51E+02	1.59E+03 ± 1.75E+02	1.56E+03 ± 5.03E+02	3.36E+03 ± 5.57E+02	1.49E+03 ± 5.77E+02
	f-rank h	4.9 ≈	3.7 ≈	7.4 +	6.8 +	4.2 ≈	3.5 ≈	3.2 ≈	8.4 +	2.9
F18	mean±std	1.84E+06 ± 8.40E+05	2.98E+04 ± 1.56E+04	9.09E+04 ± 2.46E+05	3.15E+04 ± 1.02E+04	4.53E+03 ± 3.35E+03	4.44E+03 ± 2.46E+03	2.18E+05 ± 1.26E+05	6.79E+04 ± 2.05E+04	8.40E+03 ± 5.85E+03
	f-rank h	9 +	5.1 +	4.3 ≈	5.3 +	2.2 ≈	1.9 -	7.9 +	6.9 +	2.4
F19	mean±std	1.14E+03 ± 1.12E+03	2.22E+03 ± 2.81E+03	3.51E+02 ± 3.08E+02	1.04E+03 ± 7.77E+02	9.14E+02 ± 1.15E+03	2.92E+03 ± 2.75E+03	6.19E+03 ± 6.00E+03	6.20E+03 ± 5.74E+03	8.15E+02 ± 1.48E+02
	f-rank h	4.6 ≈	5.1 ≈	2.9 ≈	4.1 ≈	3.7 ≈	7.2 +	6.9 +	7.4 +	3.1
F20	mean±std	1.80E+03 ± 3.26E+02	1.67E+03 ± 2.39E+02	2.44E+03 ± 2.56E+02	2.44E+03 ± 1.99E+02	1.57E+03 ± 2.95E+02	1.89E+03 ± 3.08E+02	1.67E+03 ± 4.01E+02	2.98E+03 ± 4.66E+02	1.90E+03 ± 3.41E+02
	f-rank h	3.5 ≈	3 ≈	7.3 +	7.6 +	2.5 ≈	4.3 ≈	3.5 ≈	8.6 +	4.7
F21	mean±std	3.83E+02 ± 3.60E+01	4.91E+02 ± 4.50E+01	4.78E+02 ± 6.48E+01	7.99E+02 ± 1.67E+02	3.74E+02 ± 1.92E+01	3.57E+02 ± 2.27E+01	3.02E+02 ± 1.92E+01	7.85E+02 ± 9.97E+01	3.43E+02 ± 1.64E+01
	f-rank h	4.1 +	6.7 +	6.3 +	8.4 +	4 +	3.4 ≈	1.1 -	8.4 +	2.6
F22	mean±std	1.51E+04 ± 9.38E+02	2.23E+04 ± 7.00E+02	1.63E+04 ± 7.37E+02	2.56E+04 ± 4.88E+02	1.64E+04 ± 2.20E+03	1.16E+04 ± 1.07E+03	1.20E+04 ± 1.16E+03	1.57E+04 ± 1.19E+03	1.21E+04 ± 1.03E+03
	f-rank h	4.7 +	8 +	6.2 +	9 +	5.8 +	1.8 ≈	2.2 ≈	5.2 +	2.1
F23	mean±std	6.50E+02 ± 1.17E+01	7.54E+02 ± 4.00E+01	7.81E+02 ± 1.56E+01	1.04E+03 ± 1.30E+01	6.63E+02 ± 1.87E+01	6.61E+02 ± 2.18E+01	6.23E+02 ± 2.27E+01	1.25E+03 ± 8.66E+01	6.44E+02 ± 2.90E+01
	f-rank h	3.1 ≈	6.2 +	6.8 +	8 +	4 ≈	3.5 ≈	1.5 ≈	9 +	2.9
F24	mean±std	1.05E+03 ± 2.75E+01	1.24E+03 ± 5.49E+01	1.10E+03 ± 4.01E+01	1.05E+03 ± 1.81E+01	1.04E+03 ± 1.65E+01	1.00E+03 ± 1.81E+01	9.48E+02 ± 1.92E+01	1.92E+03 ± 2.08E+02	1.00E+03 ± 2.41E+01
	f-rank h	5.2 +	8 +	6.9 +	5.3 +	4.2 +	2.7 ≈	1 -	9 +	2.7
F25	mean±std	8.54E+02 ± 3.09E+01	7.81E+02 ± 7.17E+01	7.36E+02 ± 3.85E+01	7.88E+02 ± 8.77E+01	7.11E+02 ± 4.62E+01	7.24E+02 ± 4.21E+01	7.65E+02 ± 3.31E+01	7.56E+02 ± 8.20E+01	7.98E+02 ± 4.79E+01
	f-rank h	8.4 +	5.2 ≈	3.8 -	5.7 ≈	2.9 -	3.2 -	4.7 ≈	5.1 ≈	6
F26	mean±std	5.12E+03 ± 2.65E+02	8.69E+03 ± 9.15E+02	5.87E+03 ± 3.15E+02	5.25E+03 ± 3.46E+02	4.88E+03 ± 2.06E+02	4.51E+03 ± 2.53E+02	3.92E+03 ± 1.91E+02	1.50E+04 ± 2.27E+03	4.40E+03 ± 1.92E+02
	f-rank h	5.3 +	8 +	6.9 +	5.4 +	4.1 +	2.8 ≈	1.2 -	9 +	2.3
F27	mean±std	6.85E+02 ± 1.85E+01	9.35E+02 ± 8.04E+01	8.07E+02 ± 4.43E+01	7.66E+02 ± 5.08E+01	7.14E+02 ± 8.96E+01	7.21E+02 ± 4.04E+01	6.51E+02 ± 1.82E+01	1.07E+03 ± 2.03E+02	7.37E+02 ± 4.91E+01
	f-rank h	2.7 -	8.2 +	6.3 +	5.4 ≈	4.6 ≈	3.4 ≈	1.4 -	8.6 +	4.4
F28	mean±std	9.75E+02 ± 2.28E+02	5.38E+02 ± 3.83E+01	5.41E+02 ± 3.19E+01	5.62E+02 ± 3.57E+01	5.16E+02 ± 1.73E+01	5.34E+02 ± 3.22E+01	5.43E+02 ± 6.81E+00	5.71E+02 ± 3.81E+01	5.49E+02 ± 3.44E+01
	f-rank h	9 +	4.5 ≈	3.9 ≈	6.1 ≈	2.2 -	4.3 ≈	4.1 ≈	5.9 ≈	5
F29	mean±std	2.04E+03 ± 3.37E+02	1.95E+03 ± 2.97E+02	2.60E+03 ± 2.12E+02	2.30E+03 ± 4.72E+02	1.77E+03 ± 2.47E+02	1.88E+03 ± 2.96E+02	1.59E+03 ± 3.57E+02	3.88E+03 ± 3.87E+02	1.94E+03 ± 2.25E+02
	f-rank h	4.7 ≈	4.1 ≈	7.4 +	6.5 +	3 ≈	3.9 ≈	2.4 ≈	9 +	4
F30	mean±std	7.82E+03 ± 2.56E+03	5.13E+03 ± 2.67E+03	3.28E+03 ± 9.58E+02	5.60E+03 ± 2.43E+03	2.43E+03 ± 1.65E+03	2.67E+03 ± 2.16E+02	2.92E+03 ± 1.51E+02	1.05E+04 ± 5.20E+03	2.72E+03 ± 2.57E+02
	f-rank h	7.8 +	6.7 +	3.8 ≈	6.6 +	2.5 ≈	2.6 ≈	3.7 ≈	8.4 +	2.9
	+/≈/−	16/10/4	19/9/2	17/9/4	18/9/3	7/19/4	3/19/8	8/13/9	27/3/0	
	average f-rank	5.80	5.74	5.17	5.86	3.71	3.43	3.95	7.70	3.65

