Journal of Network Intelligence
Taiwan Ubiquitous Information
 ISSN 2414-8105 (Online)
Volume **4**, Number **2**, May **2019**

A Parallel Quasi-Affine Transformation Evolution Algorithm for Global Optimization

Bing-Qing Jiang[1], Jeng-Shyang Pan[1,2,3*]

[1]Department of Computer Science and Engineering
Harbin Institute of Technology(Shenzhen), Shenzhen, China
[2]College of Computer Science and Technology
Sandong University of Science and Technology, Qingdao, China
[3]Department of Information Management
Chaoyang University of Technology, Taiwan
[*]Corresponding author: jspan@cc.kuas.edu.tw

Received September 2018; revised February 2019

Abstract. *QUasi-Affine TRansformation Evolutionary (QUATRE) algorithm is a new proposed population based optimization algorithm in tackling tough real-parameter optimization problems. In this paper, a new parallel QUATRE algorithm, P-QUATRE, is proposed to enhance the overall performance of the former QUATRE algorithm by incorporating a parallel mechanism. The general idea of the P-QUATRE can be summarized as follows. Firstly, all particles are partitioned into different sub-populations. Then particles in every subpopulation evolve independently with the same evolution strategy as the canonical QUATRE algorithm. Finally after every some fixed generations, the global best particle in every subpopulation immigrates to their own neighbouring sub-populations to replace some bad particles in corresponding neighbouring sub-populations, which would accelerate the convergence speed. To validate the proposed P-QUATRE, several experiments under BBOB2009 test suit are conducted to contrast the P-QUATRE, Particle Swarm Optimization (PSO) variants, Differential Evolution (DE) and QUATRE algorithm. Experimental results demonstrate that P-QUATRE outperforms the other algorithms from a general perspective of view.*
Keywords: Benchmark function, Optimization, Parallel mechanism, QUATRE

1. **Introduction.** There is a proverb in China, "absorbing it's essence and resisting it's dark side", which vividly explains what optimization is. Conventionally, we get a minimum or maximum value of one function by means of gradient of the function. However, when the function is a complex and high dimension function, calculating the gradient will be time-consuming, furthermore some of functions are non-differentiable. Sometimes it is not necessary to get the optimal solution of a function, and suboptimal solution of a function meets our demand adequately. Intelligent computing is a discipline in which algorithms solve optimization problem by imitating nature phenomenon, such as Particle Swarm Optimization (PSO) [1] simulating the behavior of bird flock, Bat algorithm (BA) [2] imitating bat searching for food, Ebb-tide-fish algorithm [3, 4] simulating the foraging behavior of fish for food, Monkey King Evolution (MKE) algorithm [5] mimicking the behavior of Monkey King, a character of a famous novel "Journal to the West", and so on. In these algorithms, differential evolution (DE) which uses mutation [6], crossover and selection operators to evolve population has attracted more attention from researchers, because of it's effectiveness and simplicity for implementation. However, the performance

of DE is highly related to the parameter settings and choice for mutation strategy [7, 8]. So there are many studies of DE on how to set up parameters or adaptive parameter, such as [9, 10, 11, 12].

Recently, a new stochastic algorithm called Quasi-Affine Transformation Evolutionary algorithm (QUATRE) [13, 14, 15, 16, 17, 18, 19] for real-parameter optimization was proposed for tackling some weakness and inconvenience of the former MKE algorithm and DE algorithm. However, QUATRE is still in it's infancy, there is some room for improvement. In QUATRE, there is only one population, going against accelerating the search process. In this paper, in order to solve the aforementioned problems, we introduce parallel mechanism that sub-populations share their knowledge obtained in the way that the global best particle in one subpopulation immigrates to neighbouring sub-populations to replace some certain percents of bad particles, after every predefined generation.

The rest of the paper is organized as follows. Section 2 reviews several related works; Section 3, the detail of proposed algorithm is described; Section 4 discusses how to set parameters and validates P-QUATRE algorithm; Section 5 gives the final conclusion.

2. **Related Work.** This part introduces QUATRE [13, 14, 15, 16, 17, 18, 19], shown in Algorithm 1, and it's variants. In Algorithm 1, a particle is represented as a vector $\vec{x}_i$, i denotes the i^{th} particle in the population. A population is represented as a matrix X. There are ps particles in a population and each particle in a D-dimension search domain. A particle and a population can be represented in Eq.(1) and Eq.(2) respectively. P is a matrix of the i^{th} row vector which represents the personal best particle of i^{th} particle ("pbest") where the i^{th} particle gets it's best fitness value. $\vec{g}$ represents the global best particle ("gbest") in population from first generation to current generation.

