Received 28 July 2023, accepted 9 August 2023, date of publication 17 August 2023, date of current version 24 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3305925

RESEARCH ARTICLE

QUATRE-PM: QUasi-Affine TRansformation Evolution With Perturbation Mechanism

JUNYUAN ZHANG AND ZHENYU MENG, (Member, IEEE)
Institute of Artificial Intelligence, Fujian University of Technology, Fuzhou 350118, China

Corresponding author: Zhenyu Meng (mzy1314@gmail.com)

This work was supported in part by the Natural Science Foundation of Fujian Province under Grant 2021J05227 and Grant 2023J01347, and in part by the Scientific Research Startup Foundation of the Fujian University of Technology under Grant GY-Z19013.

ABSTRACT Differential Evolution(DE) is a widely used technique to tackle complex optimization problems owing to its easy-implementation and excellent performance, nevertheless, the inborn weakness of the crossover operation has not been solved even in the recent state-of-the-art DE algorithms. There are two commonly used crossover schemes in DE, the exponential crossover and binomial crossover. The exponential crossover is actually a combination of 1-point and 2-point crossover schemes originated with Genetic Algorithm (GA), and it has positional bias because of the dependence on parameter separation. The binomial crossover tackles the positional bias by separating each dimension separately and treating them independently, however, bias still exists from a higher dimensional view, we name it selection bias, and that is the reason why the QUATRE algorithm was proposed. The evolution matrix is the primary component of the QUATRE algorithm which solves the selection bias of DE, however, the previous QUATRE variants still suffer the adaptation of the evolution matrix and can not be able to escape some local optima in complex optimization. Therefore, this paper proposes a new QUATRE with better adaptations of evolution matrix and control parameter, moreover, a perturbation mechanism is firstly proposed for the enhancement of population diversity. The main contributions of our algorithm can be summarized as follows. First, a new generation of evolution matrix is proposed, which can obtain better adaptation to the landscape of the objectives and help the algorithm jump out some local optima. Second, novel adaptations of control parameters are also proposed by incorporating historical memory mechanism and population reduction. Third, a new perturbation mechanism is proposed to enhance the population diversity. In order to validate the proposed algorithm, intensive experiments are conducted under 88 benchmark functions from the universal CEC2013, CEC2014, and CEC2017 test suites in comparison with several excellent DE variants and QUATRE variants, and the results support our superiority.

INDEX TERMS Differential evolution, evolution matrix, parameter adaptation, perturbation mechanism, QUATRE algorithm.

I. INTRODUCTION

The pervasiveness across a wide range of domains, including engineering, finance, economics, logistics, and numerous other fields, optimization problems hold significant importance [1], [2]. Since these complex optimization problems are NP-hard problems, traditional methods are unable to solve them. Thus, numerous meta-heuristic algorithms have been proposed to tackle these optimization problems and achieve high performance. Usually, these meta-heuristic algorithms can be divided into different categories, e.g. Swarm Intelligence and Evolutionary Computation [3], [4], [5]. Swam-based algorithms are primarily inspired by real-life phenomena and biological habits, and their representative algorithms include particle swarm optimization [6], firefly algorithm [7], bee colony algorithm [8], serial cuckoo search algorithm [9] and so on [10], [11]. EAs are inspired by Darwin's theory of evolution and the concept "survival of the fittest", and

The associate editor coordinating the review of this manuscript and approving it for publication was Wai-Keung Fung.

it evolves to better individual with good adaptation to the circumstance. The typical algorithms includes Genetic Algorithm (GA) [12], Differential Evolution (DE) [13], the QUATRE algorithms [14], and so on [15], [16]. Due to the demonstrable superiority, faster convergence speed, and simplicity of implementation, DE has been identified as a promising method for tackling various optimization problems [17], [18], [19].

DE was first proposed by Storn and Price in 1997, which merges GA and the Simulated Annealing (SA) [20]. Based on the basic operations of GA, DE also inherits the three primary operations: mutation, crossover, and selection. These operators are performed iteratively throughout the whole evolution. Usually, different combinations of mutation strategies and crossover schemes significantly affect the overall performance of different DE algorithms [21], [22], [23]. Meanwhile, DE has three essential control parameters with in the evolutionary process: population size (ps), mutation factor (F), and crossover rate (Cr). Control parameters F and Cr should be carefully assigned for generating promising offsprings. Different control parameter settings will affect the final performance by affecting the trial vector candidates [24], [25], [26]. However, without some priori knowledge, tuning suitable parameters is unavoidably a trial-and-error process, which dramatically reduces algorithm efficiency and effectiveness. To tackle this issue, many scholars have focused on proposing adaptive or self-adaptive parameter control and proposed many powerful DE variants, such as jDE [27], JADE algorithm [28], LSHADE algorithm [29], LPamDE [30] and so on [31], [32], [33]. Nearly all these excellent DE papers recommend binomial crossover as their default scheme, however, this crossover scheme still exists weakness.

There are two commonly used crossover schemes in DE: the exponential and the binomial crossover [34]. The exponential crossover is actually a combination of 1-point and 2-point crossover schemes originated with GA, and it has positional bias because of the dependence on parameter separation. The binomial crossover tackles the positional bias by separating each dimension separately and treating them independently, however, bias still exists from a higher dimensional view [35]. Fig.1 depicts the possible trial vector candidates of the target vector in 3-D view, and it can be seen that all the vertices except for the location of the target vector $X_{i,g}$ can be the trial vector candidates. In order to maintain the unbiasedness and efficiency of the algorithm, the following two rules need to be satisfied:

- If no prior knowledge obtained, the probability of selecting each trial vector candidate should be equal.
- If some prior knowledge was obtained, the better performing trial vector candidates with high probability should be of high preference.

Taken the above mentioned two conditions into consideration, bias still exists in the binomial crossover from the spatial search view though it tackles the positional bias by treating each dimension separately and independently. That

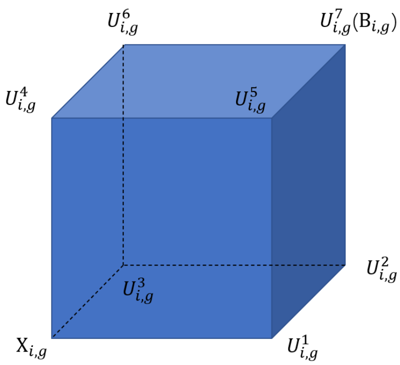


FIGURE 1. The trial vector candidates in DE/QUATRE from the 3-D view.

is also the reason why the QUasi-Affine TRansformation Evolutionary(QUATRE) algorithm [14] was proposed in 2016, by incorporating the evolution matrix M instead of using the crossover operation. The canonical QUATRE algorithm employed fixed generation of evolution matrix in the consideration of equal selection of trial vector candidates, however, its performance was not satisfactory. Later, a C-QUATRE algorithm [36] and S-QUATRE algorithm [37] were proposed by incorporating pair-wise competition and sort-based evolution for performance improvement. In these two algorithms, the number of function calls in each generation reduced by half and the total number of generation doubles. The QUATRE-EAR algorithm proposed a general convention in the evolution matrix calculation, and the calculation of evolution matrix can be converted into the calculation of the number of the k^{th} vector, $N(k)$, of the lower triangular matrix. Moreover, novel adaptations of the evolution matrix and the control parameters are also proposed in the QUATRE-EAR algorithm. The E-QUATRE algorithm [38] further improved the performance of QUATRE-EAR by incorporating adaptations of evolution matrix and the population size reduction. The IS-QUATRE algorithm [39] seeks the potential possibility selecting the trial vector candidates from the internal of the hyper-cube and good performance was obtained in high dimensional optimization. Although these QUATRE variants enhance the algorithm's performance, they still encounter two challenges. Firstly, these QUATRE variants have to obtain equilibrium of exploration and exploitation, and still suffers stagnation problem during the evolution. Secondly, the adaptation of the evolution matrix generation needs improvement and the current adaptations are not flexible.

To solve the above problem and improving the overall performance, this paper proposes a new QUATRE algorithm. The proposed QUATRE is compared with the canonical QUATRE algorithm and the latest QUATRE variants, the state-of-the-art DE algorithms, under three test suites of CEC2013 [40], CEC2014 [41], and CEC2017 [42] with 88 benchmark problems, and the results show good competitiveness. The main contributions of this paper are:

1) A new adaptation scheme of an evolution matrix is proposed. This scheme can adapt the evolution

matrix more rapidly and accurately by employing the knowledge obtained during the evolution especially when performance improvement is not sufficient.

2) A new population perturbation mechanism is proposed. When the population fall into stagnation, previously discarded inferior vectors are used for helping the population jump out the local optima.
3) Two excellent parameter controls are incorporated into our QUATRE-PM algorithm. The mechanism of the historical memory was introduced in the QUATRE-PM algorithm to avoid poor adaptation direction, and the linear population size reduction was incorporated into the QUATRE-PM algorithm, which can balance the exploration and exploitation during the different stages of the evolution.

The remainder of this paper is structured as follows: In Section II, the related QUATRE algorithms will be presented. In Section III, the proposed QUATRE algorithm will be described in detail. In Section IV, experimental analysis and comparisons of these state-of-the-art algorithms under 88 universal benchmarks from CEC2013, CEC2014, and CEC2017 test suites will be performed. Finally, conclusion of the paper will be given in Section V.

II. RELATED WORK

In this section, we introduce some closely related algorithms of our QUATRE-PM. In these algorithms, the population is denoted as $X = [X_{1,g}, X_{2,g}, \ldots, X_{i,g}, \ldots, X_{ps,g}]$, where $X_{i,g}$ denotes the vector of the i^{th} individual in the g^{th} iteration of the population. Among these vectors, each is composed of D-dimensional parameters, here D denotes the dimensionality of the optimization problem. Accordingly, the i^{th} vector can be further written as $X_{i,g} = [x_{i,1,g}, x_{i,2,g}, \ldots, x_{i,d,g}, \ldots, x_{ps,D,g}]$, where $x_{i,d,g}$ denotes the d^{th} parameter in $X_{i,g}$. The upper bound of each parameter $x_{i,d,g}$ is usually denoted as x_d^{upper}, and the lower bound is denoted as x_d^{lower}.

A. THE CANONICAL QUATRE ALGORITHM

The canonical QUATRE tackled the inborn bias of DE by replacing the crossover operator in the DE with an evolution matrix which implemented unbiasedness exploration of the solution space. There are four operations involved in QUATRE algorithms, and they are initialization, mutation, evolution, and selection. Among the four operations, mutation, evolution, and selection are repeated in a circle before algorithm termination.

1) INITIALIZATION

The initialization stage, the individuals in the QUATRE algorithm is randomly initialized by scattering into the whole search space, and Eq. (1) gives the detailed implementation of the initialization:

$$x_{i,d,0} = x_d^{lower} + rand \cdot (x_d^{upper} - x_d^{lower}) \tag{1}$$

where rand denotes a uniformly distributed random number in [0, 1].

2) MUTATION

The mutation operation aims to generate a corresponding donor vector for each target vector. In the evolutionary process, the donor vectors provide the general evolutionary direction for the target vectors, and there are seven commonly used mutation strategies in the literature and these strategies are also summarized int Table 1 where the canonical QUATRE algorithm employed the first mutation strategy. Then, the mutation matrix can be formulated via Eq. (2):

$$B_g = [B_{1,g}, B_{2,g}, \ldots, B_{i,g}, \ldots, B_{i,ps}] \tag{2}$$

where $B_{i,g}$, $B_{i,g} = [b_{i,1,G}, b_{i,2,g}, \ldots, b_{i,d,g}, \ldots, b_{i,D,g}]$, denotes the i^{th} donor vector of the population in the g^{th} iteration. The symbol $X_{best,g}$ in the mutation strategy denotes the vector obtained global best value in the g^{th} population, $X_{r_1,g}$ and $X_{r_2,g}$ denote two vectors of randomly selected individuals in the population of the g^{th} generation. F in the mutation strategy denotes the mutation factor, which is a fixed constant value in the canonical QUATRE algorithm. Boundary restriction is necessary and this confirms that all the individuals are within the search domain. Once the individuals are beyond the region, adjustment is launched according to Eq. (3):

$$\begin{cases} b_{i,d,g} = \dfrac{b_{i,d,g} + x_d^{upper}}{2} & if\ b_{i,d,g} > x_d^{upper} \\ b_{i,d,g} = \dfrac{b_{i,d,g} + x_d^{lower}}{2} & if\ b_{i,d,g} < x_d^{lower} \end{cases} \tag{3}$$

3) EVOLUTION

Instead of employing the crossover operation, the QUATRE algorithms primarily employ evolution matrix in implementing crossover operation during the evolution. The core of the evolution matrix is a boolean matrix, and the detailed evolution is given in Eq. (4):

$$\begin{cases} U_g = \overline{M} \otimes X_g + M \otimes B_g \\ U_g = [U_{1,g}, U_{2,g}, \ldots, U_{i,g}, \ldots, U_{ps,g}] \end{cases} \tag{4}$$

where U_g denotes the trial matrix at the g^{th} iteration, M denotes the evolution matrix, and $\overline{M}$ denotes the matrix obtained after the component-wise "NOT" operation of M. In order to implement the unbiased search of the solution space, two initialization schemes of the evolution matrix M are provided in the original QUATRE algorithm. The first is to perform equal selection probability for all $2^D - 1$ possibilities. This scheme generally requires setting the ps to $2^D - 1$, and then the i^{th} row of the initialization evolution matrix M_{init} is the inverse order of the binary representation of i. The second scheme is to make the number of parameters obtained from the donor vectors equal in selection probability. Accordingly, the initial evolution matrix is composed by multiple lower triangular matrices, and this scheme generally requires that the population ps equals to $n \cdot D$, where n is the number of piled

TABLE 1. The seven schemes of the calculation of matrix B in the QUATRE algorithm.

No.	*QUATRE*/*x*/*y*	Equation
1	*QUATRE*/best/1	$B_{i,g} = X_{gbest,g} + F_i \cdot (X_{r_1,g} - X_{r_2,g})$
2	*QUATRE*/rand/1	$B_{i,g} = X_{r_0,g} + F_i \cdot (X_{r_1,g} - X_{r_2,g})$
3	*QUATRE*/target/1	$B_{i,g} = X_{i,g} + F_i \cdot (X_{r_1,g} - X_{r_2,g})$
4	*QUATRE*/target-to-best/1	$B_{i,g} = X_{i,g} + F_i \cdot (X_{gbest,g} - X_{i,g}) + F_i \cdot (X_{r_1,g} - X_{r_2,g})$
5	*QUATRE*/target-to-rand/1	$B_{i,g} = X_{i,g} + F_i \cdot (X_{r_1,g} - X_{i,g}) + F_i \cdot (X_{r_2,g} - X_{r_3,g})$
6	*QUATRE*/rand/2	$B_{i,g} = X_{r_0,g} + F_i \cdot (X_{r_1,g} - X_{r_2,g}) + F_i \cdot (X_{r_3,g} - X_{r_4,g})$
7	*QUATRE*/best/2	$B_{i,g} = X_{gbest,g} + F_i \cdot (X_{r_1,g} - X_{r_2,g}) + F_i \cdot (X_{r_3,g} - X_{r_4,g})$

lower triangular matrices. During each generation, two steps are required to transform M_{init} into M: the first step permutes the dimensional elements of the row vectors in the matrix, with the row number unchanged in the population, and the second step permuted the row numbers of these vectors of the matrix. Eq. (5) describes the transformation from M_{init} to M:

$$M_{init} = \begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ & & \vdots & \\ 1 & 1 & \ldots & 1 \\ 1 & & & \\ 1 & 1 & & \\ & & \vdots & \\ 1 & 1 & \ldots & 1 \end{bmatrix} \sim \begin{bmatrix} 1 & 1 & & \\ & \vdots & & \\ 1 & 1 & \ldots & 1 \\ 1 & & & \\ & & 1 & \\ & \vdots & & \\ 1 & \ldots & 1 & \\ & 1 & & \end{bmatrix} = M \quad (5)$$

4) SELECTION

Selection is conducted after individuals' evolution in the QUATRE algorithm, and the selection operation in QUATRE is the same as in DE. By comparing the fitness values of the trial vector $U_{i,g}$ and the corresponding target vector $X_{i,g}$, the excellent $U_{i,g}$ will replace $X_{i,g}$ as the new target vector. The implementation of selection operation is given in Eq. (6):

$$X_{i,g+1} = \begin{cases} U_{i,g} & \text{if } f(U_{i,g}) < f(X_{i,g}) \\ X_{i,g} & \text{otherwise} \end{cases} \quad (6)$$

where $f(\cdot)$ denotes the fitness value of the vector. The selection operator is the last operation in each generation, and the new population consists of all survival individuals. If the termination criterion is satisfied, the vector with the best fitness will be output as the optimal solution.

B. THE QUATRE-EAR ALGORITHM

The QUATRE-EAR algorithm is an upgraded version of the original QUATRE. An adaptive evolution matrix generation scheme and a new mutation strategy are proposed in QUATRE-EAR, meanwhile, an adaptive scheme of parameters is introduced as well. By incorporating these these improved schemes, the efficiency and performance of the algorithm have been substantially enhanced.

1) MUTATION STRATEGY

QUATRE-EAR employed the "QUATRE/target-to-best/1" mutation strategy with time stamp technique, which is more competitive than the "QUATRE/best/1" mutation strategy in the original QUATRE. It guides the evolution of the population through excellent individuals and simultaneously establishes external archive to increase the diversity of the population. The external archive records the target vectors that failed in the selection operator, and random elimination scheme is used after discarding the too-old inferior solutions when the number of inferior solutions exceeds the size of the external archive. The mutation strategy is expressed in Eq. (7):

$$\begin{cases} B_{i,g} = X_{i,g} + F_i \cdot (X^{p}_{gbest,g} - X_{i,g}) + F_i \cdot (X_{r_1,g} - \widetilde{X}_{r_2,g}) \\ B_g = [B_{1,g}, B_{2,g}, \ldots, B_{i,g}, \ldots, B_{ps,g}] \end{cases} \quad (7)$$

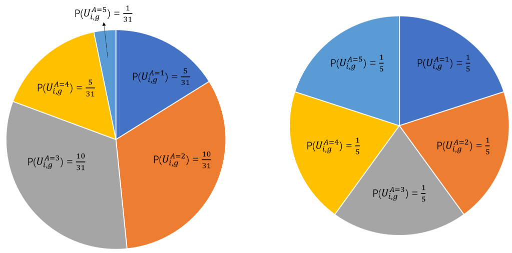
where $B_{i,g}$ denotes the donor vector, $X^{p}_{gbest,g}$ denotes the vector randomly selected from the top p% of excellent individuals in the population, $X_{r_1,g}$ denotes the randomly selected individuals in the population, $\widetilde{X}_{r_2,g}$ denotes the randomly selected individuals from the union of the population and external archive, and F_i is the mutation factor which is associated with the i^{th} individual.