better optimization ability in the middle and later stages of the algorithm, as shown in Fig. 7(h). Finally, the MHDE algorithm consistently demonstrated the fastest convergence speed and achieved the best search results among the presented convergence plots. In general, the MHDE algorithm exhibited a strong global convergence ability and fast convergence speed.

7. Analyzing the adaptive clustering mechanism

The impact of the entropy of the elite distribution on the clustering quantity was analyzed, as shown in Fig. 8, where the red and blue lines denote the variation curves of the entropy of the elite distribution and clustering quantity, respectively.

The entropy calculation determines the concentration of the elite distribution. Consequently, the entropy value for the unimodal function F1 rapidly decreased, as shown in Fig. 8(a). In addition, the number of guided clusters rapidly decreased, enhancing the exploitation ability of the population and facilitating rapid convergence towards the dominant regions.

From the perspective of multimodal functions, such as F19 and F21, the high entropy state in the early stage persisted longer than that in F1, even though the entropy values showed a decreasing trend. Consequently, the algorithm produced more clusters in the early stage, which enhanced the continuous exploratory ability of the population.

Table 6
Statistical results of the ablation experiments on CEC2017.

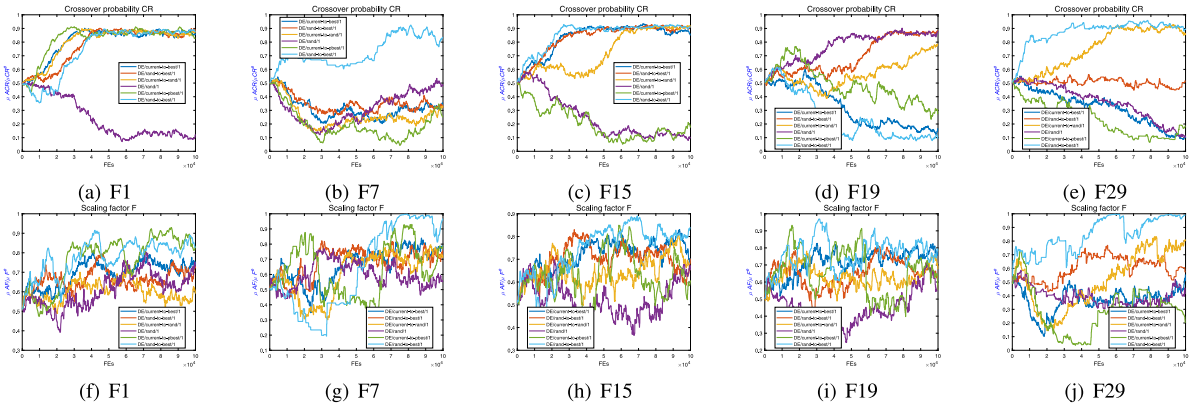
Item	MHDE-Ab1	MHDE-Ab2	MHDE-Ab3	MHDE
+/≈/−	13/12/5	5/21/4	15/14/1	
average f-rank	2.5	2.2	3.1	2.1

However, MHDE assisted the algorithm in dynamically providing a more appropriate number of clusters based on changes in entropy for certain multimodal functions with complex features. For example, the early to middle stages of F11 exhibited rapid convergence towards the dominant region, followed by further exploration in the complex dominant region during the later stages. Additionally, the algorithm maintained a balanced state of entropy in the middle and later stages when optimizing F23, facilitating a more balanced exploration and exploitation, as shown in Fig. 8(e).

Based on the above analysis, it can be concluded that the entropy values of the elite distribution effectively guided the clustering quantity of the population.

8. Ablation experiment

To analyze the impact of its key components on the overall performance of MHDE.



(a) F1 (b) F7 (c) F15 (d) F19 (e) F29

(f) F1 (g) F7 (h) F15 (i) F19 (j) F29

Fig. 9. The variation curves of the CR and F.

The MHDE algorithm three main components: adaptive clustering, individual improvement calculation, and extra-cluster evolution mechanisms.

Therefore, each of these components was individually removed or modified to test the performance of the algorithm. First, regarding the adaptive clustering mechanism, random grouping of the population replaced the clustering mechanism in the MHDE-Ab1 algorithm, and a fixed number of subpopulations were used instead of adaptive clustering. Second, the common individual contribution calculation mechanism was used to replace Eq. (16) in the MHDE-Ab2 algorithm, which is expressed in Eq. (23). Finally, the extra-cluster was not included in the evolution of the population in the MHDE-Ab3 algorithm.

$$Rc_{i,G} = \begin{cases} f(\vec{X}_i^G) - f(\vec{U}_i^G) & f(\vec{U}_i^G) < f(\vec{X}_i^G) \\ 0 & otherwise \end{cases} \tag{23}$$

The statistical results of the ablation experiments are presented in Table 6. The MHDE algorithm outperformed the MHDE-Ab1 algorithm on 13 functions and surpassed the MHDE-Ab3 algorithm on 15 functions. Moreover, the MHDE algorithm only performed worse than the MHDE-Ab1 and MHDE-Ab2 algorithms for five and one functions, respectively. This indicated that the adaptive clustering and extra-cluster mechanisms positively affected the performance of the MHDE algorithm in most cases. Additionally, the performance of the MHDE algorithm was similar to that of the MHDE-Ab2 algorithm on 21 functions. However, the MHDE algorithm still exhibited better performance on five functions. Therefore, the proposed individual improvement calculation mechanism enhanced the optimization ability of the algorithm compared with common mechanisms in certain problems.