In the initialization phase, all particles are randomly placed in the search domain, personal best particle is identical to current particle, global best particle is selected according to the fitness value of particles, and control parameter c is set.

$$\vec{x}_i = (x_{i1}, x_{i2}, ..., x_{iD}) \tag{1}$$

$$X = \begin{bmatrix} \vec{x}_1 \\ \vec{x}_2 \\ .... \\ \vec{x}_{ps} \end{bmatrix} \tag{2}$$

$$\begin{cases} B = G + c * (X_{r1} - X_{r2}) \\ X_{trial} = M \otimes X + \bar{M} \otimes B \end{cases} \tag{3}$$

After the initialization procedure, trial particles produced by Eq.(3) and selection are repeated until termination criteria is satisfied. In line 2 of Algorithm 1, trial particles are generated according to Eq.(3). Where X_{r1} and X_{r2} is a random matrix by means of random permutation for all row vectors in X. c is a scale parameter of differential matrix (the difference between X_{r1} and X_{r2}), which is recommended 0.7 in [14]. G is a matrix consisting of ps global best particles through tiling the global best particle like Eq.(4). M is a selecting matrix. In order to obtain M, a unit lower triangular matrix with D columns and D rows is tiled along column direction. Then there are two sequential steps. First step, elements in every row vector are randomly permutated. Second step, row vectors are randomly permutated, while elements in every row vector are unchanged. $\bar{M}$ is reversed M, which means that one element in M is one, corresponding value in $\bar{M}$ is zero; one element in M is zero, corresponding value in $\bar{M}$ is one. The transformation

likes Eq.(5), here we assume the search domain is 4 dimensions and there are 9 particles. The matrix M plays a role as a judgment, which determines components of the trial particle from previous generation or B. The symbol $\otimes$ in Eq.(3) represents the element-wise multiplication. X_{trial} consists of all trial particles, all of which compare with their personal best particle according to their fitness value such that the winners will survive in the next generation. In lines 3-10 of pseudo code of QUATRE, if i_{th} trial particle $X_{trial,i}$ is more optimal than it's personal best particle P_i , the i_{th} personal best particle is replaced by i_{th} trial particle $X_{trial,i}$; if i_{th} trial particle $X_{trial,i}$ is more optimal than the global best particle , the global best particle is replaced by i_{th} trial particle $X_{trial,i}$. Line 11 of pseudo code of QUATRE, particles of next generation are generated by retaining all personal best particles.

$$G = \begin{bmatrix} \vec{g} \\ \vec{g} \\ ... \\ \vec{g} \end{bmatrix} \tag{4}$$

TABLE 1. Six different schemas for generating B

No	Schema	Equation
1	QUATRE/target/1	$B = X + c * (X_{r1} - X_{r2})$
2	QUATRE/rand/1	$B = X_{r1} + c * (X_{r2} - X_{r3})$
3	QUATRE/best/1	$B = G + c * (X_{r1} - X_{r2})$
4	QUATRE/target/2	$B = X + c * (X_{r1} - X_{r2}) + c * (X_{r3} - X_{r4})$
5	QUATRE/rand/2	$B = X_{r1} + c * (X_{r2} - X_{r3}) + c * (X_{r4} - X_{r5})$
6	QUATRE/best/2	$B = G + c * (X_{r1} - X_{r2}) + c * (X_{r3} - X_{r4})$

In [13], author proposes 5 variants of QUATRE according to the different method generating B like Eq.(3), shown in table 1. In order to clearly explain the whole variants of QUATRE, we use $QUATRE/x/y$ to describe all variants of QUATRE. Where x denotes the matrix to be disturbed, and y denotes the number of difference matrix used in corresponding schema. x and y together determine the disturbed matrix, B. Schema 1, $QUATRE/target/1$, where $target$ denotes the current population X, 1 denotes population X is disturbed by one difference matrix. Schema 2, $QUATRE/rand/1$, where $rand$ denotes the random matrix obtained by permutating the row vectors of X, 1 denotes the random matrix is disturbed by one difference matrix. Schema 3, $QUATRE/best/1$, where $best$ denotes the global best particles G Eq.(5), 1 denotes the G is disturbed by one difference matrix. For schema 4-6, the meaning of symbols is identical or similar to what mentioned above.