2) EVOLUTION MATRIX *M*

Although the generation of initial evolution matrix M_{init} is easy and time-efficient in the original QUATRE algorithm, it lacks a good adaptation to the landscapes of different optimization problem. Therefore, in the QUATRE-EAR algorithm, selection probability $P(U^{A=k}_{i,g})$ is proposed to generate the M_{init}, which indicates that the i^{th} trial vector has k parameters inherited from the donor vector during the g^{th} iteration. For the first initialization scheme, $P(U^{A=k}_{i,g})$ is calculated as Eq. (8):

$$\begin{cases} P(U^{A=k}_{i,g}) = \dfrac{\frac{D!}{(D-k)!}}{2^D - 1}, & k \in [1, D] \\ P(U^{A\le k}_{i,g}) = \sum_{i=1}^{k} P(U^{A=k}_{i,g}), & k \in [1, D] \\ P(U^{A>k}_{i,g}) = 1 - P(U^{A\le k}_{i,g}), & k \in [1, D] \end{cases} \quad (8)$$

where $D!$ and (D-k)! denote the factorial of D and $(D-k)$ respectively. Then, the number, $N(k)$, of trial vectors in which there are k parameters inherited from the donor vector can be obtained according to Eq. (9). An example of the matrix,



(a) The first scheme (b) The second scheme

FIGURE 2. Comparison of $P(U_{i,g}^{A=k})$ for two scheme M_{init} initialization in $5-D$.

M_{init} with $ps = 100$ and $D = 9$ is given in Eq. (10):

$$N(k) = \begin{cases} \lceil ps \cdot P(U_{i,g}^{A>k-1}) \rceil, & if \quad k = D \\ \lceil ps \cdot P(U_{i,g}^{A>k-1}) \rceil - \sum_{i=k+1}^{D} N(i), & if \quad k \in [2, D-1] \\ ps - \sum_{i=k+1}^{D} N(i), & if \quad k = 1 \end{cases} \tag{9}$$

$$M_{init} = \begin{bmatrix} 1: & 1,0,0,0,0,0,0,0,0 \\ 7: & \begin{cases} 1,1,0,0,0,0,0,0,0 \\ \vdots \ldots \vdots \ldots \vdots \ldots \vdots \ldots \vdots \\ 1,1,0,0,0,0,0,0,0 \end{cases} \\ 17: & \begin{cases} 1,1,1,0,0,0,0,0,0 \\ \vdots \ldots \vdots \ldots \vdots \ldots \vdots \ldots \vdots \\ 1,1,1,0,0,0,0,0,0 \end{cases} \\ N(k): & \vdots \ldots \vdots \ldots \vdots \ldots \vdots \ldots \vdots \\ 1: & 1,1,1,1,1,1,1,1,0 \\ 1: & 1,1,1,1,1,1,1,1,1 \end{bmatrix}_{100 \times 9} \tag{10}$$

The second scheme is to maintain equal selection probability of the different number of parameters inherited from the donor vectors. Therefore, M_{init} is initialized by multiple piles of lower triangular matrices, and this initialization can satisfy $P(U_{i,g}^{A=1}) = P(U_{i,g}^{A=2}) = \ldots = P(U_{i,g}^{A=k}) = \ldots = P(U_{i,g}^{A=D}) = \frac{1}{D}$. When the population is not a multiple number of the dimension D, e.g. $ps = j \cdot D + k$, then the left k vectors in M_{init} are selected from the first j vectors of the lower triangular matrix. In Fig.2, the selection probabilities, $P(U_{i,g}^{A=k})$, are compared for the two different initialization scheme of M_{init} for 5-D optimization.

3) PARAMETER ADAPTATION

In QUATRE-EAR, there is only two control parameters, F and ps. As fixed population is employed during the evolution, and the parameter control is only focused on F. The adaptation scheme of F in QUATRE-EAR is shown in Eq. (11):

$$F_i = randc(\mu_F, 0.2) \tag{11}$$

where *randc* denotes Cauchy distribution, fixed scale parameter is used, $\sigma_F = 0.2$, in the Cauchy distribution and μ_F can be updated according to Eq. (12):

$$\begin{cases} \Delta f_i = f(U_{i,g}) - f(X_{i,g}) \\ \omega_i = \dfrac{\Delta f_i}{\sum_{F_i \in S_F} \Delta f_i} \\ \mu_F = \dfrac{\sum_{F_i \in S_F} \omega_i \cdot F_i^2}{\sum_{F_i \in S_F} \omega_i \cdot F_i} \end{cases} \tag{12}$$

where S_F denoted the set of parameters F which associates with the survival individuals during the evolution.

C. THE E-QUATRE ALGORITHM

The E-QUARE algorithm [38] is a further extension of the QUATRE-EAR algorithm by improving the adaptation scheme of evolution matrix and parameter control.

1) MUTATION STRATEGY

The E-QUATRE algorithm employs the same mutation strategy as the QUATRE-EAR algorithm. In order to balance exploration and exploitation during the evolution process, a scheme of decreasing the proportion of elite individual p in the population is introduced into the mutation strategy of E-QUATRE, and the decrement of p is given as follows in Eq. (13):

$$p = \frac{p_{\min} - p_{\max}}{nfe_{\max}} \cdot nfe + p_{\ max} \tag{13}$$

where p_{max} denotes the initial/maximum setting, $p_{\max} = 0.2$, $p_{\min}$ denotes the terminal/minimum setting $1/ps$, $nfe_{\max}$ denotes the max number of function evaluations, and nfe denotes the current number of function evaluations.

2) EVOLUTION MATRIX M

E-QUATRE further improves on QUATRE-EAR regarding the adaptive generation of evolution matrix M. The improvement is mainly in two aspects. First, the scheme of calculating the number of individuals by Eq. (9) in QUATRE-EAR is replaced by the generation of roulette wheel. Second, the population is divided into different sub-populations, and the excellent sub-population is allowed to record more individuals during the evolution by changing the selection probabilities in Eq. (14):

$$\begin{cases} r_j = \begin{cases} \dfrac{ns_j^2}{ns \cdot (ns_j + nf_j)} & \text{if } ns_j > 0 \\ \epsilon & \text{otherwise} \end{cases} \\ ns = \sum_{j=1}^{2} ns_j \\ P(j) = \dfrac{r_j}{\sum_{j=1}^{k} (r_j)} \end{cases} \tag{14}$$

where ns_j and ns_j denoted the number of winner and loser individuals in the j^{th} population, and the value of $P(j)$ is restricted in [0.3, 0.7].

3) PARAMETER ADAPTATION

E-QUATRE employs a dimensional improvement based adaptation in updating the distribution of mutation factor μ_F instead of employing fitness-based ones in QUATRE-EAR. The detailed adaptation is given below in Eq. (15):

$$\begin{cases} \triangle loc_i = loc(U_{i,g} - X_{i,g}) \\ k_i = \dfrac{std(\triangle loc_i)}{\sum_{F_i \in S_F} std(\triangle loc_i)} \\ \mu_F = \dfrac{\sum_{F_i \in S_F} k_i \cdot F_i^2}{\sum_{F_i \in S_F} k_i \cdot F_i} \end{cases} \tag{15}$$

where $loc(U_{i,g} - X_{i,g})$ denotes locating the corresponding parameters of the D-dimensional vector, $U_{i,g}$-$X_{i,g}$, and $std(\cdot)$ denotes the standard deviation of the set.

III. THE PROPOSED QUATRE ALGORITHM

In this section, the novel QUATRE-PM algorithm will be described in detail. The whole algorithm is divided into four parts: the first part presents the mutation strategy; the second part gives the adaptive scheme of generating the evolution matrix M; the third part describes the parameter control strategy; and the fourth part provides the perturbation mechanism.

A. MUTATION STRATEGY

In DE algorithms, the mutation strategy "DE/target-to-pbest/1/bin" with external archive proposed in JADE [28] can balance exploitation and exploration and consequently has been widely used in recent state-of-the-art DE variants. One of the reasons for the excellent performance of this strategy is the use of an external archive. However, it still exists limitation that some archived individuals remain unchanged during several consecutive generations, and this may hamper the diversity of trial vectors. The "QUATRE/target-to-best/1" with time stamp technique in QUATRE-EAR [43] effectively resolves this problem and produces superior outcomes. Therefore, the same mutation strategy is employed in our QUATRE algorithm, and the details of the mutation strategy are given in Eq. (16):

$$\begin{cases} B_{i,g} = X_{i,g} + F_i \cdot (X^p_{gbest,g} - X_{i,g}) + F_i \cdot (X_{r_1,g} - \widetilde{X}_{r_2,g}) \\ B_g = [B_{1,g}, B_{2,g}, \ldots, B_{i,g}, \ldots, B_{ps,g}] \end{cases} \tag{16}$$

where $X^p_{gbest,g}$ denotes the target vector selected from the top $p\%$ individuals, $X_{r_1,g}$ denotes the vector randomly selected from the population, and $\widetilde{X}_{r_2,g}$ denotes the vector randomly selected from the union of the current population and external archive. The external archive stores the inferior solutions of the target individuals during the evolution, and its size $|A|$ is adjusted according to the current population size, ps, shown in Eq. (17):

$$|A| = arc \cdot ps \tag{17}$$

TABLE 2. The mechanism of historical memory and its selection probabilities P_h.

Index	1	2	...	h	...	H
P	P_1	P_2	...	P_h	...	P_H

where arc denotes a constant defining the ratio of external archive $|A|$ to the whole population ps. In the "QUATRE/target-to-best/1" mutation strategy with a time stamp mechanism, each vector entering the external archive is set to a initial time-stamp value T_0 and the value will be decreased by one in each generation. When the external archive population size exceeds $|A|$, the vectors with $T_0 < 0$ are deleted preferentially. After the vectors with $T_0 < 0$ are deleted first and random elimination scheme is performed in order to remove the exceeded inferior individuals.

B. ADAPTATION OF EVOLUTION MATRIX M

The original QUATRE algorithm made a balance between unbiased search and efficient search of the solution space during the evolution, however, the generation of M_{init} lacks of adaptations to the landscape of the objective. In order to obtain better performance, M_{init} needs to be adjusted regarding different characteristics of the target problems. Therefore, QUATRE-EAR proposed the adaptive scheme of M_{init}, and E-QUATRE further improved the adaption of M in QUATRE-EAR and consequently obtained good performance. However, the adaptive scheme of M_{init} in E-QUATRE still needs to be improved for a better adaption to the landscape of the objective. First, at the beginning of the search, it is impossible to calculate the appropriate evolution matrix quickly and accurately due to lack of enough knowledge of the evolutionary process, which inevitably leading to a large amount of computational resources being wasted. Second, the excessive computation of the dominance accumulation leads to the evolution matrix being almost fixed and it is hard to change in the late stages of the search. In order to tackle these issues, novel adaptation of the evolution matrix is proposed in our QUATRE-PM algorithm.

In the novel QUATRE-PM algorithm, the historical memory is employed in the adaptation of evolution matrix to avoid being trapped in failure evolution direction. The historical memory and its corresponding selection probabilities are given in Table 2. In the memory, there are H entries, and each entry P_h is a probability vector which denotes the selection probability of each dimensional parameters, $P_h = [p_{h,1}, p_{h,2}, \ldots, p_{h,d}, \ldots, p_{h,D}]$. During each iteration, a vector is randomly selected from historical memory, and the initial evolution matrix M_{init} will be generated through selection of these vectors. The k^{th} parameter $p_{h,k}$ in the P_h vector is determined by roulette wheel selection, and it indicates that the target vector inherits k parameters from the donor vector. Figure 3 depicts the process of generating evolution matrix M via the selection probabilities of historical memory P_h.

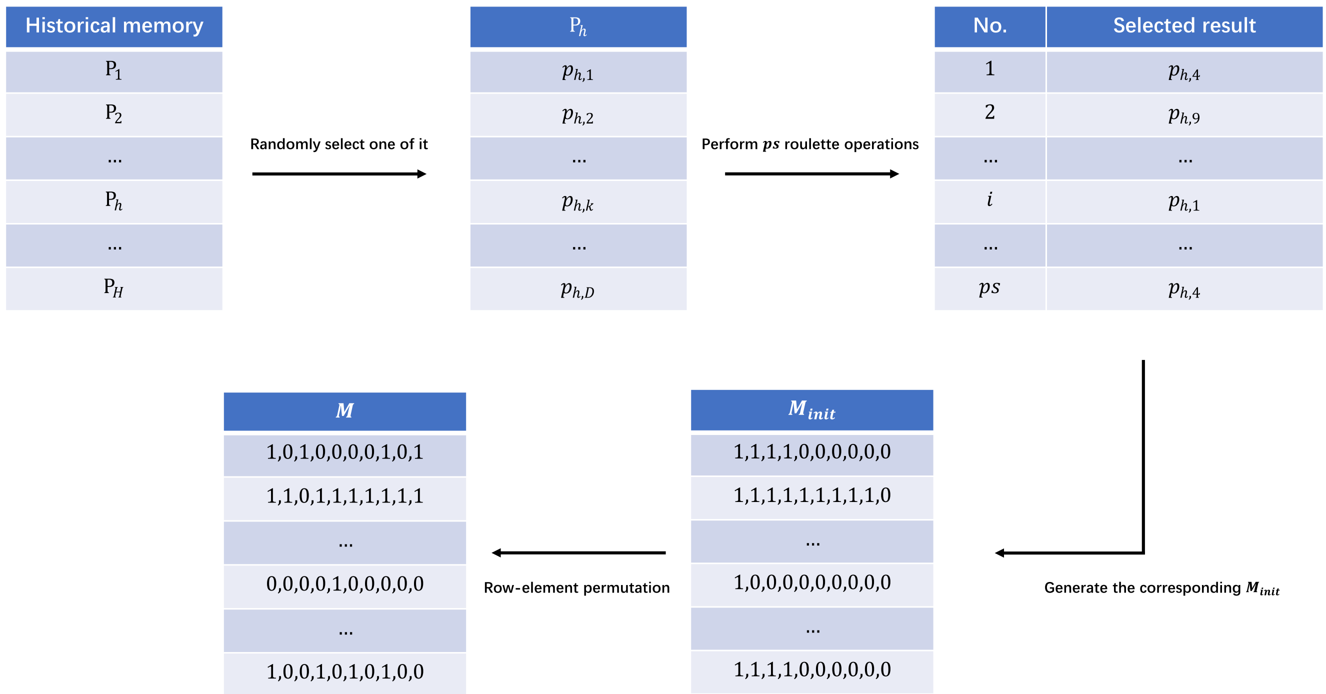


FIGURE 3. The picture depicts the transformation process of generating an evolution matrix *M* via the mechanism of historical memory of probabilities P_h.

The initial value of $P_h(h = 1, 2, \ldots, i, \ldots, D)$ is initialized by setting $p_{h,1} = p_{h,1} = \ldots. = p_{h,k} = \ldots = p_{h,H} = \frac{1}{D}$, and the inspiration of the P_h adaption is to get more suitable value in the subsequent iteration. In the first iteration, h is initialized to 1 and it increases by 1 during each iteration. When the index h satisfies $h > H$, h will be reset to 1. Based on the adaptation of the evolution matrix in E-QUATRE, the value $p_{h,k}$ which gains an advantage in the evolution process, is enhanced to expect some reduction of computational resources. When all the trial vectors fail to obtain better fitness values, the adaptation of P_h will be adjusted to expect obtaining better trial vectors in the next iteration. Eq. (18) describes the update scheme of $p_{h,k}$ which can yield better individuals during the evolution process:

$$\begin{cases} r_k = \begin{cases} \dfrac{\frac{ns_k}{ns} \cdot \frac{\sum_{i \in S_k} w_i}{ns_k}}{\sum\limits_{k=1}^{D} v_k} & if\ ns_k > 0 \\ \dfrac{1}{\pi} \cdot \dfrac{\frac{D}{20}}{(k-\alpha_g)^2 + (\frac{D}{20})^2} \cdot \dfrac{r_{\alpha_g}}{sum_h}, & otherwise \end{cases} \\ v_k = (r_k)^2 \\ p_{h,k} = \dfrac{v_k}{\sum\limits_{k=1}^{D} v_k} \end{cases} \tag{18}$$

where ns denotes the number of trial vectors achieved better fitness, ns_k denotes the number of trial vector with k parameters inherited from the donor vector in the ns trial vectors, S_k denotes the set of success trial vectors in which k parameters are inherited from the donor vector, α_g denotes the value of k that maximizes r_k in the g^{th} iteration, sum_h denotes the sum of all $p_{h,k}$ in P_h with $ns_k > 0$, and w_i denotes the weight of dimension improvement, which is used for the update of the i^{th} individual in the population. The calculation of the weight of the i^{th} individual is given in Eq. (19):

$$\begin{cases} \Delta loc_i = loc(U_{i,g} - X_{i,g}) \\ w_i = \dfrac{std(\Delta loc_i)}{\sum_{i \in S} std(\Delta loc_i)} \end{cases} \tag{19}$$

where $loc(U_{i,g} - X_{i,g})$ denotes locating the parameters of the vector $U_{i,g} - X_{i,g}$ according to the evolution matrix, and $std(\cdot)$ denotes the calculation of the standard deviation. Sometimes, none of the trial vectors can obtain more suitable fitness during a certain evolution process, and we suppose that the current probabilities P_h is no longer appropriate. Therefore, a new probability vector $p_{h,k}$ is regenerated according to Eq. (20):

$$\begin{cases} v_k = \dfrac{1}{\pi} \cdot \dfrac{\frac{D}{10}}{(k-\alpha_{g-1})^2 + (\frac{D}{10})^2} \\ p_{h,k} = \dfrac{v_k}{\sum_1^D v_k} \end{cases} \tag{20}$$

where α_g is the best subscript in P_{h-1} in the previous generation. The pseudo-code of the adaptation of evolution matrix M is given in Algorithm 1.

C. ADAPTATION OF CONTROL PARAMETER

The proposed QUATRE-PM has two control parameters: the mutation factor F and the population size ps. Though the

Algorithm 1 Pseudo Code of Adaptation Scheme for Evolution Matrix M

if $S \neq \emptyset$ **then**
    update P_h by Eq.(18);
else
    update P_h by Eq.(20);
end if
randomly select P_h from historical memory;
P_h is generated by roulette into M_{init};
Transform M_{init} to M by row-element permutation;

TABLE 3. The mechanism of historical memory of *F*.