9. Analyzing parameters

The performance of the important parameters in the MHDE algorithm were analyzed and the optimal parameter settings were determined.

9.1. Crossover probability CR and scale factor F

The MHDE algorithm provided independent parameter adaptive mechanisms for each mutation strategy. The trends of CR and F for the six mutation strategies on the five test functions were analyzed to determine the differences in the parameters across various mutation strategies, as shown in Fig. 9. There were significant differences in the parameter variation curves among the different mutation strategies. Furthermore, the same mutation strategy exhibited various parameter selection characteristics across different problems. This indicated that the parameter adaptation mechanism in the MHDE algorithm effectively assisted in selecting the appropriate parameters that suited individual mutation strategies and problem environments. This, in turn, enhanced the overall optimization capabilities of the algorithm for different types of problems.

Table 7
Statistical results of f-rank for the MHDE algorithm with different TS values on CEC2017.

	MHDE-TS-5	MHDE-TS-10	MHDE-TS-20	MHDE-TS-40	MHDE-TS-70	MHDE-TS-100
Unimodal Functions	5.5	6.2	9.9	12.4	13.6	15.4
Simple Multimodal Functions	20.5	21.5	24.1	25.8	27.1	28
Hybrid Functions	36.2	35.6	34.6	32.2	35	36.4
Composition Functions	36.4	32.5	33.7	37.8	34.4	35.2
overall f-rank	98.6	95.8	102.3	108.2	110.1	115

Table 8
Statistical results of f-rank for the MHDE algorithm with different NC_{max} and NC_{min} on CEC2017.

	MHDE-5–10	MHDE-5–15	MHDE-5–20	MHDE-10–15	MHDE-10–20	MHDE-15-20
Unimodal Functions	10.6	7.5	7.3	12.8	12.7	12.1
Simple Multimodal Functions	24.3	28.4	25.2	23.3	22	23.8
Hybrid Functions	37.4	35.2	30.2	36.6	35	35.6
Composition Functions	35.7	38.9	34.4	31.3	37.5	32.2
overall f-rank	108	110	97.1	104	107.2	103.7

9.2. Time span for the re-clustering TS

Comparative experiments were conducted on the MHDE algorithm with different TS values to analyze the impact of TS on the performance of the algorithm and determine the optimal parameter settings. The experimental results are presented in Table 7, where the MHDE algorithm with different TS values is named MHDE-TS-N where 'N' represents the value of TS.

Additionally, the overall f-rank was utilized to show the summation of f-rank across all test functions.

Smaller TS values indicated better algorithm performance for unimodal and simple multimodal functions. This suggested that frequent information exchanges within a population were beneficial for optimizing relatively simple problems using this algorithm. However, the impact of TS on MHDE was relatively small for complex problems, such as hybrid and composition functions.

Moreover, the overall f-rank verified that MHDE-TS-10 exhibited the best overall performance in comprehensive testing. Therefore, TS was configured as 10 in the experiments.

9.3. Upper and lower limits of the number of clusters NC_{max} and NC_{min}

Experimental analyses were conducted to investigate the impact of NC_{min} and NC_{max} on the MHDE algorithm. For this experiment, the values 5, 10, 15, and 20 were chosen as the test parameters for

Table 9
Statistical results of f-rank for the MHDE algorithm with different e on CEC2017.

Item	MHDE-e-0.2	MHDE-e-0.4	MHDE-e-0.6	MHDE-e-0.8
Unimodal Functions	8.5	7.3	7.3	6.9
Simple Multimodal Functions	18.4	21.2	17	13.4
Hybrid Functions	32	25.2	20.2	22.6
Composition Functions	28.5	26.3	22	23.2
overall f-rank	87.4	80	66.5	66.1

Table 10
Statistical results of f-rank the MHDE algorithm with different Bp on CEC2017.

	MHDE-Bp-0.2	MHDE-Bp-0.4	MHDE-Bp-0.6	MHDE-Bp-0.8
Unimodal Functions	7.5	6.7	8.7	7.1
Simple Multimodal Functions	15.8	17.4	17.5	19.3
Hybrid Functions	23	23.6	26.6	26.8
Composition Functions	23.8	25.9	24.2	26.1
overall f-rank	70.1	73.6	77	79.3

NC_{min} and NC_{max}. The experimental results are presented in Table 8, where the format MHDE-NC_{min}-NC_{max} is used to denote the algorithm with different parameters. Furthermore, the data presented in Table 8 represent the cumulative statistical sum of the f-rank.

First, considering unimodal functions, a smaller NC_{min} led to better algorithm performance when NC_{max} remained constant. This indicated that having fewer clusters was more favorable for the operation of the algorithm in unimodal problems.

Second, for complex multimodal functions, such as hybrid functions, a larger NC_{max} led to better algorithm performance when NC_{min} was set to five. This demonstrated that a larger NC_{max} was more conducive to improving the global exploration capabilities of the algorithm.

Finally, from the overall f-rank, the MHDE-5-20 exhibited the best algorithm performance. Therefore, NC_{min} and NC_{max} were set to 5 and 20, respectively.

9.4. Proportion of the extra-cluster e

Comparative experiments were conducted using the MHDE algorithm with different e value to analyze the impact of the extra-cluster population ratio e on the algorithm performance. Specifically, e values of 0.2, 0.4, 0.6, and 0.8 were utilized. The experimental results are presented in Table 9, where MHDE-e-N represents the MHDE algorithm with e set as N.