$$\begin{bmatrix} 1,0,0,0 \\ 1,1,0,0 \\ 1,1,1,0 \\ 1,1,1,1 \\ 1,0,0,0 \\ 1,1,0,0 \\ 1,1,1,0 \\ 1,1,1,1 \\ 1,0,0,0 \end{bmatrix} \underline{step\ one} \begin{bmatrix} 0,0,1,0 \\ 1,0,0,1 \\ 0,1,1,1 \\ 1,1,1,1 \\ 0,1,0,0 \\ 0,1,1,0 \\ 1,0,1,1 \\ 1,1,1,1 \\ 0,0,0,1 \end{bmatrix} \underline{step\ two} \begin{bmatrix} 0,0,0,1 \\ 1,1,1,1 \\ 1,0,1,1 \\ 0,1,1,0 \\ 0,1,0,0 \\ 1,1,1,1 \\ 0,1,1,1 \\ 1,0,0,1 \\ 0,0,1,0 \end{bmatrix} = M\ \bar{M} = \begin{bmatrix} 1,1,1,0 \\ 0,0,0,0 \\ 0,1,0,0 \\ 1,0,0,1 \\ 1,0,1,1 \\ 0,0,0,0 \\ 1,0,0,0 \\ 0,1,1,0 \\ 1,1,0,1 \end{bmatrix} \tag{5}$$

Algorithm 1 Pseudo code of QUATRE

Require: Initialization: Initialize the search space V, locations of particles X,$P \leftarrow X$,$\vec{g}$ and benchmark function f(X)
Require: Iteration:
1: **while** $exeTime < MaxIteration \mid !stopCriteria$ **do**
2: according to Eq.(3) to produce trial particles X_{trial}
3: **for** $i = 1 \rightarrow ps$ **do**
4: **if** $f(X_{trial,i})$ *more optimal than* $f(P_i)$ **then**
5: updating P_i $Pi \leftarrow f(X_{trial,i})$
6: **end if**
7: **if** $f(X_{trial,i})$ *more optimal than* $f(\vec{g})$ **then**
8: updating global best particle $\vec{g} \leftarrow X_{trial,i}$
9: **end if**
10: **end for**
11: $X \leftarrow P$
12: **end while**

3. **Parallel QUATRE.** QUATRE uses only one global best particle to complete the evolution, all particles evolve surrounding the global best particle, ignoring some areas of search space and going against accelerating the search process. To avoid these problems, we introduce parallel mechanism into QUATRE. The brief description of parallel mechanism is as follow. Particles are partitioned into different parallel sub-populations evolving independently, which facilitate the search of particles for different areas. After a predefined generations i, neighbouring sub-populations will communicate with each other by replacing a certain ratio of bad particles in neighbouring sub-populations with the global particle in their own subpopulation, which accelerates the searching process by means of sharing knowledge obtained by subpopulation. How to determine which two sub-populations have neighbouring relationship will be described later. In parallel QUATRE, there are four parameters, i, d, g, c. In order to explain the meaning of four parameters, we assume there are four sub-populations A, B, C, D, A have 3 neighbouring sub-populations B, C, D and the rest (B, C, D) have only one neighbouring subpopulation A. The meaning of parameter c is identical to that of QUATRE. The meaning of parameter i is that subpopulation A communicate with B, C, D after i generations, the meaning of parameter d is that d percents of bad particles of B, C and D are replaced by the global best particle of A, the meaning of parameter g is the number of sub-populations.

Initialization stage in Algorithm 2, the Pseudo code of Parallel QUATRE, the searching space V and all particles are initialized, then all particles are partitioned into g (group size) parallel sub-populations. Here we use superscript k to represent different sub-populations, k = (1, 2 , 3, ..., g). For every subpopulation, P^k and $\vec{g}^k$ is initialized, where P^k is a matrix whose row vectors represent the personal best particle of k^{th} subpopulation and $\vec{g}^k$ represents the global best particle in k^{th} subpopulation, where k = (1, 2, 3, ..., g). Afterwards *totalbest* is initialized, where *totalbest* represents the best particle in all subpopulations. Four control parameters are initialized.