Index	1	2	...	h	...	H
μ_F	$\mu_{F,1}$	$\mu_{F,2}$	...	$\mu_{F,h}$	...	$\mu_{F,H}$

original QUATRE uses fixed values for these two control parameters, improvements are made for the adaptation of F in the subsequent QUATRE variants. QUATRE-EAR employed an adaptation scheme of F by incorporating prior knowledge into its current adaptaion. E-QUATRE introduced dimension improvement based weight in the adaptation of F, and it obtained nice outcomes. Different from the previous QUATRE variants, parameter F in the QUATRE-PM algorithm introduces a historical memory [44], which is also employed in the adaptation evolution matrix of our algorithm. The historical memory mechanism provides for better stability of the parameters in the iterations, and the specific structure of the historical memory mechanism is shown in Table 3, where the location parameter of the distributions of F obeys $u_{F,1} = u_{F,2} = \ldots = u_{F,h} = \ldots = u_{F,H} = 0.5$ at initialization and updated in the subsequent iterations according to Eq. (21):

$$\mu_{F,h} = \frac{\sum_{i \in S} w_i \cdot F_i^2}{\sum_{i \in S} w_i \cdot F_i} \tag{21}$$

where F_i denotes the F value of the i^{th} individual, and w_i denotes the weights which can be calculated by Eq. (19). In the iteration, each individual is randomly assigned a $u_{F,h}$ value selected from the historical memory pool and then its corresponding F value is generated by Cauchy distribution shown in Eq. (22):

$$F_i = randn(\mu_{F_h}, 0.1) \tag{22}$$

Different from the previous QUATRE variants employing fixed population size, the population size in our QUATRE-PM algorithm adopted the linear population size reduction [29], [45] in order to make a good balance between exploitation and exploration. The detailed reduction of ps is shown in Eq. (23):

$$ps_{g+1} = round(\frac{ps_{min} - ps_{ini}}{nfe_{max}} \cdot nfe + ps_{ini}) \tag{23}$$

where ps_{min} denotes the minimum population size, ps_{ini} denotes the initial population size, nfe_{max} denotes the maximum number of fitness evaluations, and nfe denotes the current number of fitness evaluations.

Algorithm 2 Pseudo Code of Perturbation Mechanism

$S_{min} = inf$
for i = 1; i < ps; i++ **do**
    if $count_i > n$ *and* $i! = gbest\_index$ **then**
        for $j = 1; j < length(DA); j++$ **do**
            $S = dis(X_{i,g}, DA_j)$
            if $S_{min} < distance$ **then**
                $S_{min} = distance$
                k = j
            end if
        end for
        $X_{i,g+1} = DA_k$
        Remove DA_k from DA.
    end if
end for

D. PERTURBATION MECHANISM

When an individual is stuck in a stagnant state [46], there is a possibility that it is stuck in a local optimum, then the individual needs to be perturbed to some extent in the hope that it will jump out of the local optimum. However, a random perturbation is challenging to achieve more suitable fitness of individuals and wastes computation resources to explore insignificantly search space. Therefore, this paper proposes a novel perturbation mechanism to tackle this problem by creating a discarded archive.

The discarded archive consists of the trial vector that failed in the selection operator, and when an individual of the population traps the stagnation state, it will select the most similarly discarded archive members to become it. Euclidean distance is employed to evaluate the similarity between individuals and members of the discarded archive, and the smaller the Euclidean distance, the higher the similarity with the member. The discarded archive size is half the current ps, and when the newly added individuals exceed the size, the member with the lowest fitness value will remove the discarded archive. Algorithm 2 describes the detailed operation process, where $count_i$ denotes the cumulative number of times the i^{th} individual in the population fails to obtain a better fitness, $gbest\_index$ denotes the subscript of the individual with the best fitness in the population, DA denotes the discarded archive, $length(DA)$ denotes the current size of the discarded archive, DA_j denotes the j^{th} member of the discarded archive, and n denotes a value of size $2 \cdot D$ and its minimum value is 40. Algorithm 3 gives the pseudo-code of the newly proposed QUATRE.

IV. EXPERIMENT ANALYSIS

It is a typical challenge to compare the performance of two algorithms. There are often two approaches to comparison. One approach compares the number of function evaluations that reach the minimum value, and the other is by comparing

Algorithm 3 Pseudo Code of the QUATRE-PM Algorithm

Initialize population by Eq. (1)
Initialize M_{init} and transform to M by Eq. (5).
Initialize nfe_{max} number of evaluations, and calculate the fitness value of the initial population, find the individual of best fitness *gbest*
$nfe = ps$, $g = 1$
while $nfe < nfe_{max}$ **do**
  Generate F_g by Eq. (22).
  Generate B_g by Eq. (16).
  Generate U_g by Eq. (4).
  for $i = 1; i \leq ps; i++$ **do**
    if $f(U_{i,g}) < f(X_{i,g})$ **then**
      if $f(U_{i,g}) < f(gbest)$ **then**
        $gbest = U_{i,g}$
      end if
      Insert $X_{i,g}$ into external archive population A
      $X_{i,g+1} = U_{i,g}$
    else
      Insert $U_{i,g}$ into discarded population DP
      $X_{i,g+1} = X_{i,g}$
    end if
  end for
  Execute Algorithm 1.
  Update historical memory of F by Eq. (21).
  Update population size ps by Eq. (23).
  Update A and DP.
  Execute Algorithm 2.
  $nfe = nfe + ps$
end while
Output the *gbest* and its value $f(gbest)$

the minimum value that can be reached at the same number of function evaluations. The Conference on Evolutionary Computation (CEC) test suite provides a set of benchmark functions for comparing algorithms and employs the second comparison approach. The proposed algorithm is verified under a total of 88 benchmark functions from CEC2013, CEC2014 and CEC2017. In the CEC 2013, the test functions are divided into three parts: $f_{a_1} - f_{a_5}$ are uni-modal functions, $f_{a_6} - f_{a_{20}}$ are basic multimodal functions and $f_{a_{21}} - f_{a_{28}}$ are composition functions. In CEC2014, the test functions are divided into four part: $f_{b_1} - f_{b_3}$ are uni-modal functions, $f_{b_4} - f_{b_{16}}$ are simple multimodal functions, $f_{b_{17}} - f_{b_{22}}$ are hybrid functions and $f_{b_{23}} - f_{b_{30}}$ are composition functions. In CEC2017, the test functions are divided into four part: $f_{c_1} - f_{c_2}$ are uni-modal functions, $f_{c_3} - f_{c_9}$ are simple multimodal functions, $f_{c_{10}} - f_{c_{19}}$ are hybrid functions and $f_{c_{20}} - f_{c_{30}}$ are composition functions. During testing, all functions are treated as a black box. The fitness error is $eps = f_i - f_i^*$ and is considered to be zero when the eps is less than $2.2204E - 016$ in our paper. The maximum number of function evaluation $nfes_{max}$ is set to $10000 \cdot D$. All experiments are conducted on a PC running Windows 10, equipped with an AMD R7-5800H CPU clocked at 3.20GHz, using MATLAB 2021b software.

A. PARAMETER SETTINGS

This paper compares the QUATRE-PM algorithm with ten other algorithms, including five DE and five QUATRE algorithms. DE variants are LSHADE [29], EDEV [23], jSO [47], LPalmDE [48] and MadDE [49]. QUATRE algorithms are original QUATRE [14] and QUATRE variants including C-QUATRE [36], S-QUATRE [37], QUATRE-EAR [43] and E-QUATRE [38]. In these algorithms, F denotes the scale factor parameter, Cr denotes the crossover rate, and ps denotes population size.

The parameters of DE variants are shown in Table 4. In LSHADE, F utilizes the Cauchy distribution with the location parameter of the distribution μ_F and a scale parameter σ_F fixed 0.1 to generate adaptive values, Cr utilizes the normal distribution with the mean of the distribution μ_{Cr} and standard deviation σ_{Cr} fixed 0.1 to generate adaptive values. The initial value of u_F and u_{Cr} is set to 0.5. H denotes the memory size of prior μ_{Cr} and μ_F, and $H = 5$. p denotes the percentage of elite individuals, and $p = 0.11$. The population size ps decreases by the strategy of linear population size reduction, the initial population size is $ps_{ini} = 18 \cdot D$ where D denotes the dimension number, and the minimum population size denotes $p_{min} = 4$. In EDEV, λ_1, λ_2, λ_3 denote the proportion of the subpopulation used by the three algorithms to the population, λ_4 denotes the proportion of reward subpopulation to population, ng denotes the number of iterations to reselect the new reward population, $\lambda_1 = \lambda_2 = \lambda_3 = 0.1$, $\lambda_4 = 0.7$, and $ng = 0.7$. The population size $ps = 100$. In jSO, F and Cr are generated similarly to LSAHDE. The initial value of u_F is 0.3, and u_{Cr} is set at 0.8, memory size $H = 5$, percentage of elite individuals p is a linearly decreasing value with a maximum value $p_{max} = 0.25$ and a minimum value $p_{min} = 0.125$, and archive population rate $arc = 1$. In LPalmDE, the F and Cr generation methods are the same as LSHADE, except that the population is divided into K groups, an independent adaptive F is generated in each group, and the scale parameter σ_F are set to 0.2. T_0 denotes the number of generations considered for archive populations, which is set to 70. The initial population size $ps_{ini} = 23 \cdot D$, the minimum population size $p_{min} = K$, and archive population rate $arc = 1.6$. In MadDE, F and Cr have generated the same way as LSHADE, and the initial values of $u_F = 0.3$ and $u_{Cr} = 0.3$ are different. p_{qBX} denotes the probability of choosing different crossover operators, set to 0.01. The initial population size $ps_{ini} = 2 \cdot D^2$, the minimum population size $p_{min} = 4$, archive population rate $arc = 1.6$ and memory size $H = 10$.

The specific parameters of QUATRE and variants are controlled in Table 5. QUATRE and C-QUATRE have the same $ps = 1000$ and the value of $F = 0.5$. In S-QUATRE, $ps = 100$ and the value of $F = 0.5$.

TABLE 4. Recommended parameter settings of all these contrasted DE.

Algorithms.	Parameters initial settings
LSHADE	$\mu_F = 0.5, \sigma_F = 0.1, \mu_{Cr} = 0.5, \sigma_{Cr} = 0.1, ps_{ini} = 18 \cdot D, ps_{min} = 4, arc = 2.6, p = 0.11, H = 6$
EDEV	$\lambda_1 = \lambda_2 = \lambda_3 = 0.1, \lambda_4 = 0.7, ng = 20, ps = 100$
jSO	$\mu_F = 0.3, \sigma_F = 0.1, \mu_{Cr} = 0.8, \sigma_{Cr} = 0.1, ps_{ini} = 25 \times \ln D \times \sqrt{D}, ps_{min} = 4, arc = 1, p_{max} = 0.25, p_{min} = 0.125, H = 5$
LPalmDE	$\mu_{F_j} = 0.5, \sigma_F = 0.2, \mu_{Cr} = 0.5, \sigma_{Cr} = 0.1, K = 8, ps_{ini} = 23 \times D, ps_{min} = K, p = 0.1, arc = 1.6, T_0 = 70$
MadDE	$\mu_F = 0.2, \sigma_F = 0.1, \mu_{Cr} = 0.2, \sigma_{Cr} = 0.1, p_{qBX} = 0.01, p = 0.18, arc = 2.3, ps_{ini} = 2 \times D^2, ps_{min} = 4, H = 10$

TABLE 5. Recommended parameter settings of all these contrasted QUATRE.

Algorithms.	Parameters initial settings
QUATRE	$ps = 1000, F = 0.7$
C-QUATRE	$ps = 1000, F = 0.7$
S-QUATRE	$ps = 100, F = 0.7$
QUATRE-EAR	$ps = 100, \mu_F = 0.5, \sigma_F = 0.2, T_0 = 70, p = 0.1, a = 1.6$
E-QUATRE	$ps = 100, \mu_F = 0.5, \sigma_F = 0.2, T_0 = 70, p_{max} = 0.2, p_{min} = 0.01, arc = 1.6$
new QUATRE	$\mu_F = 0.5, \sigma_F = 0.05, ps_{ini} = 25 \times \ln D \times \sqrt{D}, ps_{min} = 4, arc = 1.6, p = 0.11, H = 5, T_0 = 140$

TABLE 6. Summarize the results of the comparison between the new proposed QUATRE and the DE variants using Wilcoxon's signed rank test with the significant level $\alpha = 0.05$ under the CEC2013, CEC2014, and CEC2017 benchmark on 10D, 30D, and 50D.

QUATRE-PM compare with DE variants										
Test suit:	CEC2013			CEC2014			CEC2017			All
dimension:	D=10	D=30	D=50	D=10	D=30	D=50	D=10	D=30	D=50	$\sum$
result:	>/≈/<	>/≈/<	>/≈/<	>/≈/<	>/≈/<	>/≈/<	>/≈/<	>/≈/<	>/≈/<	>/≈/<
LSHADE	3/16/9	2/10/16	2/8/18	2/15/13	1/15/14	3/6/21	3/10/17	1/14/15	6/9/15	23/103/138
EDEV	2/10/16	4/4/20	4/6/18	4/5/21	7/4/19	8/2/20	4/8/18	4/7/19	7/0/23	44/46/174
jSO	3/13/12	1/8/19	6/6/16	2/16/12	2/13/15	7/8/15	3/15/12	5/11/14	6/8/16	35/98/131
LPamDE	4/17/7	2/9/17	3/8/17	4/17/9	3/8/19	3/8/19	4/19/7	1/9/20	0/8/22	24/103/137
MadDE	6/9/13	1/1/26	2/1/25	3/5/22	3/1/26	3/1/26	7/2/21	2/1/27	1/0/29	28/21/215

In QUATRE-EAR, the F generation methods are similar to LSHADE except that the scale parameter $\sigma_F = 0.2$. T_0 has the same meaning as LPalmDE and is set to 70. The percentage of elite individuals $p = 0.1$ and archive population rate $arc = 1.6$. In E-QUATRE, F is generated the same as LPalmDE except that the population is divided into two groups. p is a linearly decreasing value with a maximum value $p_{max} = 0.2$ and a minimum value $p_{min} = 0.01$. In the following comparison, the maximum number of function evaluations is set to $nfe_{max} = 10000 \cdot D$.

B. COMPARISON WITH DE VARIANTS

This section will compare QUATRE-PM with the LSHADE, EDEV, jSO, LPalmeDE, and MadDE. The comparison results are in Table 6. In Supplementary, the paper provides a comparison of Wilcoxon's signed rank test with the significant level $\alpha = 0.05$ of 51 independent runs on the three test suites 10D, 30D, and 50D of CEC2013 and CEC2014, where Table S1, Table S2, and Table S3 are CEC2013, Table S4, Table S5, and Table S6 are CEC2014.

The following compares with the DE variants under the CEC2017 test suite. Table 7 shows the 10D comparison in CEC2017, which has 30 benchmarks; the proposed algorithm outperforms LSHADE in 12 benchmarks, performs similarly in 15 benchmarks; outperforms DEDV in 18 benchmarks, performs similarly in 8 benchmarks; outperforms jSO in 12 benchmarks, and performs similarly in 15 benchmarks; outperforms LPalmDE in 7 benchmarks, and performs similarly in 19 benchmarks; outperforms MadDE in 21 benchmarks, and performs similarly in 2 benchmarks. Compared to other algorithms the new algorithm performs best in f_{c_5}, f_{c_7}, f_{c_8}, $f_{c_{10}}$, $f_{c_{12}}$, $f_{c_{13}}$, $f_{c_{15}}$, $f_{c_{18}}$, $f_{c_{19}}$ and $f_{c_{29}}$. Table 8 shows the 30D comparison in CEC2017, which has 30 benchmarks; the proposed algorithm outperforms LSHADE in 17 benchmarks, performs similarly in 10 benchmarks; outperforms DEDV in 19 benchmarks, performs similarly in 7 benchmarks; outperforms jSO in 14 benchmarks, and performs similarly in 11 benchmarks; outperforms LPalmDE in 20 benchmarks, and performs similarly in 9 benchmarks; outperforms MadDE in 27 benchmarks, and performs similarly in 1 benchmark. Compared to other algorithms the new algorithm performs best in $f_{c_1} - f_{c_3}$, f_{c_7}, $f_{c_{10}}$, $f_{c_{13}}$, $f_{c_{14}}$, $f_{c_{16}}$, $f_{c_{17}}$, $f_{c_{19}}$, $f_{c_{20}}$ and $f_{c_{23}}$. Table 9 shows the 50D comparison in CEC2017, which has 30 benchmarks; the proposed algorithm outperforms LSHADE in 15 benchmarks, performs similarly in 9 benchmarks; outperforms DEDV in 23 benchmarks and performs similarly in 0 benchmarks; outperforms jSO in 16 benchmarks, and performs similarly in 8 benchmarks; outperforms LPalmDE in 22 benchmarks, and performs similarly in 8 benchmarks; outperforms MadDE in 29 benchmarks, and performs similarly in 0 benchmarks. Compared to other algorithms the new algorithm performs best in f_{c_7}, f_{c_9}, $f_{c_{10}}$, $f_{c_{16}}$, $f_{c_{18}}$, $f_{c_{19}}$, $f_{c_{23}}$, $f_{c_{24}}$ and $f_{c_{29}}$.

Fig. 4 and Fig. 5 provide the convergence curves of the proposed algorithm and other excellent DE algorithms by employing the median value of 51 runs under 30D of CEC2017. From the comparison results, under f_{c_10}, $f_{c_{16}}$, $f_{c_{17}}$ and $f_{c_{20}}$, our proposed algorithms outperforms other state-of-the-art DE variants, and it can competitive comparison under $f_{c_1} - f_{c_3}$ f_{c_6}, f_{c_7}, f_{c_9}, $f_{c_{11}}$, $f_{c_{13}}$, $f_{c_{14}}$, $f_{c_{15}}$, $f_{c_{18}}$, $f_{c_{19}}$ and $f_{c_{22}} - f_{c_{25}}$, $f_{c_{27}} - f_{c_{29}}$ benchmarks.'

C. COMPARISON WITH QUATRE VARIANTS

In this section, our algorithm will be compared with the QUATRE, C-QUATRE, S-QUATRE, QUATRE-EAR, and

TABLE 7. Comparison results of the new proposed QUATRE algorithm with the DE variants under CEC2017 on 10D using Wilcoxon's signed rank test with the significant level $\alpha = 0.05$.