The symbol e represents the proportion of information exchanged between clusters. In the case of unimodal and simple multimodal functions, a higher e value resulted in better algorithm performance. This suggested that increasing the level of information exchange between populations would be beneficial for optimizing simpler problems. However, the algorithm performed best when e was set to 0.6 for the hybrid and composition functions. Lower and higher values of e led to a decline in the algorithm performance, indicating that a balanced value of e was more favorable for optimization in complex problems. Overall, the MHDE algorithm demonstrated the best performance when e was set to 0.8.

9.5. Basic selection probability of mutation strategies Bp

Bp represents the basic selection probability for the mutation operator. A smaller Bp increases the capability of the algorithm to adaptively adjust the selection probability of the mutation operator. However, setting Bp too low can exclude some mutation operators from participating in the population evolution, leading to a disruption in the adaptive mechanism of the algorithm.

Comparative tests were conducted with algorithms using different Bp values to examine the impact of the Bp parameter on the MHDE algorithm. The Bp parameters were set to 0.2, 0.4, 0.6, and 0.8, and the test results are presented in Table 10.

The overall f-rank gradually decreased as Bp decreased from 0.8 to 0.2. Therefore, granting the algorithm greater adaptive adjustment capability effectively enhanced its overall performance. Consequently, Bp was set to 0.2.

10. Time complexity analysis

Understanding the time complexity of an algorithm is crucial for evaluating its efficiency and scalability. Time complexity measures the amount of computational resources required, particularly the time required, for an algorithm to solve a problem as the input size increases. By analyzing the time complexity, insight can be gained regarding the ability of the algorithm to scale with larger inputs to facilitate making informed decisions about its practicality. In subsequent sections, a detailed analysis of the algorithm's time complexity will be provided by considering factors such as loop iterations and the execution time of individual operations. This analysis provided a deeper understanding of the efficiency of the algorithm.

- The time complexities of initializing the population and estimating the fitness of the initial population are expressed as $O(NP \cdot D)$ and $O(NP \cdot f(D))$, respectively.
- The time complexity of computing the number of clusters using LSH is expressed as $O(NP \cdot D + NP) \approx O(NP \cdot D)$.
- The time complexity of clustering population by BLSH is expressed as $O(NP(D + log_2(NP) + 1)) \approx O(NP(\cdot D + log_2(NP)))$.
- The time complexity of intra-cluster evolution is expressed as $O(NP \cdot D + NP \cdot f(D))$.
- The time complexity of constructing the extra-cluster population is expressed as $O(NS)$.
- The time complexity of extra-cluster evolution is expressed as $O(NS \cdot D + NS \cdot f(D) + D \cdot NC + NC \cdot NS)$.

Based on the above analysis, the time complexity of the algorithm during G generations can be determined as: $O(NP \cdot D + NP \cdot f(D) + (G/TS) \cdot NP(D + log_2(NP)) + G \cdot (NP \cdot (D + f(D)) + NS + NS \cdot D + NS \cdot f(D) + NC \cdot D + NC \cdot NS)) \approx O((G/TS) \cdot NP \cdot (D + log_2(NP)) + G \cdot ((NP + NS) \cdot (D + f(D)) + NC \cdot (D + NS)))$

The time complexity testing method defined in "Problem Definitions and Evaluation Criteria for the CEC2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization" (Awad et al., 2016) was employed to evaluate all the participating algorithms and visually illustrate the time complexity of the MHDE algorithm. The experimental results are presented in Table 11.

The MHDE algorithm exhibited a high time complexity. This was primarily owing to the MHDE algorithm employing a population clustering mechanism to evaluate the population state. This mechanism requires additional computational resources to acquire information on the state of the population. However, this information is crucial for the performance of the algorithm. The MHDE algorithm can dynamically adjust the number of clusters by analyzing this state information, thereby enhancing the global search capabilities and optimization performance the algorithm.

In practical applications, these additional computational resources were found to be acceptable because they directly contributed to the outstanding performance of the MHDE algorithm in CEC2017.

In conclusion, the higher time complexity of the MHDE algorithm was a reasonable trade-off for improving the algorithm performance and adaptability. In addition, it enabled the algorithm to excel in solving various types of problems. In the future, efforts will continue to enhance the algorithm while reducing its time complexity and maintaining its exceptional performance.

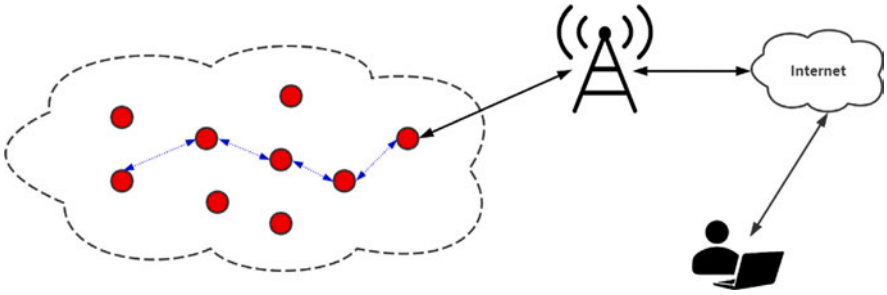


Fig. 10. Wireless sensor network topology.

Table 11
The time complexity of algorithms in different dimensions.

Dimension	SDE	SaDE	JADE	AGDE	EDEV	DEET	DCDE	EHDE	MHDE
30D	196.42	478.95	116.65	78.11	155.72	275.39	229.26	97.73	749.10
50D	315.73	592.23	210.88	138.21	165.26	535.93	375.67	347.39	1118.69
100D	353.86	628.65	280.16	177.03	155.92	782.86	391.01	1371.68	1314.55

Table 12
WSNs deployment scenarios.

Scenarios	Deployment space	Number of sensors	R_s	Problem dimension
Scenario 1	10 × 10	5	2.5	10
Scenario 2	10 × 10	8	2.5	16
Scenario 3	30 × 30	15	5	30
Scenario 4	30 × 30	20	5	40
Scenario 5	100 × 100	30	10	60
Scenario 6	100 × 100	40	10	80

11. Node deployment in wireless sensor networks

Advancements in wireless networking and intelligent sensor technologies have led to the development of wireless sensor devices. Wireless sensor networks (WSNs) are distributed networks comprised of sensor nodes capable of perceiving and detecting the surrounding environment. Information can be collected, processed, and transmitted to the network in a timely manner by deploying WSNs in target areas, ultimately providing feedback to users. The working principle of WSNs is illustrated in Fig. 10. WSNs have a wide range of applications owing to their excellent performances and practical features. WSNs have been widely employed in various fields such as military applications (Felemban, 2013), medical monitoring (Chang et al., 2012), disaster warning (Yadav et al., 2019), environmental monitoring (Baloch et al., 2015), and agricultural production (Bozorgi & Bidgoli, 2019). In the future, WSNs are expected to be present in all aspects of social development and daily life.