$$sort(neighbor(g)) * (d * ps) = \vec{p}^{\,g} \tag{6}$$

In the iteration stage, in line 3 of pseudo code of parallel QUATRE, every subpopulation evolves independently to generate trial particles according to Eq.(3), where X_k denotes trail particles in k^{th} sub-population. In lines 5-8 of pseudo code of parallel QUATRE,

Algorithm 2 Pseudo code of Parallel QUATRE

Initialization:

Initialize the searching space V, locations of particles for every group X^k,individual best history for every group P^k let $P^g = X^g$,calculate the gbest of every group $\vec{g}^g$, calculate the totalbest.Initialize the four control parameters i,g,d,c

1: **while** $exeTime < MaxIteration | !stopCriteria$ **do**
2: **for** $k = 1 \rightarrow gs$ **do**
3: updating location of particles of each group according to equation (3)
4: **for** $j = 1 \rightarrow ps$ **do**
5: **if** $f(X_j^k)$ *more optimal than* $f(P_j^k)$ **then**
6: updating P_j^k and $\vec{p}_j{}^k$
7: $P_j^k \leftarrow X_{g,j}$, $\vec{p}_j{}^k \leftarrow f(X_j^k)$
8: **end if**
9: **if** $f(X_j^k)$ *more optimal than* $f(\vec{g}^{\,k})$ **then**
10: $\vec{g}^k \leftarrow f(X_j^k)$, $\vec{g}^{\,k} \leftarrow X_j^k$
11: **end if**
12: $X^k \leftarrow P^k$
13: **end for**
14: **if** $\vec{g}^k$ *more optimal thantotalbest* **then**
15: $totalbest \leftarrow g^k$
16: **end if**
17: **end for**
18: **if** exeTime % i == 0 **then**
19: **for** $j = 1 \rightarrow gs$ **do**
20: according to equation (6),best particle of every group immigrate to his neighbors
21: **end for**
22: **end if**
23: **end while**

if the trial particle is more optimal than corresponding personal best particle, personal best particle is replaced by the trial particle, where P_j^k denotes the j^{th} personal best particle in k^{th} subpopulation and X_j^k denotes the j^{th} trial particle in k^{th} subpopulation. In lines 9-11 of pseudo code of parallel QUATRE, if the trial particle is more optimal than global best particle, global best particle is replaced by the trial particle, where $\vec{g}^k$ denotes the global best particle in k^{th} subpopulation. In lines 13-15 of pseudo code of parallel QUATRE, if the global best particle in k^{th} subpopulation is more optimal than *totalbest* particle, the total global best particle is replaced by the global best particle. In line 16 of pseudo code of parallel QUATRE, next generation of particles in every subpopulation are generated. In lines 18-22 of pseudo code of parallel QUATRE, the global best particle in every sub-population immigrates to neighboring sub-populations every i iteration.

In parallel particle swarm optimization (PPSO)[20], there are three strategies to implement commutation among sub-populations. In case of weak correlation or independency between variables of function, the *totalbest* particle immigrates to all sub-populations. In case of strong correlation between variables of function, the global best particle in every sub-population immigrates to their neighbouring sub-populations. In case of unknown correlation between variables of function, strategy three is produced by merging together strategy one and strategy two. Because most of functions are non-sparable function, we use strategy two, and following experiments show strategy two has comparative

TABLE 2. BBOB2009 benchmark functions

No	function	No	function
f_1	Sphere Function	f_{13}	Sharp Ridge Function
f_2	Ellipsoidal Function	f_{14}	Different Powers Function
f_3	Rastrigin Function	f_{15}	Rastrigin Function
f_4	Buche-Rastrigin Function	f_{16}	Weierstrass Function
f_5	Linear Slope	f_{17}	Schaffers F7 Function
f_6	Attractive Sector Function	f_{18}	Schaffers F7 Function, moderately ill-conditioned
f_7	Step Ellipsoidal Function	f_{19}	Composite Griewank-Rosenbrock Function F8F2
f_8	Rosenbrock Function, original	f_{20}	Schwefel Function
f_9	Rosenbrock Function, rotated	f_{21}	Gallagher's Gaussian 101-me Peaks Function
f_{10}	Ellipsoidal Function	f_{22}	Gallagher's Gaussian 21-hi Peaks Function
f_{11}	Discus Function	f_{23}	Katsuura Function
f_{12}	Bent Cigar Function	f_{24}	Lunacek bi-Rastrigin Function