Function	Statistic	LSHADE	EDEV	jSO	LPalmDE	MadDE	proposed algorithm
f_{c_1}	Mean/Std	*0/0(≈)*	9.7525E-15/6.9647E-14(≈)	*0/0(≈)*	*0/0(≈)*	2.7586E-14/3.2828E-14(<)	*0/0*
	p-value	1.0000	1.0000	1.0000	1.0000	< 0.0001	-
f_{c_2}	Mean/Std	*0/0(≈)*	3.0233E-11/9.1274E-11(<)	*0/0(≈)*	*0/0(≈)*	6.1302E-15/1.4283E-14(<)	*0/0*
	p-value	1.0000	0.0071	1.0000	1.0000	0.0030	-
f_{c_3}	Mean/Std	*0/0(≈)*	*0/0(≈)*	*0/0(≈)*	*0/0(≈)*	*0/0(≈)*	*0/0*
	p-value	1.0000	1.0000	1.0000	1.0000	1.0000	-
f_{c_4}	Mean/Std	*0/0(≈)*	*0/0(≈)*	*0/0(≈)*	*0/0(≈)*	1.7833E-14/2.6638E-14(<)	*0/0*
	p-value	1.0000	1.0000	1.0000	1.0000	< 0.0001	-
f_{c_5}	Mean/Std	2.4604E+00/9.1943E-01(<)	3.9479E+00/9.9692E-01(<)	2.0109E+00/8.7690E-01(≈)	2.2630E+00/1.2754E+00(<)	3.8103E+00/1.0673E+00(<)	**1.7168E+00/9.1360E-01**
	p-value	< 0.0001	< 0.0001	0.1040	0.0064	< 0.0001	-
f_{c_6}	Mean/Std	**0/0**(>)	3.0094E-13/4.8544E-13(<)	2.2292E-15/1.5919E-14(≈)	1.1146E-14/3.4143E-14(≈)	3.3883E-13/2.9949E-13(<)	6.9663E-09/4.9749E-08
	p-value	0.0131	< 0.0001	0.0708	0.6078	< 0.0001	-
f_{c_7}	Mean/Std	1.2047E+01/8.7351E-01(≈)	1.5384E+01/1.3549E+00(<)	1.2148E+01/6.2201E-01(<)	1.2961E+01/9.9040E-01(<)	1.4523E+01/1.2058E+00(<)	**1.1813E+01/8.4634E-01**
	p-value	0.0757	< 0.0001	0.0024	< 0.0001	< 0.0001	-
f_{c_8}	Mean/Std	2.6942E+00/8.9615E-01(<)	4.7841E+00/1.0048E+00(<)	2.1070E+00/7.6157E-01(<)	2.5167E+00/1.1832E+00(<)	4.9998E+00/1.3414E+00(<)	**1.6583E+00/1.1020E+00**
	p-value	< 0.0001	< 0.0001	0.0044	0.0006	< 0.0001	-
f_{c_9}	Mean/Std	*0/0(≈)*	*0/0(≈)*	*0/0(≈)*	*0/0(≈)*	*0/0(≈)*	*0/0*
	p-value	1.0000	1.0000	1.0000	1.0000	1.0000	-
$f_{c_{10}}$	Mean/Std	1.8669E+01/3.4037E+01(<)	2.2405E+02/9.8878E+01(<)	4.8480E+01/6.4191E+01(<)	4.9683E+01/6.9334E+01(<)	1.0824E+02/6.7903E+01(<)	**1.3149E+01/2.7862E+01**
	p-value	0.0212	< 0.0001	0.0003	0.0021	< 0.0001	-
$f_{c_{11}}$	Mean/Std	3.4200E-01/6.5236E-01(<)	1.3882E+00/6.3556E-01(<)	*0/0(≈)*	*0/0(≈)*	1.4013E+00/6.3139E-01(<)	1.9509E-02/1.3932E-01
	p-value	0.0013	< 0.0001	0.3711	0.3711	< 0.0001	-
$f_{c_{12}}$	Mean/Std	2.9595E+01/5.3454E+01(<)	9.1235E+01/9.3873E+01(<)	2.6661E+00/1.6752E+01(≈)	2.3883E+01/4.7800E+01(<)	2.1816E+01/4.5668E+01(<)	**2.5304E-01/1.8295E-01**
	p-value	0.0003	< 0.0001	0.0566	0.0036	< 0.0001	-
$f_{c_{13}}$	Mean/Std	3.5954E+00/2.1990E+00(<)	4.7339E+00/2.2460E+00(<)	2.8850E+00/2.3811E+00(<)	2.0366E+00/2.4244E+00(≈)	2.9987E+00/2.2350E+00(<)	**1.4732E+00/2.2265E+00**
	p-value	< 0.0001	< 0.0001	0.0011	0.5263	0.0004	-
$f_{c_{14}}$	Mean/Std	1.7738E-01/3.5961E-01(≈)	7.2579E-01/5.1253E-01(<)	1.1705E-01/3.2376E-01(>)	**5.8527E-02/2.3644E-01**(>)	4.6007E-01/4.9522E-01(<)	3.3165E-01/5.5098E-01
	p-value	0.5300	0.0009	0.0089	0.0007	0.0031	-
$f_{c_{15}}$	Mean/Std	1.0936E-01/1.8020E-01(≈)	4.5275E-01/1.9276E-01(<)	1.9965E-01/2.1048E-01(<)	1.7563E-01/2.1653E-01(≈)	2.6930E-01/1.9398E-01(<)	**1.0568E-01/1.6549E-01**
	p-value	0.3367	< 0.0001	0.0064	0.1816	< 0.0001	-
$f_{c_{16}}$	Mean/Std	3.7910E-01/1.8093E-01(<)	4.6939E-01/3.0734E-01(<)	6.5752E-01/3.5313E-01(<)	**1.5791E-01/1.5791E-01**(>)	5.1773E-01/1.9527E-01(<)	2.4644E-01/2.3503E-01
	p-value	0.0016	< 0.0001	< 0.0001	0.0207	< 0.0001	-
$f_{c_{17}}$	Mean/Std	1.3277E-01/1.9747E-01(≈)	8.7667E-01/3.8703E-01(<)	5.1241E-01/4.1173E-01(<)	**1.1871E-01/1.8748E-01(≈)**	2.6778E-01/2.5506E-01(<)	1.6710E-01/3.2308E-01
	p-value	0.2432	< 0.0001	< 0.0001	0.7088	0.0013	-
$f_{c_{18}}$	Mean/Std	1.4621E-01/1.8444E-01(≈)	7.8188E-01/5.6433E-01(<)	2.7276E-01/2.0380E-01(<)	2.3676E-01/2.0789E-01(<)	2.9441E-01/2.1475E-01(<)	**1.4213E-01/1.8874E-01**
	p-value	0.9142	< 0.0001	0.0011	0.0177	0.0012	-
$f_{c_{19}}$	Mean/Std	1.1039E-02/1.0101E-02(<)	1.2512E-01/6.2550E-02(<)	1.3049E-02/1.4950E-02(<)	1.0869E-02/1.0599E-02(≈)	3.2924E-02/1.5419E-02(<)	**8.5774E-03/9.6472E-03**
	p-value	0.0166	< 0.0001	0.0199	0.3471	< 0.0001	-
$f_{c_{20}}$	Mean/Std	*0/0(≈)*	1.3241E-12/7.8106E-12(<)	4.5525E-01/2.6110E-01(<)	*0/0(≈)*	2.2292E-14/6.8286E-14(<)	*0/0*
	p-value	1.0000	0.0040	< 0.0001	1.0000	0.0184	-
$f_{c_{21}}$	Mean/Std	1.5106E+02/5.1860E+01(≈)	1.2279E+02/4.3895E+01(≈)	1.3843E+02/5.0376E+01(≈)	1.4473E+02/5.1874E+01(≈)	**9.8151E+01/1.4024E+01**(>)	1.4857E+02/5.2041E+01
	p-value	0.6704	0.2944	0.3388	0.8710	< 0.0001	-
$f_{c_{22}}$	Mean/Std	9.8987E+01/7.2350E+00(>)	9.6107E+01/1.9610E+01(>)	1.0000E+02/8.9223E-14(>)	1.0001E+02/4.8645E-02(>)	**8.8925E+01/2.2536E+01**(>)	9.8423E+01/1.2408E+01
	p-value	0.0003	0.0027	0.0005	0.0014	< 0.0001	-
$f_{c_{23}}$	Mean/Std	3.0303E+02/1.5027E+00(<)	3.0298E+02/2.1911E+00(<)	3.0117E+02/1.6653E+00(≈)	3.0088E+02/1.5085E+00(≈)	**2.8142E+02/8.2927E+01**(<)	3.0072E+02/1.3269E+00
	p-value	< 0.0001	< 0.0001	0.3528	0.5745	< 0.0001	-
$f_{c_{24}}$	Mean/Std	3.0851E+02/6.6492E+01(<)	1.9384E+02/1.1024E+02(>)	2.7488E+02/9.7991E+01(≈)	2.8526E+02/8.1444E+01(>)	**9.8039E+01/1.4003E+01**(>)	3.1773E+02/4.4844E+01
	p-value	0.0030	< 0.0001	0.5019	0.0004	< 0.0001	-
$f_{c_{25}}$	Mean/Std	4.1133E+02/2.0871E+01(≈)	4.1483E+02/2.2309E+01(≈)	4.0509E+02/1.6796E+01(≈)	4.1130E+02/2.0890E+01(≈)	**3.9775E+02/4.1010E-02**(>)	4.0949E+02/1.9996E+01
	p-value	0.2321	0.5824	0.4542	0.3530	< 0.0001	-
$f_{c_{26}}$	Mean/Std	3.0000E+02/0.0000E+00(≈)	3.0000E+02/0.0000E+00(≈)	3.0000E+02/0.0000E+00(≈)	3.0000E+02/0.0000E+00(≈)	**1.5490E+02/1.4875E+02**(>)	3.0000E+02/0.0000E+00
	p-value	1.0000	1.0000	1.0000	1.0000	< 0.0001	-
$f_{c_{27}}$	Mean/Std	3.8943E+02/1.8872E-01(>)	**3.7106E+02/2.2259E-01**(>)	3.8942E+02/2.0549E-01(>)	3.9342E+02/1.3036E+00(<)	3.8857E+02/6.9755E-01(>)	3.9111E+02/2.3910E+00
	p-value	< 0.0001	< 0.0001	0.0001	< 0.0001	< 0.0001	-
$f_{c_{28}}$	Mean/Std	3.2892E+02/8.8695E+01(≈)	3.3243E+02/6.7010E+01(≈)	3.3069E+02/8.7781E+01(≈)	3.1113E+02/5.5625E+01(≈)	**2.8235E+02/7.1291E+01**(>)	3.1669E+02/6.7428E+01
	p-value	0.5697	0.5179	0.6292	0.7656	0.0170	-
$f_{c_{29}}$	Mean/Std	2.3353E+02/2.7859E+00(<)	2.5196E+02/7.0551E+00(<)	2.3548E+02/3.1870E+00(<)	2.3239E+02/3.1955E+00(≈)	2.4906E+02/6.1553E+00(<)	**2.3111E+02/2.9069E+00**
	p-value	< 0.0001	< 0.0001	< 0.0001	0.0749	< 0.0001	-
$f_{c_{30}}$	Mean/Std	1.6425E+04/1.1443E+05(<)	**2.1562E+02/2.3234E+01**(>)	3.9451E+02/2.3304E-02(<)	3.9784E+02/1.1596E+01(≈)	9.3180E+02/1.3246E+03(<)	3.9639E+02/9.4409E+00
	p-value	0.0190	< 0.0001	0.0019	0.1612	< 0.0001	-
>/≈/<		3/15/12	4/8/18	3/15/12	4/19/7	7/2/21	-/-/-

E-QUATRE. The comparison results are given in Table 10. In Supplementary, the paper provides a comparison of Wilcoxon's signed rank test with the significant level $\alpha = 0.05$ of 51 independent runs under the three test suites including CEC2013, CEC2014, CEC2017 on 10D, 30D, and 50D respectively. Table S7, Table S8, and Table S9 are under CEC2013, Table S10, Table S11, and Table S12 are under CEC2014, and Table S13, Table S14 and Table S15 are under CEC2017.

The following is a specific comparison with the QUATRE and QUATRE variants in the CEC2017 test suite. For the 10D optimization in CEC2017, which has 30 benchmarks, the proposed algorithm outperforms QUATRE in 26 benchmarks, performs similarly in 3 benchmarks; outperforms C-QUATRE in 26 benchmarks, performs similarly in 4 benchmarks; outperforms S-QUATRE in 24 benchmarks, and performs similarly in 6 benchmarks; outperforms QUATRE-EAR in 14 benchmarks, and performs similarly in

TABLE 8. Comparison results of the new proposed QUATRE algorithm with the DE variants under CEC2017 on 30D using Wilcoxon's signed rank test with the significant level $\alpha = 0.05$.

Function	Statistic	LSHADE	EDEV	jSO	LPalmDE	MadDE	proposed algorithm
f_{c_1}	Mean/Std	5.5729E-16/2.7859E-15(≈)	2.7864E-16/1.9899E-15(≈)	3.3437E-15/6.0880E-15(<)	8.3593E-16/3.3770E-15(≈)	2.0085E+03/4.4686E+02(<)	**0/0**
	p-value	0.3458	1.0000	0.0003	0.1489	< 0.0001	-
f_{c_2}	Mean/Std	6.1302E-15/1.1806E-14(<)	3.8957E+03/2.7821E+04(<)	1.6719E-15/6.7540E-15(≈)	7.8020E-15/1.4014E-14(<)	5.2305E+14/6.9127E+14(<)	**5.5729E-16/4.0194E-15**
	p-value	0.0010	< 0.0001	0.4237	0.0009	< 0.0001	-
f_{c_3}	Mean/Std	4.4583E-15/1.5434E-14(≈)	1.6340E-02/5.3210E-02(<)	4.4583E-14/2.3612E-14(<)	1.3375E-14/2.4352E-14(<)	3.0987E+04/1.0225E+04(<)	**0/0**
	p-value	0.0719	< 0.0001	< 0.0001	0.0003	< 0.0001	-
f_{c_4}	Mean/Std	5.8562E+01/2.6567E-14(≈)	**1.7980E+00/2.0035E+00**(>)	5.8562E+01/2.6567E-14(≈)	5.0512E+01/2.1683E+01(≈)	8.9962E+01/1.4548E+01(<)	5.5461E+01/1.3990E+01
	p-value	0.3964	< 0.0001	0.3964	0.4578	< 0.0001	-
f_{c_5}	Mean/Std	**6.4251E+00/1.4713E+00**(>)	3.2835E+01/5.7464E+00(<)	8.1949E+00/1.9837E+00(≈)	1.1374E+01/3.1792E+00(<)	7.8362E+01/9.3459E+00(<)	8.0182E+00/2.4807E+00
	p-value	0.0004	< 0.0001	0.4533	< 0.0001	< 0.0001	-
f_{c_6}	Mean/Std	8.0917E-09/3.2689E-08(>)	**1.1369E-13/0.0000E+00**(>)	4.0892E-09/2.0117E-08(>)	2.6975E-09/1.9164E-08(>)	1.0636E-01/3.3839E-02(<)	1.5493E-07/4.3189E-07
	p-value	0.0005	< 0.0001	0.0002	0.0002	< 0.0001	-
f_{c_7}	Mean/Std	3.7632E+01/1.6532E+00(≈)	6.2959E+01/5.7342E+00(<)	3.8310E+01/1.4798E+00(<)	4.1712E+01/3.1380E+00(<)	1.0747E+02/1.2213E+01(<)	**3.7482E+01/2.1931E+00**
	p-value	0.6766	< 0.0001	0.0110	< 0.0001	< 0.0001	-
f_{c_8}	Mean/Std	**7.1352E+00/1.7752E+00(≈)**	3.4231E+01/6.1539E+00(<)	8.7458E+00/1.7379E+00(<)	1.1989E+01/3.7513E+00(<)	7.2267E+01/8.3518E+00(<)	7.4329E+00/2.5666E+00
	p-value	0.5392	< 0.0001	0.0003	< 0.0001	< 0.0001	-
f_{c_9}	Mean/Std	*0/0(≈)*	7.0218E-03/2.4309E-02(≈)	*0/0(≈)*	*0/0(≈)*	2.1889E+01/1.4430E+01(<)	*0/0*
	p-value	1.0000	0.0719	1.0000	1.0000	< 0.0001	-
$f_{c_{10}}$	Mean/Std	1.4243E+03/2.3434E+02(<)	2.8804E+03/6.5971E+02(<)	1.6182E+03/2.2835E+02(<)	1.7223E+03/2.5973E+02(<)	2.8533E+03/3.3394E+02(<)	**1.2208E+03/2.8618E+02**
	p-value	0.0004	< 0.0001	< 0.0001	< 0.0001	< 0.0001	-
$f_{c_{11}}$	Mean/Std	2.0243E+01/2.5338E+01(<)	2.3604E+01/2.2513E+01(<)	**5.9525E+00/1.4002E+01(≈)**	1.1354E+01/1.9389E+01(≈)	7.4655E+01/1.5860E+01(<)	6.7473E+00/1.3995E+01
	p-value	< 0.0001	< 0.0001	0.1268	0.0907	< 0.0001	-
$f_{c_{12}}$	Mean/Std	1.0671E+03/3.6860E+02(<)	2.0670E+03/2.2431E+03(<)	**1.7497E+02/1.0603E+02**(>)	1.0803E+03/3.9680E+02(<)	3.4809E+05/1.3720E+05(<)	4.1582E+02/2.6752E+02
	p-value	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	-
$f_{c_{13}}$	Mean/Std	1.4986E+01/6.9365E+00(≈)	7.1825E+01/6.8903E+01(<)	1.5174E+01/5.8773E+00(≈)	1.5340E+01/6.3798E+00(≈)	1.3548E+04/4.2719E+03(<)	**1.3200E+01/7.3174E+00**
	p-value	0.1424	< 0.0001	0.0907	0.0962	< 0.0001	-
$f_{c_{14}}$	Mean/Std	2.1732E+01/1.2093E+00(<)	2.9746E+01/8.7092E+00(<)	2.1991E+01/1.2496E+00(<)	2.0345E+01/6.9135E+00(<)	8.4096E+01/1.6627E+01(<)	**1.5207E+01/8.9286E+00**
	p-value	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	-
$f_{c_{15}}$	Mean/Std	3.0185E+00/1.8875E+00(≈)	1.4858E+01/1.2034E+01(<)	**1.4980E+00/1.0804E+00**(>)	4.1637E+00/1.5854E+00(<)	3.0525E+02/2.2383E+02(<)	2.5609E+00/1.7714E+00
	p-value	0.1756	< 0.0001	0.0013	< 0.0001	< 0.0001	-
$f_{c_{16}}$	Mean/Std	5.5161E+01/5.8107E+01(<)	4.1379E+02/1.3570E+02(<)	6.5177E+01/7.7876E+01(<)	1.5570E+02/1.3047E+02(<)	4.6894E+02/1.3235E+02(<)	**1.3793E+01/2.0795E+01**
	p-value	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	-
$f_{c_{17}}$	Mean/Std	3.2577E+01/5.5521E+00(<)	5.7126E+01/1.1899E+01(<)	3.4035E+01/7.8457E+00(<)	3.1256E+01/1.0701E+01(<)	7.4468E+01/1.4736E+01(<)	**1.4415E+01/6.4812E+00**
	p-value	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	-
$f_{c_{18}}$	Mean/Std	2.1879E+01/1.0440E+00(<)	6.2441E+01/8.5356E+01(<)	**2.0358E+01/2.8610E+00(≈)**	2.1619E+01/2.7713E+00(<)	6.3992E+04/2.8374E+04(<)	2.0733E+01/2.9950E+00
	p-value	0.0002	< 0.0001	0.6816	0.0002	< 0.0001	-
$f_{c_{19}}$	Mean/Std	5.1093E+00/1.2062E+00(<)	1.4049E+01/3.3475E+00(<)	4.4866E+00/1.6637E+00(≈)	4.9045E+00/1.6382E+00(<)	2.3875E+02/3.3859E+02(<)	**4.0348E+00/1.3927E+00**
	p-value	0.0002	< 0.0001	0.1697	0.0043	< 0.0001	-
$f_{c_{20}}$	Mean/Std	3.1034E+01/5.8775E+00(<)	9.3590E+01/6.0212E+01(<)	2.8692E+01/7.2874E+00(<)	3.4746E+01/3.0883E+01(<)	9.9810E+01/4.8429E+01(<)	**2.0380E+01/2.9433E+01**
	p-value	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	-
$f_{c_{21}}$	Mean/Std	2.0745E+02/1.3250E+00(<)	2.3340E+02/6.5960E+00(<)	2.0919E+02/2.0707E+00(<)	2.1065E+02/2.3352E+00(<)	**1.9301E+02/6.7925E+01**(>)	2.0657E+02/2.8168E+00
	p-value	0.0136	< 0.0001	< 0.0001	< 0.0001	0.0036	-
$f_{c_{22}}$	Mean/Std	*1.0000E+02/1.4352E-14(≈)*	*1.0000E+02/1.0808E-13(≈)*	*1.0000E+02/1.4352E-14(≈)*	*1.0000E+02/1.4352E-14(≈)*	*1.0000E+02/1.7163E-04(≈)*	*1.0000E+02/6.4311E-14*
	p-value	1.0000	1.0000	1.0000	1.0000	1.0000	-
$f_{c_{23}}$	Mean/Std	3.4961E+02/2.5383E+00(≈)	3.7722E+02/1.0769E+01(<)	3.5044E+02/2.6881E+00(<)	3.4910E+02/5.2452E+00(≈)	4.1390E+02/1.0748E+01(<)	**3.4821E+02/4.7784E+00**
	p-value	0.0662	< 0.0001	0.0030	0.3989	< 0.0001	-
$f_{c_{24}}$	Mean/Std	4.2572E+02/1.3491E+00(<)	4.3837E+02/8.7285E+00(<)	4.2687E+02/2.0969E+00(<)	**4.2329E+02/3.6098E+00(≈)**	4.8644E+02/9.3642E+00(<)	4.2346E+02/2.6842E+00
	p-value	< 0.0001	< 0.0001	< 0.0001	0.9664	< 0.0001	-
$f_{c_{25}}$	Mean/Std	3.8674E+02/2.1391E-02(<)	**3.7828E+02/6.5877E-01**(>)	3.8670E+02/7.7419E-03(>)	3.8678E+02/3.0799E-02(<)	3.8674E+02/8.4198E-01(<)	3.8672E+02/1.9649E-02
	p-value	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	-
$f_{c_{26}}$	Mean/Std	9.2449E+02/3.6025E+01(<)	9.4298E+02/4.5208E+02(≈)	9.3338E+02/4.3359E+01(<)	9.2393E+02/5.6049E+01(<)	**2.6863E+02/4.6863E+01**(>)	8.5719E+02/4.2467E+01
	p-value	< 0.0001	0.8993	< 0.0001	< 0.0001	< 0.0001	-
$f_{c_{27}}$	Mean/Std	5.0404E+02/4.9066E+00(<)	5.0001E+02/1.7345E-04(≈)	**4.9846E+02/5.8107E+00(≈)**	5.0814E+02/4.1166E+00(<)	5.1379E+02/3.6275E+00(<)	4.9923E+02/7.3979E+00
	p-value	0.0004	0.5580	0.5835	< 0.0001	< 0.0001	-
$f_{c_{28}}$	Mean/Std	3.4334E+02/5.7768E+01(<)	3.4253E+02/5.1338E+01(≈)	**3.1522E+02/3.8592E+01(≈)**	3.3071E+02/5.1236E+01(≈)	3.9812E+02/3.5030E+00(<)	3.2353E+02/4.5395E+01
	p-value	0.0205	0.2163	0.3826	0.5648	< 0.0001	-
$f_{c_{29}}$	Mean/Std	4.3239E+02/7.1840E+00(<)	**4.1548E+02/5.0326E+01(≈)**	4.3350E+02/1.5383E+01(<)	4.2997E+02/1.4194E+01(<)	5.3306E+02/2.3253E+01(<)	4.1858E+02/8.3261E+00
	p-value	< 0.0001	0.3707	< 0.0001	< 0.0001	< 0.0001	-
$f_{c_{30}}$	Mean/Std	1.9741E+03/4.3820E+01(>)	**4.2405E+02/6.3793E+02**(>)	1.9703E+03/1.2544E+01(>)	2.0594E+03/6.6290E+01(<)	1.2277E+04/3.7167E+03(<)	2.0038E+03/3.8431E+01
	p-value	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	-
>/≈/<		3/10/17	4/7/19	5/11/14	1/9/20	2/1/27	-/-/-