Node deployment is a critical issue in the application of WSNs. Insufficient deployment schemes for sensor nodes inevitably result in a significant overlap in the sensing areas. Repetitive sensing in the same area leads to redundant data in WSNs, weakening the data processing capability of the network. Additionally, meaningless monitoring and data forwarding increase the overall power consumption of the network, thereby reducing its lifetime and impacting the service quality of the entire WSN. Moreover, this increases the deployment cost of WSNs. Therefore, a reasonable node deployment scheme for WSNs can effectively improve the energy efficiency of a network and enhance its service quality in the target area.

11.1. WSN coverage model

The sensors were represented as $S = \{s_1, s_2, \ldots, s_n\}$ for the deployment of n homogeneous sensors on a 2D target monitoring area TA, where the position of sensor s_i was represented by the 2D coordinates $s_i = (sx_i, sy_i)$. Sensors were capable of perceiving the spatial region within a certain range around themselves. The specific sensing range was represented by the sensing radius R_s. Moreover, the sensing models of sensors were categorized into two types: binary and probabilistic. In order to closely simulate the real-world working conditions of WSNs, this study adopted a more realistic probabilistic sensing model. Its mathematical formula is expressed in Eqs. (24)–(27).

$$SP(s_i, z) = \begin{cases} 1 & d(s_i, z) \le R_s - R_e \\ \exp\left(\frac{-\lambda_1 \alpha_1^{\beta_1}}{\alpha_2^{\beta_2} + \lambda_2}\right) & R_s - R_e < d(s_i, z) < R_s + R_e \\ 0 & otherwise \end{cases} \tag{24}$$

$$d(s_i, z) = \sqrt{(sx_i - zx)^2 + (sy_i - zy)} \tag{25}$$

$$\alpha_1 = R_e - R_s + d(c_i, z) \tag{26}$$

$$\alpha_2 = R_e + R_s - d(c_i, z) \tag{27}$$

where $z = (zx, zy)$ represents an arbitrary position in the TA, $d(s_i, z)$ represents the Euclidean distance between z and sensor s_i, R_e represents the reliability parameter, and $\beta_1, \beta_2, \lambda_1$ and λ_2 represent the measurement parameters of the sensors. Therefore, the joint sensing probability of n sensors in TA for any point z can be represented by Eq. (28).

$$SP(S, z) = 1 - \prod_{i=1}^{n} \left(1 - SP\left(s_i, z\right)\right) \tag{28}$$

It is impossible to perform sensing detection for all positions in the TA owing to the existence of an infinite number of points z in the TA. Therefore, the TA was divided into a grid of $m \times n$ pixel points to evaluate the coverage of WSNs on the TA. The calculation of coverage on the TA was then converted into the calculation of coverage on these pixel points. The calculation method is expressed in Eq. (29).

$$SP_{\text{cov}} = \frac{\sum_{x=1}^{m} \sum_{y=1}^{n} SP\left(S, (x, y)\right)}{m \times n} \tag{29}$$

11.2. Simulations

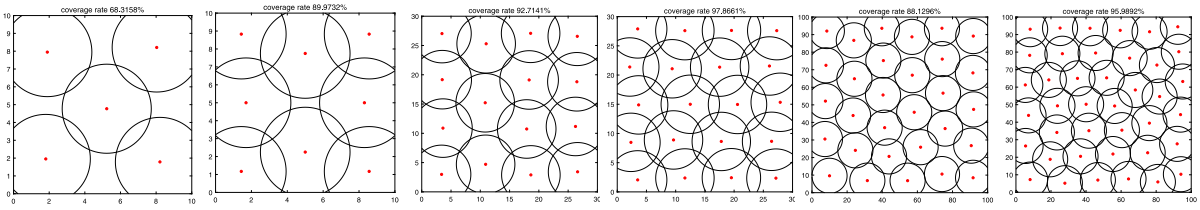
Experiments were conducted to simulate the deployment requirements under six distinct scenarios to validate the effectiveness of the MHDE algorithm in addressing the deployment problem of WSNs. Subsequently, the MHDE algorithm and eight comparative algorithms were used to optimize the sensor deployment schemes for WSNs. Six different deployment scenarios with the problem progressively increasing in difficulty from Scenario 1 to 6 are listed in Table 12.

The β_1, β_2, λ_1, and λ_2 were set to 1, 1.5, 1, and 0, respectively. The optimization results are presented in Table 13.

AGDE performed the best in the relatively simple Scenario 1, whereas the MHDE and DEET algorithms demonstrated excellent performance with similar f-rank values in Scenario 2. The MHDE algorithm exhibited superior performance as the complexity of the problem increased, achieving the best optimization results in Scenarios 3 to 6. The optimal sensor deployment schemes achieved by the MHDE algorithm under different scenarios are shown in Fig. 11.

Table 13
The comparison results of MHDE with other algorithms on WSNs.