performance even in separable functions. Here, we introduce how to determine two sub-populations have neighbouring relationship. Firstly, g sub-populations are numbered from 0 to (g-1). Then those numbers in decimal system are transformed into corresponding binary format. If two binary numbers have only one different bit, the two binary numbers have neighbouring relationship [20]. There is an example to explain the method. We assume there are 4 sub-populations numbered with 0, 1, 2, 3. The corresponding binary numbers of 0, 1, 2, 3 are 00, 01, 10, 11. 00 has one different bit compared with 01 and 10, so the neighbors of subpopulation 0 are subpopulation 1 and 2. In Eq.(6), the function neighbor(k) denotes finding all neighbors of subpopulation k and returning all neighbor sub-populations of k^{th} subpopulation. Function sort() sorts all particles in a subpopulation by ascending fitness value. d percentage of bad particles in a population according to ascending fitness value are replaced by $\vec{g}^k$, the global best particle in k^{th} subpopulation

4. **Experiment.** In [21, 22], many complex benchmark functions are described. In this paper, we utilize BBOB2009 test suite [21] to conduct several experiments. Benchmark functions in BBOB2009 test suit are classified into five categories, separable function (f_1-f_5), lowly or moderately conditional function (f_6-f_9), highly conditional and unimodal function (f_{10}-f_{14}), multi-modal functions of strong global structure (f_{15}-f_{19}), multi-modal functions of weak global structure (f_{20}-f_{24}). In each benchmark function in BBOB2009 test suit, there are many different instances, each of which is transformed. So on different instances of each benchmark function, the minimums are different and locate in different locations which are shifted to x_{opt}. There is a global optima value $f(x_{opt})$ in every instance of all benchmark functions and a best value found by an algorithm denoted by symbol $f*$. The fitness error is $(f* - f_{fopt})$. The 24 functions are listed in table 2, more detail, please read [21]. There are two criteria to validate the performance of one optimization algorithm, one is to compare precision (fitness error) of different algorithms under a fixed number of function evaluations, and the other is to compare the number of function evaluations firstly meeting the predefined target precision (fitness error). In following experiments, we adopt the first criteria.

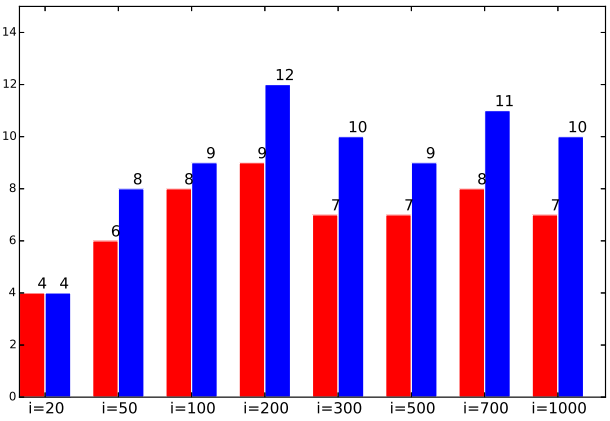
4.1. **Parameters Setting.** In this section, we discuss how to set up parameters, i, g and d, conducting three experiments. The basic setting of the three experiments is that times of iteration is 5000, the number of variables of function is 20, there are 320 particles in total and our proposed algorithm is run 10 times for each of benchmark function, using first 10 instances of each benchmark function. In order to show how to determine

TABLE 3. Comparison results of best fitness error of 10-runs for Parallel QUATRE with different coefficient i values