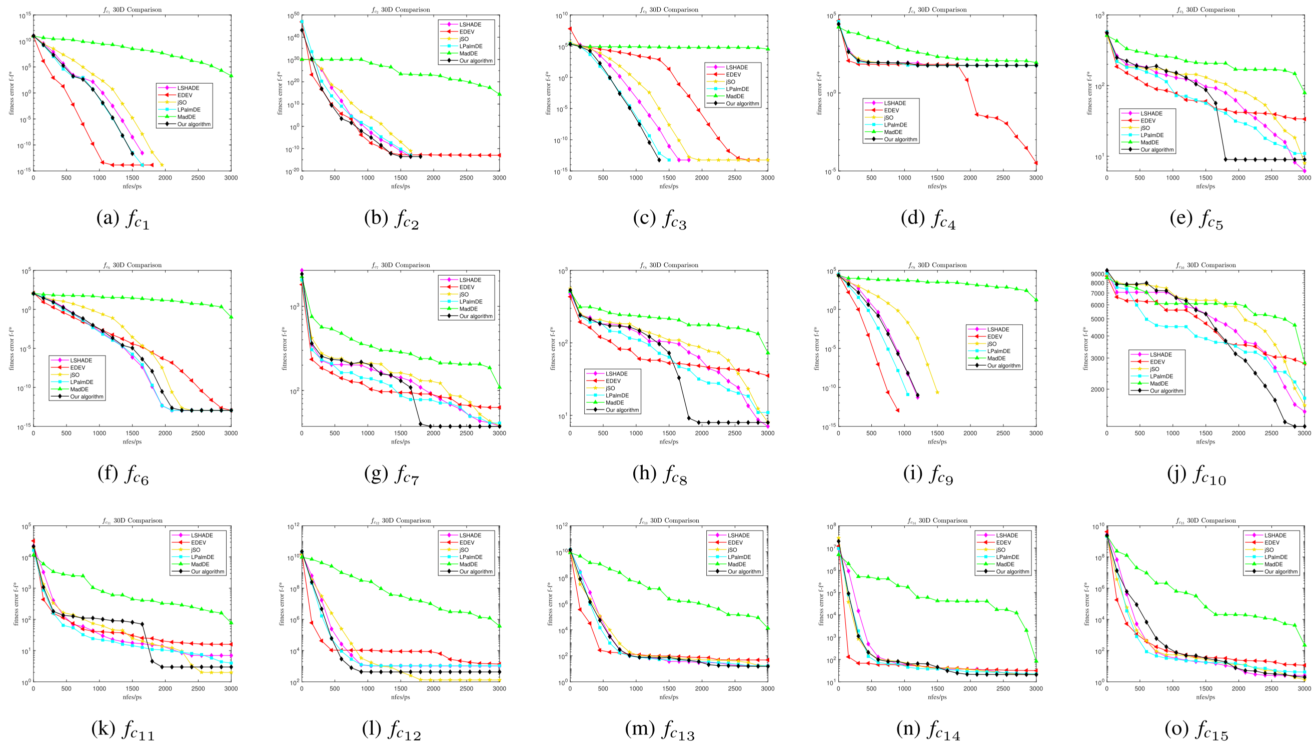
14 benchmarks; outperforms E-QUATRE in 12 benchmarks, and performs similarly in 14 benchmarks. Compared to other algorithms the new algorithm performs best in f_{c_5}, f_{c_7}, f_{c_8}, $f_{c_{10}}$, $f_{c_{12}}$, $f_{c_{15}}$, $f_{c_{16}}$, $f_{c_{19}}$, $f_{c_{20}}$, $f_{c_{29}}$ and $f_{c_{30}}$. For the 30D comparison in CEC2017, which has 30 benchmarks, the proposed algorithm outperforms QUATRE in 29 benchmarks, performs similarly in 1 benchmark; outperforms C-QUATRE in 30 benchmarks, performs similarly in 0 benchmarks; outperforms S-QUATRE in 28 benchmarks, and performs similarly in 1 benchmarks; outperforms QUATRE-EAR in 22 benchmarks, and performs similarly in 7 benchmarks; outperforms E-QUATRE in 24 benchmarks, and performs similarly in 5 benchmarks. Compared to other algorithms the new algorithm performs best in $f_{c_1} - f_{c_3}$, f_{c_5}, f_{c_7}, f_{c_8}, $f_{c_{10}} - f_{c_{13}}$, $f_{c_{14}} - f_{c_{21}}$, $f_{c_{23}} - f_{c_{26}}$ and $f_{c_{28}} - f_{c_{30}}$. For the 50D comparison in CEC2017, which has 30 benchmarks, the proposed algorithm

TABLE 9. Comparison results of the new proposed QUATRE algorithm with the DE variants under CEC2017 on 50D using Wilcoxon's signed rank test with the significant level $\alpha = 0.05$.

Function	Statistic	LSHADE	EDEV	jSO	LPalmDE	MadDE	proposed algorithm
f_{c_1}	Mean/Std	1.8112E-14/7.5617E-15(≈)	**1.0031E-14/2.9849E-14**(>)	2.6750E-14/8.8277E-15(<)	2.2570E-14/7.6139E-15(≈)	1.2010E+04/5.4914E+03(<)	6.9187E-13/3.6840E-12
	p-value	0.8380	0.0001	0.0139	0.1962	< 0.0001	-
f_{c_2}	Mean/Std	**3.6224E-14/2.6098E-14**(≈)	1.6218E+02/1.1582E+03(<)	7.8578E-14/1.7424E-13(≈)	4.6812E-14/4.7157E-14(≈)	1.0000E+30/1.4214E+14(<)	4.3078E-13/2.5958E-12
	p-value	0.7942	< 0.0001	0.2180	0.4812	< 0.0001	-
f_{c_3}	Mean/Std	1.6050E-13/4.7838E-14(<)	9.2392E-01/2.1488E+00(<)	2.7196E-13/7.0370E-14(<)	**1.4935E-13/4.8181E-14**(<)	1.2439E+05/1.0812E+04(<)	9.4465E-07/6.7459E-06
	p-value	0.0012	< 0.0001	< 0.0001	0.0101	< 0.0001	-
f_{c_4}	Mean/Std	9.2928E+01/4.9860E+01(≈)	**2.1414E+01/3.6780E+01**(>)	4.3007E+01/3.7709E+01(>)	7.2106E+01/4.8632E+01(≈)	1.1569E+02/2.3266E+01(<)	7.9236E+01/4.4661E+01
	p-value	0.1460	< 0.0001	0.0001	0.3564	< 0.0001	-
f_{c_5}	Mean/Std	**1.1664E+01/2.3028E+00**(>)	6.5451E+01/1.0843E+01(<)	1.6649E+01/2.7240E+00(<)	2.2610E+01/4.1939E+00(<)	3.1145E+02/1.6820E+01(<)	1.3878E+01/4.1592E+00
	p-value	0.0012	< 0.0001	0.0001	< 0.0001	< 0.0001	-
f_{c_6}	Mean/Std	2.1708E-04/9.2252E-04(>)	**1.1369E-13/0.0000E+00**(>)	6.2760E-07/7.8642E-07(≈)	4.8645E-04/1.6313E-03(≈)	2.0821E+00/3.6064E-01(<)	7.8105E-05/3.5826E-04
	p-value	0.0236	< 0.0001	0.0587	0.1126	< 0.0001	-
f_{c_7}	Mean/Std	6.2976E+01/1.9344E+00(≈)	1.1521E+02/1.4143E+01(<)	6.6275E+01/2.7924E+00(<)	7.0366E+01/4.7221E+00(<)	3.5313E+02/2.3333E+01(<)	**6.2864E+01/4.4870E+00**
	p-value	0.2960	< 0.0001	0.0002	< 0.0001	< 0.0001	-
f_{c_8}	Mean/Std	**1.1872E+01/1.7966E+00**(>)	6.1719E+01/1.1080E+01(<)	1.6408E+01/3.0320E+00(<)	2.4447E+01/4.6136E+00(<)	3.0698E+02/1.7224E+01(<)	1.3726E+01/4.1393E+00
	p-value	0.0048	< 0.0001	0.0010	< 0.0001	< 0.0001	-
f_{c_9}	Mean/Std	5.7958E-14/5.7398E-14(<)	1.5601E+00/1.6821E+00(<)	8.0250E-14/5.2316E-14(<)	7.1333E-14/5.5513E-14(<)	2.7474E+03/7.9795E+02(<)	**8.9166E-15/3.2155E-14**
	p-value	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	-
$f_{c_{10}}$	Mean/Std	3.2167E+03/2.6700E+02(<)	5.4851E+03/1.6699E+03(<)	3.1675E+03/3.8920E+02(<)	3.3116E+03/4.4944E+02(<)	8.1868E+03/4.9820E+02(<)	**2.5326E+03/4.0232E+02**
	p-value	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	-
$f_{c_{11}}$	Mean/Std	4.6295E+01/6.5772E+00(<)	6.6511E+01/2.7473E+01(<)	**2.8621E+01/3.5603E+00**(≈)	6.7866E+01/1.2660E+01(<)	3.4087E+02/2.6447E+01(<)	3.0037E+01/4.3051E+00
	p-value	< 0.0001	< 0.0001	0.1289	< 0.0001	< 0.0001	-
$f_{c_{12}}$	Mean/Std	2.1230E+03/4.3131E+02(>)	8.9506E+03/2.9886E+04(<)	**1.7929E+03/4.4058E+02**(>)	2.2459E+03/6.3405E+02(≈)	3.2127E+06/4.7400E+05(<)	2.3902E+03/8.3050E+02
	p-value	0.0194	< 0.0001	< 0.0001	0.4997	< 0.0001	-
$f_{c_{13}}$	Mean/Std	5.5352E+01/3.2840E+01(≈)	5.1501E+02/8.8794E+02(<)	**3.4099E+01/2.1570E+01**(>)	7.6148E+01/4.1356E+01(<)	2.4848E+04/7.9340E+03(<)	5.7140E+01/3.8853E+01
	p-value	0.9813	< 0.0001	0.0013	0.0212	< 0.0001	-
$f_{c_{14}}$	Mean/Std	2.9893E+01/2.7436E+00(<)	1.9622E+02/5.8799E+02(<)	**2.4627E+01/1.6467E+00**(≈)	3.2574E+01/4.6792E+00(<)	1.0214E+05/4.0551E+04(<)	2.5025E+01/2.0785E+00
	p-value	< 0.0001	< 0.0001	0.4505	< 0.0001	< 0.0001	-
$f_{c_{15}}$	Mean/Std	3.9407E+01/1.0237E+01(<)	1.7324E+02/1.6832E+02(<)	**2.3345E+01/2.9664E+00**(≈)	4.6521E+01/1.5713E+01(<)	1.6001E+04/2.3481E+03(<)	2.3693E+01/3.7476E+00
	p-value	< 0.0001	< 0.0001	0.8623	< 0.0001	< 0.0001	-
$f_{c_{16}}$	Mean/Std	3.7923E+02/1.1373E+02(<)	6.8253E+02/1.7365E+02(<)	4.5596E+02/1.3669E+02(<)	4.0017E+02/1.2056E+02(<)	1.0810E+03/1.7065E+02(<)	**1.6785E+02/8.5223E+01**
	p-value	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	-
$f_{c_{17}}$	Mean/Std	**2.4604E+02/6.2787E+01**(>)	5.1503E+02/1.4184E+02(<)	2.8190E+02/9.9812E+01(≈)	3.0477E+02/1.0182E+02(≈)	7.7755E+02/1.3328E+02(<)	3.0139E+02/1.3641E+02
	p-value	0.0133	< 0.0001	0.3657	0.8293	< 0.0001	-
$f_{c_{18}}$	Mean/Std	3.7264E+01/8.8759E+00(<)	1.4972E+03/3.9546E+03(<)	2.4637E+01/1.5882E+00(≈)	4.6730E+01/1.7570E+01(<)	8.0578E+05/2.4935E+05(<)	**2.4598E+01/2.8788E+00**
	p-value	< 0.0001	< 0.0001	0.5706	< 0.0001	< 0.0001	-
$f_{c_{19}}$	Mean/Std	2.4205E+01/5.3957E+00(<)	6.8418E+01/4.7653E+01(<)	1.3709E+01/2.6850E+00(<)	3.2151E+01/1.2657E+01(<)	1.6661E+04/1.5324E+03(<)	**1.1448E+01/2.6324E+00**
	p-value	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	-
$f_{c_{20}}$	Mean/Std	1.6075E+02/6.7797E+01(≈)	2.6008E+02/1.0866E+02(<)	**1.5062E+02/8.0352E+01**(>)	1.6478E+02/1.1185E+02(≈)	5.6369E+02/1.1690E+02(<)	2.0321E+02/1.2964E+02
	p-value	0.0589	0.0118	0.0217	0.2284	< 0.0001	-
$f_{c_{21}}$	Mean/Std	**2.1253E+02/2.6106E+00**(>)	2.6168E+02/1.2958E+01(<)	2.1743E+02/3.2060E+00(<)	2.2455E+02/3.9987E+00(<)	4.7043E+02/1.4674E+01(<)	2.1590E+02/5.2644E+00
	p-value	0.0005	< 0.0001	0.0066	< 0.0001	< 0.0001	-
$f_{c_{22}}$	Mean/Std	2.6471E+03/1.5713E+03(<)	1.1702E+03/2.6219E+03(<)	1.8033E+03/1.8354E+03(<)	4.6026E+02/1.1205E+03(≈)	**1.4360E+02/2.8226E+01**(<)	3.7666E+02/9.5905E+02
	p-value	< 0.0001	0.0127	< 0.0001	0.3627	< 0.0001	-
$f_{c_{23}}$	Mean/Std	4.3038E+02/4.6320E+00(≈)	4.8966E+02/2.1907E+01(<)	4.3227E+02/5.6971E+00(<)	4.3656E+02/5.6615E+00(<)	7.0828E+02/2.0503E+01(<)	**4.2810E+02/1.0754E+01**
	p-value	0.3367	< 0.0001	0.0130	< 0.0001	< 0.0001	-
$f_{c_{24}}$	Mean/Std	5.0595E+02/2.6792E+00(<)	5.4965E+02/1.4720E+01(<)	5.0912E+02/4.1401E+00(<)	5.0606E+02/6.4201E+00(<)	7.6715E+02/1.7173E+01(<)	**5.0089E+02/6.4159E+00**
	p-value	< 0.0001	< 0.0001	< 0.0001	0.0001	< 0.0001	-
$f_{c_{25}}$	Mean/Std	4.8199E+02/4.3794E+00(≈)	**4.4347E+02/1.6449E+01**(>)	4.7997E+02/3.3259E+00(>)	5.0483E+02/3.3899E+01(<)	6.0814E+02/1.1132E+00(<)	4.8289E+02/8.1235E+00
	p-value	0.7462	< 0.0001	< 0.0001	0.0001	< 0.0001	-
$f_{c_{26}}$	Mean/Std	1.1527E+03/4.8444E+01(<)	1.6373E+03/3.0286E+02(<)	1.1578E+03/5.5864E+01(<)	1.1628E+03/8.7305E+01(<)	**3.0636E+02/1.3379E+00**(>)	1.0114E+03/8.2822E+01
	p-value	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	-
$f_{c_{27}}$	Mean/Std	5.3270E+02/1.1833E+01(<)	**5.0001E+02/2.1917E-04**(>)	5.1448E+02/1.4238E+01(>)	5.3956E+02/8.6241E+00(<)	7.1222E+02/2.4731E+01(<)	5.2402E+02/9.5900E+00
	p-value	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	-
$f_{c_{28}}$	Mean/Std	4.6731E+02/1.8486E+01(≈)	**4.5502E+02/1.4554E+01**(>)	4.5981E+02/6.8398E+00(≈)	5.0291E+02/1.4670E+01(<)	5.5553E+02/8.9258E+00(<)	4.6460E+02/1.5894E+01
	p-value	0.5640	< 0.0001	0.0726	< 0.0001	< 0.0001	-
$f_{c_{29}}$	Mean/Std	3.5285E+02/1.3190E+01(<)	3.9828E+02/1.0202E+02(<)	3.6264E+02/1.4407E+01(<)	3.5452E+02/2.0916E+01(<)	1.1263E+03/1.1904E+02(<)	**3.4740E+02/2.6264E+01**
	p-value	0.0171	0.0013	0.0001	0.0114	< 0.0001	-
$f_{c_{30}}$	Mean/Std	6.6468E+05/6.8089E+04(<)	**6.5014E+02/8.7964E+02**(>)	6.2262E+05/4.7686E+04(<)	6.2786E+05/3.6103E+04(<)	3.9943E+06/3.4649E+05(<)	6.0120E+05/3.1632E+04
	p-value	< 0.0001	< 0.0001	0.0066	< 0.0001	< 0.0001	-
>/≈/<		6/9/15	7/0/23	6/8/16	0/8/22	1/0/29	-/-/-