Scenarios	Item	SDE	SaDE	JADE	AGDE	EDEV	DEET	DCDE	EFDE	MHDE
Scenario 1(10D)	mean± std	0.68316 ± 1.24E−11	0.68315 ± 1.38E−05	0.68243 ± 3.39E−04	0.68316 ± 1.17E−16	0.68295 ± 6.93E−05	0.68041 ± 5.79E−03	0.68309 ± 2.77E−05	0.68292 ± 2.07E−03	0.68336 ± 2.71E−10
	f-rank h	5.10 ≈	7.00 +	2.20 +	7.50 −	3.20 +	2.20 ≈	6.20 ≈	4.10 ≈	7.50
Scenario 2(16D)	mean± std	0.89861 ± 1.41E−03	0.89534 ± 3.09E−03	0.89039 ± 1.77E−03	0.89762 ± 2.49E−03	0.89584 ± 1.58E−03	0.89864 ± 1.41E−03	0.89445 ± 2.79E−03	0.89401 ± 1.59E−03	0.89918 ± 1.16E−03
	f-rank h	6.20 +	4.10 +	2.20 +	6.00 ≈	3.90 +	6.80 ≈	4.20 +	4.40 +	7.20
Scenario 3(30D)	mean± std	0.92043 ± 6.77E−03	0.90010 ± 4.75E−03	0.89107 ± 2.33E−03	0.89525 ± 2.73E−03	0.90478 ± 2.90E−03	0.92565 ± 1.28E−03	0.89865 ± 3.74E−03	0.91018 ± 4.55E−03	0.92676 ± 7.86E−04
	f-rank h	6.10 +	5.20 +	1.50 +	3.20 +	4.80 +	7.30 ≈	3.83 +	5.47 ≈	7.60
Scenario 4(40)	mean± std	0.96732 ± 5.81E−03	0.95950 ± 2.19E−03	0.95392 ± 1.95E−03	0.95432 ± 2.22E−03	0.96272 ± 1.59E−03	0.97747 ± 7.77E−04	0.96376 ± 1.36E−03	0.96190 ± 2.00E−03	0.97781 ± 5.16E−04
	f-rank h	5.80 +	4.30 +	2.40 +	2.60 +	4.90 +	7.40 ≈	5.67 ≈	4.34 +	7.60
Scenario 5(60)	mean± std	0.84559 ± 2.03E−02	0.81240 ± 4.50E−03	0.80292 ± 5.37E−03	0.80114 ± 4.22E−03	0.81982 ± 4.39E−03	0.86025 ± 2.26E−03	0.82103 ± 4.31E−03	0.82018 ± 3.82E−03	0.87837 ± 2.36E−03
	f-rank h	6.30 +	4.30 +	2.90 +	2.70 +	5.10 +	6.90 +	4.40 +	4.20 +	8.20
Scenario 6(80)	mean± std	0.91249 ± 2.79E−02	0.89933 ± 3.35E−03	0.89506 ± 3.43E−03	0.89076 ± 5.25E−03	0.91154 ± 2.95E−03	0.94132 ± 3.53E−03	0.90330 ± 3.85E−03	0.91604 ± 3.19E−03	0.95899 ± 6.63E−04
	f-rank h	4.70 +	4.30 +	3.70 +	2.60 +	5.50 +	7.00 +	4.60 +	4.80 +	7.80
	+/≈/−	5/1/0	6/0/0	6/0/0	4/1/1	6/0/0	2/4/0	4/2/0	4/2/0	
	average f-rank	5.70	4.87	2.48	4.10	4.57	6.27	4.60	4.77	7.65



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3 (d) Scenario 4 (e) Scenario 5 (f) Scenario 6

Fig. 11. Sensor distributions optimized by MHDE.

12. Conclusion

Based on the various contributions made in this research, the proposed MHDE algorithm demonstrated promising capabilities for addressing the challenges of efficient population exploration and optimization in different solution spaces. MHDE achieved fast and balanced clustering by incorporating the BLSH technique, thereby improving the population partitioning efficiency. The introduction of a population state evaluation method allowed for the adaptive adjustment of the clustering quantity, thereby enhancing the exploration and exploitation abilities of the population. Additionally, a novel method for calculating individual improvements was integrated with the selection probability of mutation strategies and adaptive parameter mechanisms, contributing to improved performance. Moreover, the MHDE algorithm incorporated the construction of EcPop, enabling information exchange between clusters and further enhancing the global search capability of the population.

Through extensive experiments, the MHDE demonstrated its effectiveness and efficiency in solving optimization problems in unimodal and multimodal scenarios. Furthermore, MHDE exhibited remarkable performance when applied to the node deployment problems in WSNs, affirming its suitability for real-world applications. These results established MHDE as a competitive algorithm among the existing DE variants and highlighted its potential for addressing optimization challenges in various domains.

CRediT authorship contribution statement

Xianglong Bu: Conceptualization, Methodology, Investigation, Software, Writing – original draft, Visualization, Data curation, Formal analysis, Validation. **Qingke Zhang:** Software, Supervision, Investigation, Writing – review & editing, Project administration, Funding acquisition. **Hao Gao:** Software, Visualization, Data curation. **Huaxiang Zhang:** Resources, Funding acquisition.

Declaration of competing interest

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

Data availability

The authors do not have permission to share data.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 62006144), the Major Fundamental Research Project of Shandong, China (No. ZR2019ZD03), and the Taishan Scholar Project of Shandong, China (No. ts20190924).

References

Al-Betar, M. A., & Awadallah, M. A. (2018). Island bat algorithm for optimization. *Expert Systems with Applications*, *107*, 126–145. http://dx.doi.org/10.1016/j.eswa.2018.04.024.

Awad, N. H., Ali, M. Z., Suganthan, P. N., Liang, J. J., & Qu, B. Y. (2016). *Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective real-parameter numerical optimization: Technical Report*, Nanyang Technological University, Singapore and Jordan University of Science and Technology, Jordan and Zhengzhou University, Zhengzhou China.

Baloch, J. A., Ahmed, S., Shigrii, S., & Bhatti, J. R. (2015). MUETSenses: A wireless sensor network based indoor environment monitoring system. In *2015 International symposium on antennas and propagation* (pp. 1–4).

Banati, H., & Chaudhary, R. (2016). Enhanced shuffled bat algorithm (EShBAT). In *2016 International conference on advances in computing, communications and informatics* (pp. 731–738). http://dx.doi.org/10.1109/ICACCI.2016.7732134.

Bozorgi, S. M., & Bidgoli, A. M. (2019). HEEC: a hybrid unequal energy efficient clustering for wireless sensor networks. *Wireless Networks*, *25*, 4751–4772. http://dx.doi.org/10.1007/s11276-018-1744-x.

Cai, Z., Gong, W., Ling, C. X., & Zhang, H. (2011). A clustering-based differential evolution for global optimization. *Applied Soft Computing*, *11*(1), 1363–1379. http://dx.doi.org/10.1016/j.asoc.2010.04.008.

Chang, Y.-J., Chen, C.-H., Lin, L.-F., Han, R.-P., Huang, W.-T., & Lee, G.-C. (2012). Wireless sensor networks for vital signs monitoring: Application in a nursing home. *International Journal of Distributed Sensor Networks*, *8*(11), Article 685107. http://dx.doi.org/10.1155/2012/685107.