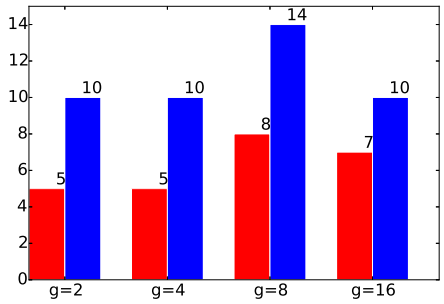
	i=50	i=100	i=200	i=300	i=500	i=700	i=1000
f_1	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
f_2	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
f_3	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
f_4	1.99e+00	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
f_5	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
f_6	7.11e-15	7.11e-15	**3.55e-15**	**3.55e-15**	7.11e-15	**3.55e-15**	**3.55e-15**
f_7	2.27e-02	2.60e-02	2.82e-02	1.93e-02	2.88e-02	**0.00e+00**	2.64e-02
f_8	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
f_9	1.78e-14	1.42e-14	**0.00e+00**	4.74e-10	3.57e-10	6.04e-08	2.98e-07
f_{10}	**5.88e+00**	1.76e+01	4.92e+01	1.02e+01	2.44e+01	1.41e+01	1.21e+01
f_{11}	**2.28e-10**	4.38e-08	6.40e-06	5.75e-05	1.16e-03	3.77e-03	1.03e-02
f_{12}	3.90e-03	3.19e-03	4.13e-03	**9.68e-05**	4.44e-04	2.45e-04	1.41e-04
f_{13}	1.21e-02	9.70e-05	1.80e-05	**1.12e-06**	1.20e-04	**1.12e-06**	6.45e-06
f_{14}	5.44e-06	**3.41e-06**	9.82e-06	1.06e-05	6.61e-06	5.47e-06	7.94e-06
f_{15}	3.04e+01	1.39e+01	2.09e+01	1.79e+01	**1.09e+01**	**1.09e+01**	1.19e+01
f_{16}	1.75e+00	2.32e+00	1.84e+00	2.25e+00	**3.42e-01**	8.80e-01	1.83e+00
f_{17}	8.38e-03	3.20e-03	2.04e-03	4.64e-04	8.81e-04	1.60e-03	**3.84e-04**
f_{18}	1.01e-01	1.21e-01	**2.47e-02**	3.91e-02	2.83e-02	3.81e-02	4.26e-02
f_{19}	3.22e+00	2.41e+00	3.14e+00	2.12e+00	1.75e+00	**1.72e+00**	2.13e+00
f_{20}	2.90e-01	2.37e-01	**5.92e-02**	1.78e-01	1.18e-01	1.18e-01	1.18e-01
f_{21}	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
f_{22}	6.92e-01	**0.00e+00**	**0.00e+00**	5.68e-14	6.92e-01	6.92e-01	6.92e-01
f_{23}	1.17e+00	1.05e+00	1.10e+00	4.41e-01	**3.39e-01**	5.57e-01	5.49e-01
f_{24}	6.21e+01	5.36e+01	5.82e+01	3.36e+01	4.00e+01	3.71e+01	**2.73e+01**

parameter i, we use i = (50, 100, 200, 300, 500, 700, 1000) to implement experiment, the other control parameters, $g = 8$, $d = 0.25$, $c = 0.7$; for parameter g, we use g = (2, 4, 8, 16) to conduct experiment, the other control parameters, $i = 100$, $d = 0.25$, $c = 0.7$; for parameter d, we utilize d = (0.5, 0.25, 0.2, 0.1, 0.05) to conduct experiment, the other control parameters, $i = 100$, $g = 8$, $c = 0.7$. The results of three experiments are showed in table 3 to table 5 and figure 1. In table 3 to table 5, the global optima of corresponding function is zero, and the best minimum among the proposed algorithms with different parameter is emphasized in boldface. In figure 1, the number of function whose global optima is found is denoted by red bar (left bar) and the number of function whose best minimum among the proposed algorithms with different parameters setting is denoted by blue bar(right bar)

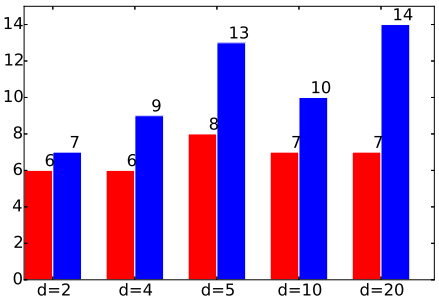
From table 3 to table 5 and figure 1, we can see that $i = 200$ is the best choice for parameter i, $g = 8$ is the best choice for parameter g, $d = 20$ is the best choice for parameter d. For parameter i, if i is too small, particles in each subpopulation have not fully explore and exploit current area, then immigrants come here. Specially, in the later of iteration, there are almost same particles in every subpopulation. Therefore i cannot be too small. For parameter g, if g is too small, the advantage of parallel mechanism is restrained, if g is too big, the diversity of particles is weakened. Therefore, g can't be too small and large. For parameter d, if d is too big, a large amount of immigrant particles from neighbouring sub-populations join in current subpopulation, however these particles