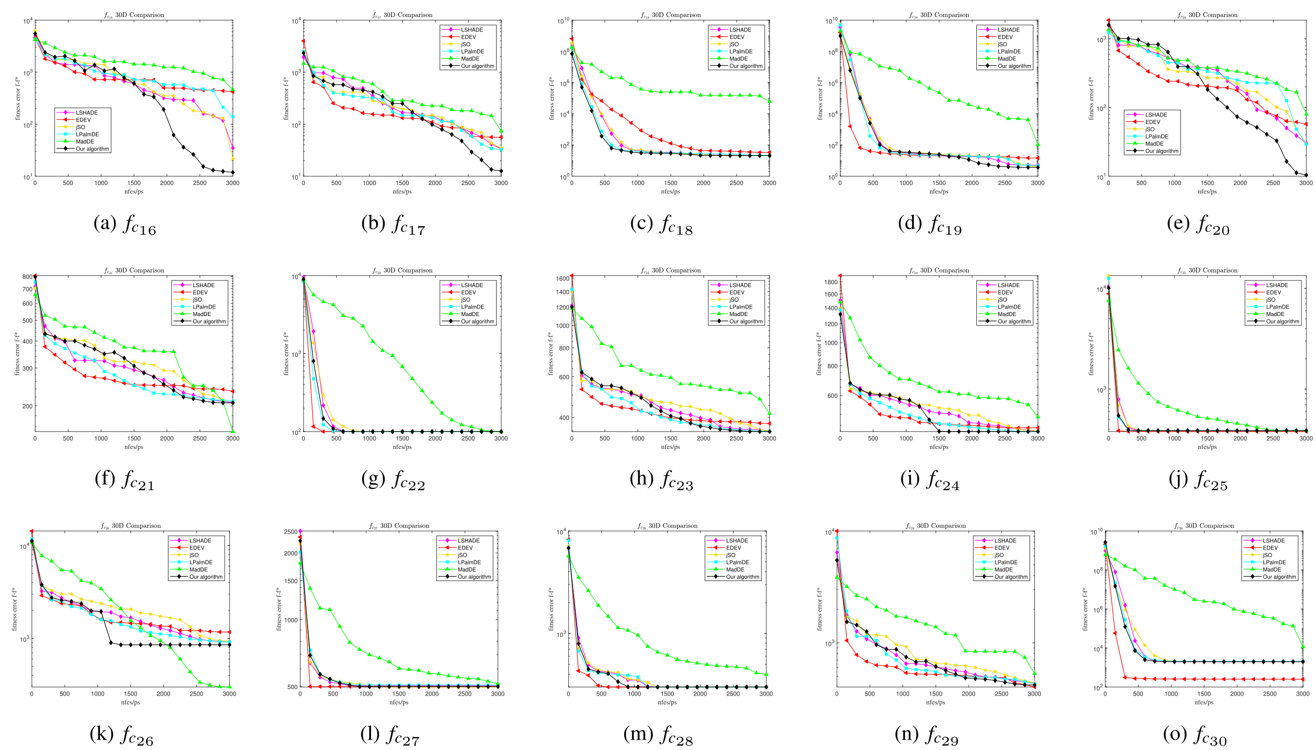
outperforms QUATRE in 30 benchmarks, and performs similarly in 0 benchmarks; outperforms C-QUATRE in 30 benchmarks, and performs similarly in 0 benchmarks; outperforms S-QUATRE in 29 benchmarks, and performs similarly in 1 benchmarks; outperforms QUATRE-EAR in 28 benchmarks, and performs similarly in 0 benchmarks; outperforms E-QUATRE in 27 benchmarks, and performs similarly in 1 benchmark. Compared to other algorithms the new algorithm performs best in f_{c_2}, f_{c_5} and $f_{c_7} - f_{c_{30}}$.

Fig.6 and Fig.7 provide the convergence curves of the new algorithm and other excellent QUATRE algorithms by employing the median value of 51 runs under 30D of CEC2017. From the comparison results, under $f_{c_2}, f_{c_3}, f_{c_5}, f_{c_7}, f_{c_8}, f_{c_10} - f_{c_{12}}, f_{c_{15}} - f_{c_{17}}, f_{c_{19}} - f_{c_{21}}, f_{c_{23}}, f_{c_{24}}, f_{c_{26}}$ and



(a) fc_1 (b) fc_2 (c) fc_3 (d) fc_4 (e) fc_5
(f) fc_6 (g) fc_7 (h) fc_8 (i) fc_9 (j) fc_{10}
(k) fc_{11} (l) fc_{12} (m) fc_{13} (n) fc_{14} (o) fc_{15}

FIGURE 4. Convergence speed comparison by employing the median value of 51 runs obtained by each algorithm on CEC2017 30-D optimization. There are total 30 comparison figures and the first 15 figures are presented here.



(a) fc_{16} (b) fc_{17} (c) fc_{18} (d) fc_{19} (e) fc_{20}
(f) fc_{21} (g) fc_{22} (h) fc_{23} (i) fc_{24} (j) fc_{25}
(k) fc_{26} (l) fc_{27} (m) fc_{28} (n) fc_{29} (o) fc_{30}

FIGURE 5. Convergence speed comparison by employing the median value of 51 runs obtained by each algorithm on CEC2017 30-D optimization. There are total 30 comparison figures and the last 15 figures are presented here.

fc_{29} benchmarks, the proposed algorithm outperforms other QUATRE variants, and it has competitive performance under $fc_1, fc_6, fc_9, fc_{18}, fc_{22}, fc_{25}, fc_{27}, fc_{28}$ and fc_{30} benchmarks.

D. ANALYSIS OF EVOLUTION MATRIX M

In the QUATRE algorithm, there are two schemes to initialize the evolution matrix M. The first scheme is to set the

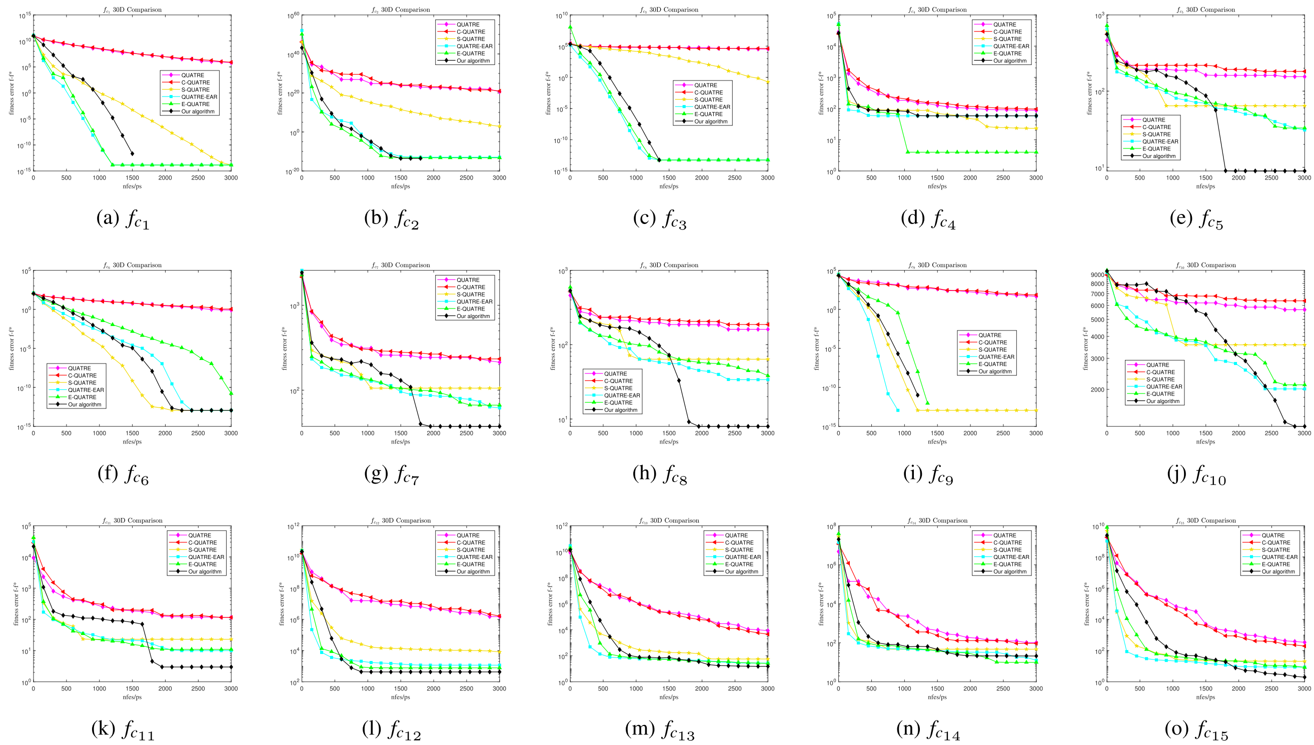


FIGURE 6. **Convergence speed comparison by employing the median value of 51 runs obtained by each algorithm on CEC2017 30-D optimization. There are total 30 comparison figures and the first 15 figures are presented here.**

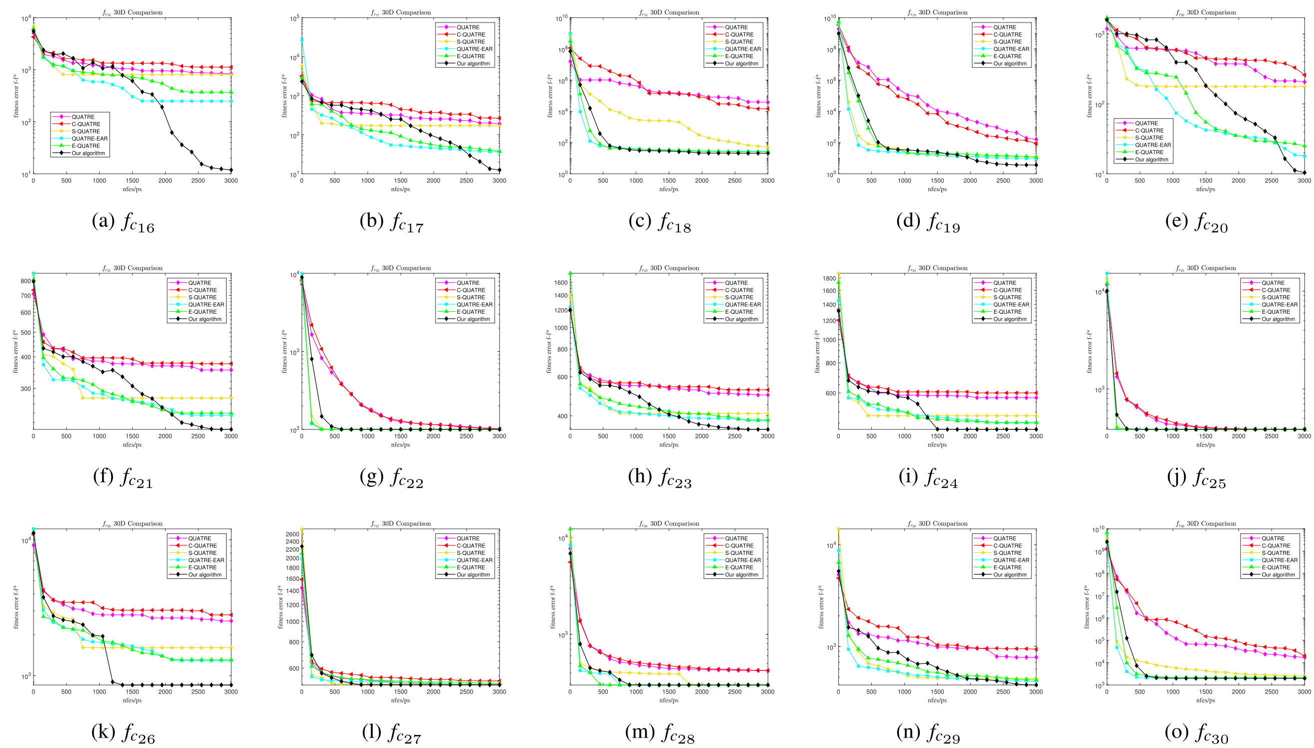


FIGURE 7. **Convergence speed comparison by employing the median value of 51 runs obtained by each algorithm on CEC2017 30-D optimization. There are total 30 comparison figures and the last 15 figures are presented here.**

probability of each different form of the trial vector from the donor vector to be equal, and the second scheme is to set the probability that the number of parameters obtained by the trial vector from the donor vector is equal. In order to compare the influence of the two initialization schemes on the algorithm, the new algorithm uses the two initialization schemes to

TABLE 10. Summarize the results of the comparison between the new proposed QUATRE and the QUATRE variants using Wilcoxon's signed rank test with the significant level $\alpha = 0.05$ under the CEC2013, CEC2014, and CEC2017 benchmark on 10D, 30D, and 50D.

QUATRE-PM compare with QUATRE and QUATRE variants										
Test suit:	CEC2013			CEC2014			CEC2017			All
dimension:	D=10	D=30	D=50	D=10	D=30	D=50	D=10	D=30	D=50	$\sum$
result:	>/≈/<	>/≈/<	>/≈/<	>/≈/<	>/≈/<	>/≈/<	>/≈/<	>/≈/<	>/≈/<	>/≈/<
QUATRE	2/1/25	1/1/26	1/0/27	0/2/28	0/1/29	1/0/29	1/3/26	0/1/29	0/0/30	6/9/249
C-QUATRE	2/0/26	1/0/27	1/1/26	0/2/28	0/1/29	1/0/29	0/4/26	0/0/30	0/0/30	5/8/251
S-QUATRE	2/3/23	1/2/25	1/2/25	0/4/26	0/3/27	0/2/28	0/6/24	1/1/28	0/1/29	5/24/235
QUATRE-EAR	4/8/16	3/5/20	3/4/21	0/10/20	3/3/24	2/5/23	2/14/14	1/7/22	2/0/28	20/56/188
E-QUATRE	3/11/14	4/8/16	3/5/20	2/9/19	2/3/25	4/3/23	4/14/12	1/5/24	2/1/27	25/59/180

TABLE 11. The results of the comparison of the 51-run adaptation errors of the first initialization scheme *M* and the second initialization scheme *M* in the newly proposed QUATRE algorithm at the significance level $\alpha = 0.05$ under the Wilcoxon signed rank test, with the same number of function evaluations as the total $nfe_{max} = 10000 \cdot D$.