Chaudhary, R., & Banati, H. (2020). Study of population partitioning techniques on efficiency of swarm algorithms. *Swarm and Evolutionary Computation*, *55*, Article 100672. http://dx.doi.org/10.1016/j.swevo.2020.100672.

Chen, Z.-G., Zhan, Z.-H., Kwong, S., & Zhang, J. (2022). Evolutionary computation for intelligent transportation in smart cities: A survey [review article]. *IEEE Computational Intelligence Magazine*, *17*(2), 83–102. http://dx.doi.org/10.1109/MCI.2022.3155330.

Cui, L., Li, G., Lin, Q., Chen, J., & Lu, N. (2016). Adaptive differential evolution algorithm with novel mutation strategies in multiple sub-populations. *Computers & Operations Research*, *67*, 155–173. http://dx.doi.org/10.1016/j.cor.2015.09.006.

Datar, M., Immorlica, N., Indyk, P., & Mirrokni, V. S. (2004). Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on computational geometry* (pp. 253–262). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/997817.997857.

Deng, L., Li, C., Lan, Y., Sun, G., & Shang, C. (2022). Differential evolution with dynamic combination based mutation operator and two-level parameter adaptation strategy. *Expert Systems with Applications*, *192*, Article 116298. http://dx.doi.org/10.1016/j.eswa.2021.116298.

Felemban, E. (2013). Advanced border intrusion detection and surveillance using wireless sensor network technology. *International Journal of Communications, Network and System Sciences*, *6*(5), 251–259. http://dx.doi.org/10.4236/ijcns.2013.65028.

Gray, R. M. (2011). *Entropy and information theory*. Springer Science & Business Media.

Guendouzi, W., & Boukra, A. (2022). A new differential evolution algorithm for cooperative fuzzy rule mining: application to anomaly detection. *Evolutionary Intelligence*, *15*(4), 2667–2678. http://dx.doi.org/10.1007/s12065-021-00637-3.

Hadjout, D., Sebaa, A., Torres, J. F., & Martínez-Álvarez, F. (2023). Electricity consumption forecasting with outliers handling based on clustering and deep learning with application to the Algerian market. *Expert Systems with Applications*, *227*, Article 120123. http://dx.doi.org/10.1016/j.eswa.2023.120123.

Hongru, L., Jinxing, H., & Shouyong, J. (2018). A hybrid PSO based on dynamic clustering for global optimization. *IFAC-PapersOnLine*, *51*(18), 269–274. http://dx.doi.org/10.1016/j.ifacol.2018.09.311, 10th IFAC Symposium on Advanced Control of Chemical Processes ADCHEM 2018.

Kumar, R., & Kumar, A. (2021). Application of differential evolution for wind speed distribution parameters estimation. *Wind Engineering*, *45*(6), 1544–1556. http://dx.doi.org/10.1177/0309524X21999964.

Kumar, N., Rani, P., Kumar, V., Verma, P. K., & Koundal, D. (2023). TEEECH: Three-tier extended energy efficient clustering hierarchy protocol for heterogeneous wireless sensor network. *Expert Systems with Applications*, *216*, Article 119448. http://dx.doi.org/10.1016/j.eswa.2022.119448.

Li, Y., & Li, G. (2020). Differential evolutionary algorithm with an evolutionary state estimation method and a two-level selection mechanism. *Soft Computing*, *24*(15), 11561–11581.

Li, C., Sun, G., Deng, L., Qiao, L., & Yang, G. (2023). A population state evaluation-based improvement framework for differential evolution. *Information Sciences*, *629*, 15–38. http://dx.doi.org/10.1016/j.ins.2023.01.120.

Liang, J., Qiao, K., Yue, C., Yu, K., Qu, B., Xu, R., Li, Z., & Hu, Y. (2021). A clustering-based differential evolution algorithm for solving multimodal multi-objective optimization problems. *Swarm and Evolutionary Computation*, *60*, Article 100788. http://dx.doi.org/10.1016/j.swevo.2020.100788.

Liu, X.-F., Zhan, Z.-H., & Zhang, J. (2022). Resource-aware distributed differential evolution for training expensive neural-network-based controller in power electronic circuit. *IEEE Transactions on Neural Networks and Learning Systems*, *33*(11), 6286–6296. http://dx.doi.org/10.1109/TNNLS.2021.3075205.

Mallipeddi, R., Suganthan, P., Pan, Q., & Tasgetiren, M. (2011). Differential evolution algorithm with ensemble of parameters and mutation strategies. *Applied Soft Computing*, *11*(2), 1679–1696. http://dx.doi.org/10.1016/j.asoc.2010.04.024, The Impact of Soft Computing for the Progress of Artificial Intelligence.

Meng, Z. (2023). Dimension improvements based adaptation of control parameters in Differential Evolution: A fitness-value-independent approach. *Expert Systems with Applications*, *223*, Article 119848. http://dx.doi.org/10.1016/j.eswa.2023.119848.

Mohamed, A. W., Hadi, A. A., & Jambi, K. M. (2019). Novel mutation strategy for enhancing SHADE and LSHADE algorithms for global numerical optimization. *Swarm and Evolutionary Computation*, *50*, Article 100455. http://dx.doi.org/10.1016/j.swevo.2018.10.006.

Mohamed, A. W., & Mohamed, A. K. (2019). Adaptive guided differential evolution algorithm with novel mutation for numerical optimization. *International Journal of Machine Learning and Cybernetics*, *10*(2), 253–277. http://dx.doi.org/10.1007/s13042-017-0711-7.

Omran, M. G. H., Salman, A., & Engelbrecht, A. P. (2005). Self-adaptive Differential Evolution. In Y. Hao, J. Liu, Y. Wang, Y.-m. Cheung, H. Yin, L. Jiao, J. Ma, & Y.-C. Jiao (Eds.), *Lecture notes in computer science*, *Computational intelligence and security* (pp. 192–199). Berlin, Heidelberg: Springer, http://dx.doi.org/10.1007/11596448_28.

Poikolainen, I., Neri, F., & Caraffini, F. (2015). Cluster-based population initialization for differential evolution frameworks. *Information Sciences*, *297*, 216–235. http://dx.doi.org/10.1016/j.ins.2014.11.026.