Function	Statistic	first scheme	second scheme	first scheme	second scheme	first scheme	second scheme
f_1	Mean/Std	*0/0(≈)*	*0/0*	2.2681E+00/1.6134E+01(<)	**2.1313E-12/9.6516E-12**	7.0776E-14/4.6122E-13(<)	**0/0**
	p-value	1.0000	-	< 0.0001	-	0.0013	-
f_2	Mean/Std	2.2564E+02/1.1644E+03(<)	**8.9166E-15/4.5475E-14**	1.6719E-15/6.9619E-15(≈)	**0/0**	6.9994E-07/4.9783E-06(<)	**5.5729E-16/4.0194E-15**
	p-value	< 0.0001	-	0.1489	-	< 0.0001	-
f_3	Mean/Std	3.3450E+04/2.1171E+05(<)	**3.9531E-03/1.9711E-02**	7.8578E-12/2.7845E-11(<)	**0/0**	2.0565E-10/1.4256E-09(<)	**0/0**
	p-value	< 0.0001	-	< 0.0001	-	0.0002	-
f_4	Mean/Std	6.6875E-14/1.4735E-13(<)	**0/0**	9.2024E-02/2.1166E-01(<)	**1.5782E-12/1.0687E-11**	**4.8721E+01/2.1845E+01**(≈)	5.5461E+01/1.3990E+01
	p-value	0.0003	-	< 0.0001	-	0.1575	-
f_5	Mean/Std	**0/0**(>)	8.0250E-14/9.6466E-14	**2.0102E+01/5.4120E-02**(>)	2.0134E+01/6.1427E-02	1.2023E+01/3.6365E+00(<)	**8.0182E+00/2.4807E+00**
	p-value	< 0.0001	-	0.0062	-	< 0.0001	-
f_6	Mean/Std	3.3547E+00/3.7743E+00(<)	**4.7661E-05/2.4372E-04**	**1.2285E-05/8.7734E-05**(≈)	1.0175E-02/7.2665E-02	**1.2483E-13/3.5951E-14**(>)	1.5493E-07/4.3189E-07
	p-value	< 0.0001	-	0.8539	-	< 0.0001	-
f_7	Mean/Std	2.0018E-01/2.5597E-01(<)	**1.5675E-02/2.5341E-02**	*0/0(≈)*	*0/0*	4.1864E+01/3.7038E+00(<)	**3.7482E+01/2.1931E+00**
	p-value	< 0.0001	-	1.0000	-	< 0.0001	-
f_8	Mean/Std	2.1128E+01/8.2737E-02(≈)	**2.1126E+01/7.8298E-02**	**4.0125E-14/6.8212E-14**(≈)	4.2354E-14/7.0081E-14	1.0457E+01/3.0268E+00(<)	**7.4329E+00/2.5666E+00**
	p-value	0.9813	-	0.8427	-	< 0.0001	-
f_9	Mean/Std	1.3921E+01/3.8254E+00(<)	**8.4543E+00/5.1947E+00**	1.1003E+01/4.5872E+00(<)	**7.6085E+00/3.1641E+00**	*0/0(≈)*	*0/0*
	p-value	< 0.0001	-	0.0003	-	1.0000	-
f_{10}	Mean/Std	1.3964E-02/8.9198E-03(<)	**0/0**	1.1022E-02/1.7345E-02(≈)	**1.0206E-02/1.3410E-02**	1.2317E+03/2.8999E+02(≈)	**1.2208E+03/2.8618E+02**
	p-value	< 0.0001	-	0.8929	-	0.6904	-
f_{11}	Mean/Std	**2.0062E-14/3.4106E-14**(≈)	1.3656E-01/9.7525E-01	1.0172E+03/2.7369E+02(≈)	**1.0114E+03/3.4760E+02**	**5.0957E+00/8.5695E+00**(≈)	6.7473E+00/1.3995E+01
	p-value	0.7167	-	0.6904	-	0.7894	-
f_{12}	Mean/Std	7.9207E+00/2.7209E+00(<)	**4.8577E+00/2.0932E+00**	**7.0289E-02/2.6458E-02**(>)	8.0216E-02/2.7765E-02	7.8072E+02/4.0752E+02(<)	**4.1582E+02/2.6752E+02**
	p-value	< 0.0001	-	0.0248	-	< 0.0001	-
f_{13}	Mean/Std	8.8970E+00/4.8950E+00(<)	**3.8817E+00/2.8753E+00**	3.9277E-02/1.8220E-02(<)	**3.0767E-02/9.9207E-03**	5.4229E+02/1.6881E+03(<)	**1.3200E+01/7.3174E+00**
	p-value	< 0.0001	-	0.0044	-	0.0018	-
f_{14}	Mean/Std	**8.4093E-02/4.6172E-02**(≈)	8.5726E-02/4.7227E-02	**2.2837E-01/3.1711E-02**(>)	2.5718E-01/4.6361E-02	5.1090E+01/8.2116E+01(<)	**1.5207E+01/8.9286E+00**
	p-value	0.8518	-	0.0004	-	< 0.0001	-
f_{15}	Mean/Std	**2.5248E+03/4.5782E+02**(≈)	2.5883E+03/4.9131E+02	2.0846E+00/4.2499E-01(≈)	**1.9651E+00/3.7573E-01**	1.3517E+02/5.0946E+02(<)	**2.5609E+00/1.7714E+00**
	p-value	0.6973	-	0.1059	-	< 0.0001	-
f_{16}	Mean/Std	9.4795E-01/9.1274E-01(≈)	**8.8053E-01/1.0676E+00**	7.3438E+00/7.2586E-01(≈)	**7.3207E+00/7.9240E-01**	1.2260E+02/1.1349E+02(<)	**1.3793E+01/2.0795E+01**
	p-value	0.3757	-	0.9179	-	< 0.0001	-
f_{17}	Mean/Std	*3.0434E+01/6.4910E-05(≈)*	*3.0434E+01/9.4299E-07*	1.0195E+02/8.6708E+01(<)	**5.3297E+01/2.8819E+01**	**1.3528E+01/7.7179E+00**(≈)	1.4415E+01/6.4812E+00
	p-value	1.0000	-	< 0.0001	-	0.5835	-
f_{18}	Mean/Std	4.0945E+01/3.4853E+00(<)	**3.8395E+01/2.4770E+00**	3.8948E+01/9.6334E+01(<)	**2.0094E+00/1.2248E+00**	3.3439E+03/1.1441E+04(<)	**2.0733E+01/2.9950E+00**
	p-value	0.0001	-	< 0.0001	-	0.0012	-
f_{19}	Mean/Std	**8.9818E-01/1.5049E-01**(≈)	1.0156E+00/4.3531E-01	2.4590E+00/5.2846E-01(≈)	**2.2188E+00/7.7133E-01**	1.3200E+02/5.2539E+02(<)	**4.0348E+00/1.3927E+00**
	p-value	0.3181	-	0.1424	-	0.0013	-
f_{20}	Mean/Std	8.3308E+00/5.1447E-01(≈)	**8.2696E+00/5.7234E-01**	5.2512E+02/1.0849E+03(<)	**2.4530E+00/1.1365E+00**	4.2523E+01/5.1490E+01(<)	**2.0380E+01/2.9433E+01**
	p-value	0.6095	-	< 0.0001	-	0.0071	-
f_{21}	Mean/Std	3.0563E+02/2.8140E+01(≈)	**3.0342E+02/4.9361E+01**	1.3195E+03/5.1022E+03(<)	**1.3284E+01/2.5594E+01**	2.1060E+02/4.2014E+00(<)	**2.0657E+02/2.8168E+00**
	p-value	0.8336	-	< 0.0001	-	< 0.0001	-
f_{22}	Mean/Std	**1.0585E+02/6.9317E-01**(≈)	1.0616E+02/1.1273E+00	1.0354E+02/5.9011E+01(<)	**4.8012E+01/5.2287E+01**	*1.0000E+02/0.0000E+00(≈)*	*1.0000E+02/6.4311E-14*
	p-value	0.1148	-	< 0.0001	-	1.0000	-
f_{23}	Mean/Std	2.2734E+03/4.7461E+02(≈)	**2.2533E+03/4.3299E+02**	*3.1524E+02/9.2973E-13(≈)*	*3.1524E+02/9.7320E-13*	3.5103E+02/3.8145E+00(<)	**3.4821E+02/4.7784E+00**
	p-value	0.8476	-	1.0000	-	0.0030	-
f_{24}	Mean/Std	2.0001E+02/8.5334E-03(<)	**2.0000E+02/3.7048E-03**	2.2183E+02/3.5735E-01(<)	**2.1677E+02/9.4260E+00**	4.2509E+02/2.5952E+00(<)	**4.2346E+02/2.6842E+00**
	p-value	0.0005	-	0.0014	-	0.0002	-
f_{25}	Mean/Std	2.4361E+02/1.2050E+01(<)	**2.2067E+02/2.0752E+01**	2.0326E+02/9.2026E-01(<)	**2.0261E+02/4.9404E-02**	3.8682E+02/5.0961E-02(<)	**3.8672E+02/1.9649E-02**
	p-value	< 0.0001	-	< 0.0001	-	< 0.0001	-
f_{26}	Mean/Std	*2.0000E+02/1.2152E-04(≈)*	*2.0000E+02/7.8765E-14*	1.0004E+02/1.3445E-02(≈)	**1.0003E+02/9.8654E-03**	8.8352E+02/4.1485E+01(<)	**8.5719E+02/4.2467E+01**
	p-value	1.0000	-	0.1535	-	0.0007	-
f_{27}	Mean/Std	3.0054E+02/8.5432E-01(<)	**3.0006E+02/2.0792E-01**	*3.0196E+02/1.4003E+01(≈)*	*3.0196E+02/1.4003E+01*	5.0230E+02/3.5528E+00(<)	**4.9923E+02/7.3979E+00**
	p-value	< 0.0001	-	1.0000	-	0.0043	-
f_{28}	Mean/Std	*3.0000E+02/2.6123E-13(≈)*	*3.0000E+02/0.0000E+00*	8.3832E+02/2.3393E+01(<)	**8.2461E+02/2.3394E+01**	**3.0852E+02/2.9535E+01**(≈)	3.2353E+02/4.5395E+01
	p-value	1.0000	-	0.0046	-	0.1014	-
f_{29}	Mean/Std	-		6.3006E+02/2.2576E+02(≈)	**6.2536E+02/2.1293E+02**	4.2572E+02/1.3674E+01(<)	**4.1858E+02/8.3261E+00**
	p-value			0.3657	-	0.0004	-
f_{30}	Mean/Std	-		7.0485E+02/3.2560E+02(<)	**4.3768E+02/5.6858E+01**	2.0262E+03/5.3430E+01(<)	**2.0038E+03/3.8431E+01**
	p-value			< 0.0001	-	0.0136	-
>/≈/<		1/14/13	-/-/-	3/13/14	-/-/-	1/7/22	-/-/-

compare under the 30 dimensions of CEC2013, CEC2014 and CEC2017, and the specific results are shown in Table 11. For the 30D comparison in CEC2013, the first initialization scheme is better than the second initialization scheme on f_{a_5}. The second initialization method is better than the first one on $f_{a_2} - f_{a_4}, f_{a_6}, f_{a_7}, f_{a_9}, f_{a_{10}}, f_{a_{12}}, f_{a_{13}}, f_{a_{18}}, f_{a_{24}}, f_{a_{25}}$ and $f_{a_{27}}$. For the 30D comparison in CEC2014, the first initialization scheme is better than the second initialization scheme on f_{b_5}, $f_{b_{12}}$ and $f_{b_{14}}$. The second initialization method is better than the first one on $f_{b_1}, f_{b_3}, f_{b_4}, f_{b_9}, f_{b_{13}}, f_{b_{17}}, f_{b_{18}}, f_{b_{20}} - f_{b_{22}}, f_{b_{24}}$, $f_{b_{25}}$ $f_{b_{28}}$ and $f_{b_{30}}$. For the 30D comparison in CEC2017, the first initialization scheme is better than the second initialization scheme on f_{c_6}. The second initialization method is better than the first one on $f_{c_1} - f_{c_3}, f_{c_5}, f_{c_7}, f_{c_8}, f_{c_{12}} - f_{c_{16}}, f_{c_{18}} - f_{c_{21}}$, $f_{c_{23}} - f_{c_{27}}, f_{c_{29}}$ and $f_{c_{30}}$. In summary, for the QUATRE-PM, the second initialization scheme will be more suitable.

TABLE 12. The results of the comparison of the 51-run adaptation errors of used and none used a perturbation strategy in the newly proposed QUATRE algorithm at the significance level $\alpha = 0.05$ under the Wilcoxon signed rank test, with the same number of function evaluations as the total $nfe_{max} = 10000 \cdot D$.

Function	Statistic	none	used	none	used	none	used
f_1	Mean/Std	*0/0(≈)*	*0/0*	3.2582E-11/1.2412E-10(<)	**2.1313E-12/9.6516E-12**	2.7864E-16/2.0097E-15(≈)	**0/0**
	p-value	1.0000	-	0.0193	-	1.0000	-
f_2	Mean/Std	4.9041E-14/1.0665E-13(<)	**8.9166E-15/4.5475E-14**	5.5729E-16/4.0194E-15(≈)	**0/0**	4.4583E-15/1.1369E-14(<)	**5.5729E-16/4.0194E-15**
	p-value	0.0070	-	1.0000	-	0.0117	-
f_3	Mean/Std	**4.1069E-08/2.4106E-07**(≈)	3.9531E-03/1.9711E-02	*0/0(≈)*	*0/0*	*0/0(≈)*	*0/0*
	p-value	0.0576	-	1.0000	-	1.0000	-
f_4	Mean/Std	*0/0(≈)*	*0/0*	**3.5666E-14/7.0081E-14**(≈)	1.5782E-12/1.0687E-11	5.5740E+01/1.3845E+01(≈)	**5.5461E+01/1.3990E+01**
	p-value	1.0000	-	0.1428	-	0.7729	-
f_5	Mean/Std	1.0923E-13/4.5475E-14(<)	**8.0250E-14/9.6466E-14**	**2.0133E+01/7.4703E-02**(≈)	2.0134E+01/6.1427E-02	**7.7870E+00/2.7224E+00**(≈)	8.0182E+00/2.4807E+00
	p-value	0.0050	-	0.8956	-	0.6160	-
f_6	Mean/Std	1.2255E-02/7.5067E-02(≈)	**4.7661E-05/2.4372E-04**	**3.5734E-05/2.5306E-04**(≈)	1.0175E-02/7.2665E-02	**8.6754E-08/2.4782E-07**(≈)	1.5493E-07/4.3189E-07
	p-value	0.1621	-	0.3964	-	0.4317	-
f_7	Mean/Std	**1.5049E-02/3.1391E-02**(≈)	1.5675E-02/2.5341E-02	2.2292E-15/1.6078E-14(≈)	**0/0**	3.9465E+01/3.3108E+00(<)	**3.7482E+01/2.1931E+00**
	p-value	0.7678	-	1.0000	-	0.0011	-
f_8	Mean/Std	**2.0879E+01/1.0319E-01**(>)	2.1126E+01/7.8298E-02	**3.3437E-14/6.2269E-14**(≈)	4.2354E-14/7.0081E-14	8.7273E+00/3.5281E+00(<)	**7.4329E+00/2.5666E+00**
	p-value	< 0.0001	-	0.4236	-	0.0094	-
f_9	Mean/Std	2.6111E+01/1.7495E+00(<)	**8.4543E+00/5.1947E+00**	8.3336E+00/3.1393E+00(≈)	**7.6085E+00/3.1641E+00**	*0/0(≈)*	*0/0*
	p-value	< 0.0001	-	0.4015	-	1.0000	-
f_{10}	Mean/Std	1.4502E-04/1.0357E-03(≈)	**0/0**	**6.9397E-03/1.2946E-02**(≈)	1.0206E-02/1.3410E-02	1.4987E+03/2.3422E+02(<)	**1.2208E+03/2.8618E+02**
	p-value	1.0000	-	0.3543	-	< 0.0001	-
f_{11}	Mean/Std	**2.3406E-14/3.6839E-14**(≈)	1.3656E-01/9.7525E-01	1.2633E+03/1.8358E+02(<)	**1.0114E+03/3.4760E+02**	1.3500E+01/2.3034E+01(≈)	**6.7473E+00/1.3995E+01**
	p-value	0.4254	-	< 0.0001	-	0.2883	-
f_{12}	Mean/Std	5.6850E+00/1.8672E+00(<)	**4.8577E+00/2.0932E+00**	1.7183E-01/3.7338E-02(<)	**8.0216E-02/2.7765E-02**	**3.5921E+02/2.6514E+02**(≈)	4.1582E+02/2.6752E+02
	p-value	0.0097	-	< 0.0001	-	0.2007	-
f_{13}	Mean/Std	9.0426E+00/5.8078E+00(<)	**3.8817E+00/2.8753E+00**	1.1566E-01/1.9001E-02(<)	**3.0767E-02/9.9207E-03**	1.5605E+01/6.6596E+00(≈)	**1.3200E+01/7.3174E+00**
	p-value	< 0.0001	-	< 0.0001	-	0.0837	-
f_{14}	Mean/Std	**5.8376E-02/3.9506E-02**(>)	8.5726E-02/4.7227E-02	**1.8751E-01/2.6007E-02**(>)	2.5718E-01/4.6361E-02	**1.2852E+01/6.3481E+00**(≈)	1.5207E+01/8.9286E+00
	p-value	0.0017	-	< 0.0001	-	0.0669	-
f_{15}	Mean/Std	2.7402E+03/2.7874E+02(<)	**2.5883E+03/4.9131E+02**	2.1992E+00/2.3636E-01(<)	**1.9651E+00/3.7573E-01**	4.1546E+00/1.6196E+00(<)	**2.5609E+00/1.7714E+00**
	p-value	0.0150	-	0.0004	-	< 0.0001	-
f_{16}	Mean/Std	**8.1515E-01/1.8110E-01**(≈)	8.8053E-01/1.0676E+00	8.9557E+00/2.6558E-01(<)	**7.3207E+00/7.9240E-01**	8.6311E+01/7.4481E+01(<)	**1.3793E+01/2.0795E+01**
	p-value	0.0565	-	< 0.0001	-	< 0.0001	-
f_{17}	Mean/Std	*3.0434E+01/1.3202E-06(≈)*	*3.0434E+01/9.4299E-07*	6.6086E+01/3.4693E+01(<)	**5.3297E+01/2.8819E+01**	3.1350E+01/7.6623E+00(<)	**1.4415E+01/6.4812E+00**
	p-value	1.0000	-	0.0058	-	< 0.0001	-
f_{18}	Mean/Std	5.2305E+01/4.2367E+00(<)	**3.8395E+01/2.4770E+00**	2.0801E+00/1.1312E+00(≈)	**2.0094E+00/1.2248E+00**	**1.9785E+01/4.7529E+00**(≈)	2.0733E+01/2.9950E+00
	p-value	< 0.0001	-	0.8403	-	0.8038	-
f_{19}	Mean/Std	1.1986E+00/1.2524E-01(<)	**1.0156E+00/4.3531E-01**	2.4692E+00/6.1112E-01(≈)	**2.2188E+00/7.7133E-01**	5.8789E+00/9.8280E-01(<)	**4.0348E+00/1.3927E+00**
	p-value	< 0.0001	-	0.0962	-	< 0.0001	-
f_{20}	Mean/Std	9.1485E+00/4.3470E-01(<)	**8.2696E+00/5.7234E-01**	2.7180E+00/7.9928E-01(≈)	**2.4530E+00/1.1365E+00**	4.1521E+01/2.8634E+01(<)	**2.0380E+01/2.9433E+01**
	p-value	< 0.0001	-	0.1816	-	< 0.0001	-
f_{21}	Mean/Std	**2.9779E+02/4.0239E+01**(≈)	3.0342E+02/4.9361E+01	2.0109E+01/2.4719E+01(<)	**1.3284E+01/2.5594E+01**	2.0857E+02/3.0999E+00(<)	**2.0657E+02/2.8168E+00**
	p-value	0.4007	-	< 0.0001	-	0.0009	-
f_{22}	Mean/Std	**1.0595E+02/6.6437E-01**(≈)	1.0616E+02/1.1273E+00	5.5515E+01/5.0013E+01(≈)	**4.8012E+01/5.2287E+01**	*1.0000E+02/6.4311E-14(≈)*	*1.0000E+02/6.4311E-14*
	p-value	0.4196	-	0.0712	-	1.0000	-
f_{23}	Mean/Std	2.7708E+03/2.5611E+02(<)	**2.2533E+03/4.3299E+02**	*3.1524E+02/9.1854E-13(≈)*	*3.1524E+02/9.7320E-13*	**3.4501E+02/4.2940E+00**(>)	3.4821E+02/4.7784E+00
	p-value	< 0.0001	-	1.0000	-	0.0008	-
f_{24}	Mean/Std	*2.0000E+02/1.3061E-02(≈)*	*2.0000E+02/3.7048E-03*	**2.1550E+02/1.0120E+01**(≈)	2.1677E+02/9.4260E+00	**4.2186E+02/2.4542E+00**(>)	4.2346E+02/2.6842E+00
	p-value	0.0811	-	0.6820	-	0.0006	-
f_{25}	Mean/Std	2.2748E+02/1.9996E+01(≈)	**2.2067E+02/2.0752E+01**	2.0262E+02/8.0647E-02(≈)	**2.0261E+02/4.9404E-02**	*3.8672E+02/2.0362E-02(≈)*	*3.8672E+02/1.9649E-02*
	p-value	0.0873	-	0.5111	-	0.6119	-
f_{26}	Mean/Std	2.0196E+02/1.4003E+01(≈)	**2.0000E+02/7.8765E-14**	1.0011E+02/2.2247E-02(<)	**1.0003E+02/9.8654E-03**	**8.5084E+02/5.8634E+01**(≈)	8.5719E+02/4.2467E+01
	p-value	1.0000	-	< 0.0001	-	0.8075	-
f_{27}	Mean/Std	**3.0004E+02/6.7473E-02**(≈)	3.0006E+02/2.0792E-01	3.0197E+02/1.4062E+01(≈)	**3.0196E+02/1.4003E+01**	**4.9831E+02/6.5308E+00**(≈)	4.9923E+02/7.3979E+00
	p-value	0.6326	-	1.0000	-	0.4338	-
f_{28}	Mean/Std	*3.0000E+02/1.7612E-13(≈)*	*3.0000E+02/0.0000E+00*	8.3028E+02/2.7775E+01(≈)	**8.2461E+02/2.3394E+01**	**3.0852E+02/2.9535E+01**(≈)	3.2353E+02/4.5395E+01
	p-value	1.0000	-	0.1816	-	0.0956	-
f_{29}	Mean/Std	-		6.2558E+02/2.1303E+02(≈)	**6.2536E+02/2.1293E+02**	4.3343E+02/8.5120E+00(<)	**4.1858E+02/8.3261E+00**
	p-value			0.5560	-	< 0.0001	-
f_{30}	Mean/Std	-		4.4173E+02/7.0546E+01(≈)	**4.3768E+02/5.6858E+01**	**1.9956E+03/2.9303E+01**(≈)	2.0038E+03/3.8431E+01
	p-value			0.9290	-	0.2796	-
>/≈/<		2/16/10	-/-/-	1/20/9	-/-/-	2/17/11	-/-/-