Qin, A. K., Huang, V. L., & Suganthan, P. N. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, *13*(2), 398–417. http://dx.doi.org/10.1109/TEVC.2008.927706.

Storn, R., & Price, K. (1997). Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, *11*(4), 341.

Tan, Z., Tang, Y., Li, K., Huang, H., & Luo, S. (2022). Differential evolution with hybrid parameters and mutation strategies based on reinforcement learning. *Swarm and Evolutionary Computation*, *75*, Article 101194. http://dx.doi.org/10.1016/j.swevo.2022.101194.

Tanabe, R., & Fukunaga, A. (2013). Success-history based parameter adaptation for Differential Evolution. In *2013 IEEE congress on evolutionary computation* (pp. 71–78). http://dx.doi.org/10.1109/CEC.2013.6557555.

Toledo, C. F. M., de Oliveira, R. R. R., & Morelato França, P. (2013). A hybrid multi-population genetic algorithm applied to solve the multi-level capacitated lot sizing problem with backlogging. *Computers & Operations Research*, *40*(4), 910–919. http://dx.doi.org/10.1016/j.cor.2012.11.002.

Wang, Y., Cai, Z., & Zhang, Q. (2011). Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Transactions on Evolutionary Computation*, *15*(1), 55–66. http://dx.doi.org/10.1109/TEVC.2010.2087271.

Wang, Z.-J., Zhan, Z.-H., Kwong, S., Jin, H., & Zhang, J. (2021). Adaptive granularity learning distributed particle swarm optimization for large-scale optimization. *IEEE Transactions on Cybernetics*, *51*(3), 1175–1188. http://dx.doi.org/10.1109/TCYB.2020.2977956.

Wu, G., Mallipeddi, R., Suganthan, P. N., Wang, R., & Chen, H. (2016). Differential evolution with multi-population based ensemble of mutation strategies. In *Special issue on discovery science*: *Information Sciences*, In *Special issue on discovery science*: *329*, 329–345.http://dx.doi.org/10.1016/j.ins.2015.09.009,

Wu, G., Shen, X., Li, H., Chen, H., Lin, A., & Suganthan, P. (2018). Ensemble of differential evolution variants. *Information Sciences*, *423*, 172–186. http://dx.doi.org/10.1016/j.ins.2017.09.053.

Xia, X., Gui, L., & Zhan, Z.-H. (2018). A multi-swarm particle swarm optimization algorithm based on dynamical topology and purposeful detecting. *Applied Soft Computing*, *67*, 126–140. http://dx.doi.org/10.1016/j.asoc.2018.02.042.

Yadav, D. K., Karthik, G., Jayanthu, S., & Das, S. K. (2019). Design of real-time slope monitoring system using time-domain reflectometry with wireless sensor network. *IEEE Sensors Letters*, *3*(2), 1–4. http://dx.doi.org/10.1109/LSENS.2019.2892435.

Yang, Q., Yan, J.-Q., Gao, X.-D., Xu, D.-D., Lu, Z.-Y., & Zhang, J. (2022). Random neighbor elite guided differential evolution for global numerical optimization. *Information Sciences*, *607*, 1408–1438. http://dx.doi.org/10.1016/j.ins.2022.06.029.

Yu, W.-J., Shen, M., Chen, W.-N., Zhan, Z.-H., Gong, Y.-J., Lin, Y., Liu, O., & Zhang, J. (2014). Differential evolution with two-level parameter adaptation. *IEEE Transactions on Cybernetics*, *44*(7), 1080–1099. http://dx.doi.org/10.1109/TCYB.2013.2279211.

Zhan, Z.-H., Liu, X.-F., Zhang, H., Yu, Z., Weng, J., Li, Y., Gu, T., & Zhang, J. (2017). Cloudde: A heterogeneous differential evolution algorithm and its distributed cloud version. *IEEE Transactions on Parallel and Distributed Systems*, *28*(3), 704–716. http://dx.doi.org/10.1109/TPDS.2016.2597826.

Zhan, Z.-H., Wang, Z.-J., Jin, H., & Zhang, J. (2020). Adaptive distributed differential evolution. *IEEE Transactions on Cybernetics*, *50*(11), 4633–4647. http://dx.doi.org/10.1109/TCYB.2019.2944873.

Zhang, Q., Bu, X., Zhan, Z.-H., Li, J., & Zhang, H. (2023). An efficient optimization state-based coyote optimization algorithm and its applications. *Applied Soft Computing*, *147*, Article 110827. http://dx.doi.org/10.1016/j.asoc.2023.110827.

Zhang, Q., Gao, H., Zhan, Z.-H., Li, J., & Zhang, H. (2023). Growth Optimizer: A powerful metaheuristic algorithm for solving continuous and discrete global optimization problems. *Knowledge-Based Systems*, *261*, Article 110206. http://dx.doi.org/10.1016/j.knosys.2022.110206.

Zhang, Y.-H., Gong, Y.-J., Zhang, H.-X., Gu, T.-L., & Zhang, J. (2017). Toward fast niching evolutionary algorithms: A locality sensitive hashing-based approach. *IEEE Transactions on Evolutionary Computation*, *21*(3), 347–362. http://dx.doi.org/10.1109/TEVC.2016.2604362.

Zhang, J., & Sanderson, A. C. (2009). JADE: adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, *13*(5), 945–958. http://dx.doi.org/10.1109/TEVC.2009.2014613.

Zhao, F., Zhao, L., Wang, L., & Song, H. (2020). An ensemble discrete differential evolution for the distributed blocking flowshop scheduling with minimizing makespan criterion. *Expert Systems with Applications*, *160*, Article 113678. http://dx.doi.org/10.1016/j.eswa.2020.113678.

Zhong, X., & Cheng, P. (2021). An elite-guided hierarchical differential evolution algorithm. *Applied Intelligence*, *51*, 4962–4983. http://dx.doi.org/10.1007/s10489-020-02091-7.

Zhou, Y.-Z., Yi, W.-C., Gao, L., & Li, X.-Y. (2017). Adaptive differential evolution with sorting crossover rate for continuous optimization problems. *IEEE Transactions on Cybernetics*, *47*(9), 2742–2753. http://dx.doi.org/10.1109/TCYB.2017.2676882.