TABLE 13. Time complexity comparison of these algorithms on 30D optimization under benchmark f 14 from CEC2013 test suite for real-parameter single-objective optimization.

Algorithms.	$\overline{T_0}$	$\overline{T_1}$	$\overline{T_2}$	$\frac{\overline{T_2}-\overline{T_1}}{\overline{T_0}}$
LSHADE	0.0543	0.6799	1.1368	6.8798
EVDE			3.5333	48.6307
jSO			1.1500	7.10976
LPalmDE			1.7073	16.8188
MadDE			1.3627	10.8153
QUATRE			1.6632	18.1087
C-QUATRE			1.9854	24.0423
S-QUATRE			1.8547	19.3868
QUATRE-EAR			2.1624	24.7474
E-QUATRE			2.3901	28.7143
Our algorithm			1.8549	19.3902

E. ANALYSIS OF PERTURBATION STRATEGY

In order to prevent the algorithm from falling into local optimal points, a new perturbation mechanism is proposed in this paper and applied to the QUATRE-PM. The primary purpose is to determine the individuals in the stagnant phase and trace them back to the discarded excellent trial vector. In order to verify the generality of the method for algorithm improvement, this paper verifies the proposed QUATRE under CEC2013, CEC2014, and CEC2017 on 30D used and without used using the perturbation mechanism, as shown in Table 12. For the 30D comparison in CEC2013, which has 28 benchmarks, the used perturbation mechanism obtained better performance on 13 benchmarks and similar performance on 14 benchmarks. For the 30D comparison in CEC2014, which has 30 benchmarks, the used perturbation mechanism obtained better performance on 14 benchmarks and similar performance on 13 benchmarks. For the 30D comparison in CEC2017, which has 30 benchmarks, the used perturbation mechanism obtained better performance on 22 benchmarks and similar performance on 7 benchmarks.

F. ANALYSIS OF ALGORITHM COMPLEXITY

For the space complexity, in the QUATRE-PM, the memory space is $(0.5 + 1.6) * ps$ during the iterative process, where the external archive population size is $1.6 \cdot ps$, and the discarded population size is $0.5{*}ps$; ps is one at the beginning, decreases gradually during the iteration, and is 4 at the end. To summarize, the space of the new QUATRE algorithm is $O(ps)$ at the beginning of the iteration, and the end is $O(1)$.

For the time complexity, the paper follows the Congress on Evolutionary Computation Competition test suite guidelines. Three variables, $\overline{T_0}$, $\overline{T_1}$, and $\overline{T_2}$, are used in the evaluation, where $\overline{T_0}$ denotes the time consumption of basic arithmetic expressions in the CEC2013 test suite, $\overline{T_1}$ denotes the time consumed to perform 200000 evaluations of the 14^{th} function of 30D of CEC2013, $\overline{T_2}$ denotes the time required for the algorithm to complete 300000 maximum number of fitness evaluations for 14^{th} functions in 30D of CEC2013, $\frac{\overline{T_2}-\overline{T_1}}{\overline{T_0}}$ denotes the time complexity required for each algorithm. To determine the average values of $\overline{T_0}$, $\overline{T_1}$, and $\overline{T_2}$, up to 51 separate runs may be made. Table 13 shows the time complexity of the DE variants and QUATRE variants. It can be seen from the table that the time complexity of the QUATRE is slightly higher than that of DE, but the proposed QUATRE has a good performance in the QUATRE. It is a necessary consumption, and the generation of the evolution matrix consumes extra time to improve the performance of the algorithm.

V. CONCLUSION

DE stands out from other algorithms because of its simple implementation and fast convergence ability. However, the crossover operator of DE has an unavoidable bias in the spatial search. QUATRE solves this bias in DE algorithms using the evolution matrix but still has significant search efficiency and final result challenges. Therefore, we propose QUATRE-PM to tackle these problems and improve the performance of the algorithm.

This paper proposes QUATRE-PM, which uses a new evolution matrix generation method and individual perturbation mechanism and weights based dimensionality improvement. In the new proposed QUATRE, there are three main contributions. First, the new evolution matrix initialization is generated in an unbiased scheme. It can react more quickly in iterations than the previous evolution matrix generation method to generate a suitable and stable evolution matrix and adjust when it is not suitable. Second, the perturbation mechanism is proposed in this paper. If individuals are trapped in the local optimum, the perturbation mechanism may help them jump out using excellent discarded trial vectors. Third, two excellent schemes of parameter control are employed. The historical memory mechanism can avoid parameter F and the evolution matrix trapped in the wrong direction, and linear population size reduction for parameter ps can balance the exploration and exploitation during the evolution process. The proposed QUATRE is compared with several excellent DE variants and QUATRE algorithms under 88 benchmark functions in three test suites, including CEC2013, CEC2014, and CEC2017, and the results show that the performance of our algorithm is highly competitive both in terms of optimization accuracy and convergence speed.

Although the QUATRE-PM algorithm obtained good results, some work can still be enhanced in the future. In particular, the operation mechanism in the adaption of the evolutionary matrix is slightly complicated, and searching for some parameter control strategies is more suitable for using the QUATRE algorithms. In future work, we plan to solve these problems and develop better algorithms.

REFERENCES

[1] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. Cham, Switzerland: Springer, 2015.
[2] N. Liu, J.-S. Pan, C. Sun, and S.-C. Chu, "An efficient surrogate-assisted quasi-affine transformation evolutionary algorithm for expensive optimization problems," *Knowl.-Based Syst.*, vol. 209, Dec. 2020, Art. no. 106418.
[3] Z. Meng, Y. Zhong, G. Mao, and Y. Liang, "PSO-sono: A novel PSO variant for single-objective numerical optimization," *Inf. Sci.*, vol. 586, pp. 176–191, Mar. 2022.
[4] D. Liu, Z. Hu, Q. Su, and M. Liu, "A niching differential evolution algorithm for the large-scale combined heat and power economic dispatch problem," *Appl. Soft Comput.*, vol. 113, Dec. 2021, Art. no. 108017.
[5] G. Sun, R. Han, L. Deng, C. Li, and G. Yang, "Hierarchical structure-based joint operations algorithm for global optimization," *Swarm Evol. Comput.*, vol. 79, Jun. 2023, Art. no. 101311.
[6] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 4, Nov. 1995, pp. 1942–1948.
[7] H. Peng, W. Xiao, Y. Han, A. Jiang, Z. Xu, M. Li, and Z. Wu, "Multi-strategy firefly algorithm with selective ensemble for complex engineering optimization problems," *Appl. Soft Comput.*, vol. 120, May 2022, Art. no. 108634.
[8] H. Peng, C. Wang, Y. Han, W. Xiao, X. Zhou, and Z. Wu, "Micro multi-strategy multi-objective artificial bee colony algorithm for microgrid energy optimization," *Future Gener. Comput. Syst.*, vol. 131, pp. 59–74, Jun. 2022.
[9] H. Peng, Z. Zeng, C. Deng, and Z. Wu, "Multi-strategy serial cuckoo search algorithm for global optimization," *Knowl.-Based Syst.*, vol. 214, Feb. 2021, Art. no. 106729.
[10] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, Nov. 2006.
[11] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014.
[12] D. Whitley, "A genetic algorithm tutorial," *Statist. Comput.*, vol. 4, no. 2, pp. 65–85, Jun. 1994.
[13] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.
[14] Z. Meng, J.-S. Pan, and H. Xu, "QUasi-Affine TRansformation Evolutionary (QUATRE) algorithm: A cooperative swarm based algorithm for global optimization," *Knowl.-Based Syst.*, vol. 109, pp. 104–121, Oct. 2016.
[15] K.-H. Han, K.-H. Park, C.-H. Lee, and J.-H. Kim, "Parallel quantum-inspired genetic algorithm for combinatorial optimization problem," in *Proc. Congr. Evol. Comput.*, vol. 2, Jun. 2001, pp. 1422–1429.
[16] P. A. Vikhar, "Evolutionary algorithms: A critical review and its future prospects," in *Proc. Int. Conf. Global Trends Signal Process., Inf. Comput. Commun. (ICGTSPICC)*, Dec. 2016, pp. 261–265.
[17] S. Das, S. S. Mullick, and P. N. Suganthan, "Recent advances in differential evolution—An updated survey," *Swarm Evol. Comput.*, vol. 27, pp. 1–30, Apr. 2016.
[18] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.
[19] F. Neri and V. Tirronen, "Recent advances in differential evolution: A survey and experimental analysis," *Artif. Intell. Rev.*, vol. 33, nos. 1–2, pp. 61–106, Feb. 2010.
[20] P. J. M. Van Laarhoven and E. H. L. Aarts, *Simulated Annealing*. Cham, Switzerland: Springer, 1987.

[21] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Appl. Soft Comput.*, vol. 11, no. 2, pp. 1679–1696, Mar. 2011.

[22] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 55–66, Feb. 2011.

[23] G. Wu, X. Shen, H. Li, H. Chen, A. Lin, and P. N. Suganthan, "Ensemble of differential evolution variants," *Inf. Sci.*, vol. 423, pp. 172–186, Jan. 2018.

[24] R. Gämperle, S. D. Müller, and P. Koumoutsakos, "A parameter study for differential evolution," *Adv. Intell. Syst., Fuzzy Syst., Evol. Comput.*, vol. 10, no. 10, pp. 293–298, 2002.

[25] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello, "A comparative study of differential evolution variants for global optimization," in *Proc. 8th Annu. Conf. Genetic Evol. Comput.*, Jul. 2006, pp. 485–492.

[26] K. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Berlin, Germany: Springer, 2006.

[27] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp. 646–657, Dec. 2006.

[28] J. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, Oct. 2009.

[29] R. Tanabe and A. S. Fukunaga, "Improving the search performance of SHADE using linear population size reduction," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2014, pp. 1658–1665.

[30] Z. Meng, J.-S. Pan, and L. Kong, "Parameters with adaptive learning mechanism (PALM) for the enhancement of differential evolution," *Knowl.-Based Syst.*, vol. 141, pp. 92–112, Feb. 2018.

[31] F. Zhao, X. He, G. Yang, W. Ma, C. Zhang, and H. Song, "A hybrid iterated local search algorithm with adaptive perturbation mechanism by success-history based parameter adaptation for differential evolution (SHADE)," *Eng. Optim.*, vol. 52, no. 3, pp. 367–383, Mar. 2020.

[32] H. Peng, Y. Han, C. Deng, J. Wang, and Z. Wu, "Multi-strategy co-evolutionary differential evolution for mixed-variable optimization," *Knowl.-Based Syst.*, vol. 229, Oct. 2021, Art. no. 107366.

[33] G. Sun, G. Yang, and G. Zhang, "Two-level parameter cooperation-based population regeneration framework for differential evolution," *Swarm Evol. Comput.*, vol. 75, Dec. 2022, Art. no. 101122.

[34] D. Zaharie, "Influence of crossover on the behavior of differential evolution algorithms," *Appl. Soft Comput.*, vol. 9, no. 3, pp. 1126–1138, Jun. 2009.

[35] Z. Meng, J.-S. Pan, and F. Lin, "The QUATRE structure: An efficient approach to tackling the structure bias in differential evolution," in *Proc. IEEE Int. Conf. Syst., Man Cybern. (SMC)*, Oct. 2019, pp. 3074–3079.

[36] Z. Meng and J.-S. Pan, "A competitive QUasi-affine TRansformation evolutionary (C-QUATRE) algorithm for global optimization," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2016, pp. 001644–001649.

[37] J.-S. Pan, Z. Meng, S.-C. Chu, and J. F. Roddick, "Quatre algorithm with sort strategy for global optimization in comparison with De and PSO variants," in *Proc. Euro-China Conf. Intell. Data Anal. Appl.* Cham, Switzerland: Springer, 2017, pp. 314–323.

[38] Z. Meng, Y. Chen, X. Li, C. Yang, and Y. Zhong, "Enhancing QUasi-Affine TRansformation Evolution (QUATRE) with adaptation scheme on numerical optimization," *Knowl.-Based Syst.*, vol. 197, Jun. 2020, Art. no. 105908.

[39] S.-C. Chu, Y. Chen, F. Meng, C. Yang, J.-S. Pan, and Z. Meng, "Internal search of the evolution matrix in QUasi-affine transformation evolution (QUATRE) algorithm," *J. Intell. Fuzzy Syst.*, vol. 38, no. 5, pp. 5673–5684, May 2020.

[40] J. J. Liang, B. Qu, P. N. Suganthan, and A. G. Hernández-Díaz, "Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization," Comput. Intell. Lab., Zhengzhou Univ., Zhengzhou, China and Nanyang Technol. Univ., Singapore, Tech. Rep. 201212, no. 34, 2013, pp. 281–295.

[41] J. J. Liang, B. Y. Qu, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization," Comput. Intell. Lab., Zhengzhou Univ., Nanyang Technol. Univ., Zhengzhou China, Singapore, Tech. Rep. 635, 2013, p. 490.

[42] G. Wu, R. Mallipeddi, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2017 competition and special session on constrained single objective real-parameter optimization," IEEE, Zhengzhou, China, Tech. Rep. 201709, 2016.

[43] Z. Meng and J.-S. Pan, "QUasi-Affine TRansformation Evolution with External ARchive (QUATRE-EAR): An enhanced structure for differential evolution," *Knowl.-Based Syst.*, vol. 155, pp. 35–53, Sep. 2018.

[44] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2013, pp. 71–78.

[45] C. Li, G. Sun, L. Deng, L. Qiao, and G. Yang, "A population state evaluation-based improvement framework for differential evolution," *Inf. Sci.*, vol. 629, pp. 15–38, Jun. 2023.

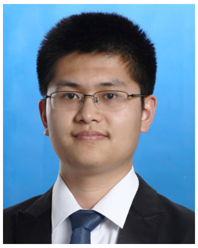
[46] Z. Meng and C. Yang, "Two-stage differential evolution with novel parameter control," *Inf. Sci.*, vol. 596, pp. 321–342, Jun. 2022.

[47] J. Brest, M. S. Maucec, and B. Boškovic, "Single objective real-parameter optimization: Algorithm jSO," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2017, pp. 1311–1318.

[48] Z. Meng, J.-S. Pan, and K.-K. Tseng, "PaDE: An enhanced differential evolution algorithm with novel control parameter adaptation schemes for numerical optimization," *Knowl.-Based Syst.*, vol. 168, pp. 80–99, Mar. 2019.

[49] S. Biswas, D. Saha, S. De, A. D. Cobb, S. Das, and B. A. Jalaian, "Improving differential evolution through Bayesian hyperparameter optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2021, pp. 832–840.

JUNYUAN ZHANG received the bachelor's degree in computer science and technology from the Beijing University of Posts and Telecommunications, in 2020. He is currently pursuing the master's degree with the Fujian University of Technology. His research interest includes evolutionary computation.



ZHENYU MENG (Member, IEEE) received the B.S. degree in computer science from Shandong Normal University, in 2008, the M.Phil. degree from the Harbin Institute of Technology Shenzhen Graduate School, in 2011, and the Ph.D. degree in computer science from the Harbin Institute of Technology, Shenzhen, in 2018.

After completing the M.Phil. degree, he was with the Guangzhou Institute of Advanced Technology, Chinese Academy of Sciences, as a Research Assistant, from 2012 to 2013. Currently, he is the Vice Dean of Computer Science and Mathematics, the Director of the Institute of Artificial Intelligence, and a Full Professor with the Fujian Key Provincial Key Laboratory of Data Mining and Application, Fujian University of Technology. His research interests include evolutionary computation, swarm intelligence, and vehicle navigation.