(a) Parameter i comparison



(b) Parameter g comparison



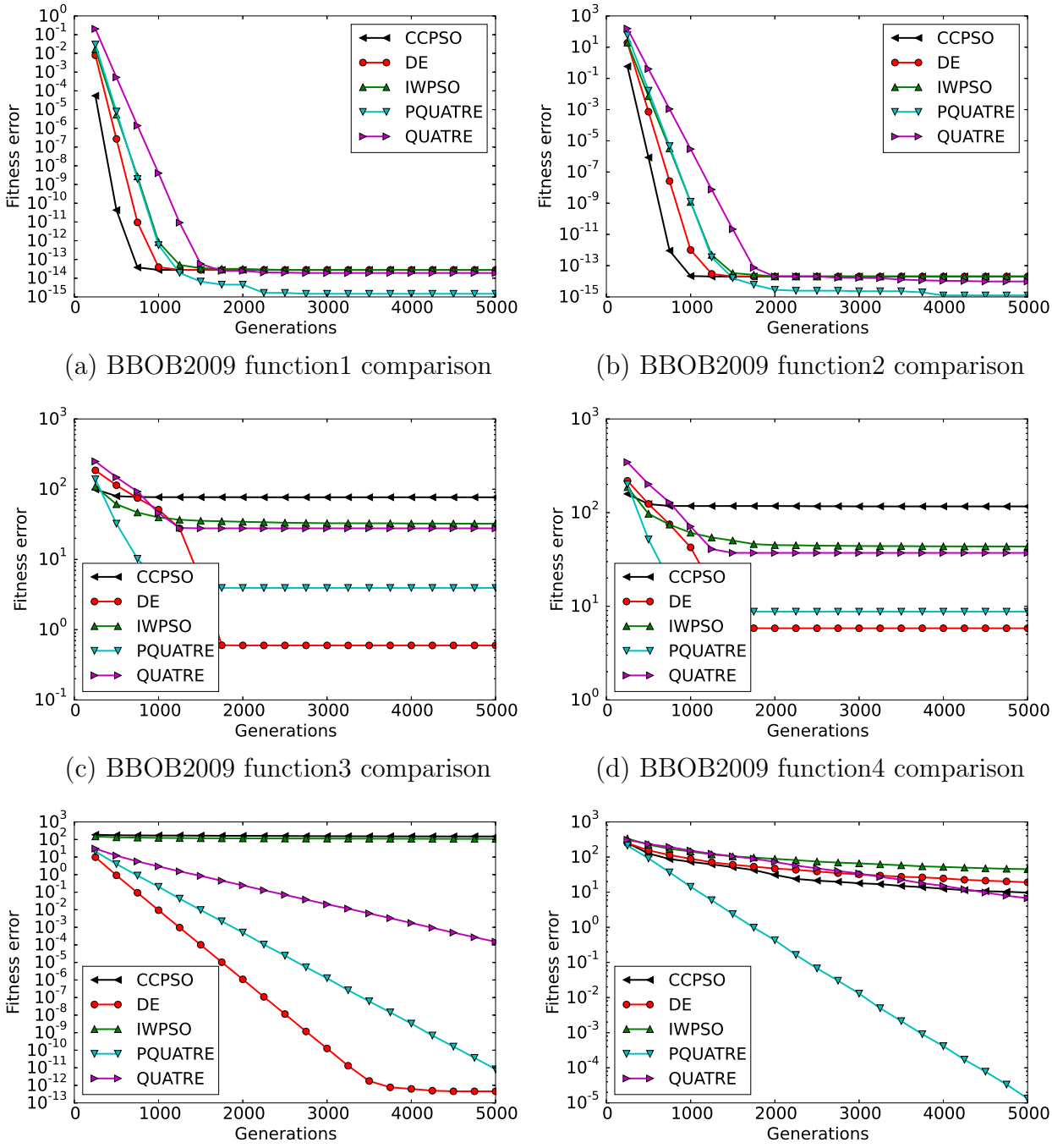
(c) Parameter d comparison

FIGURE 1. The number of functions with 10 trials that obtains global optima and best minimum among different parameter settings

belong to only g(group size) types, which lead to reduction of diversity of particles, so d cannot be too big.

4.2. **Algorithm Analysis and Comparison.** In order to validate the performance of our proposed algorithm, we compare it with other algorithms, such as Particle Swarm Optimization with inertia weight (IWPSO) [23], the constriction coefficient PSO (CCPSO) [24], QUATRE [14], DE [6] in terms of convergence speed, precision and stability, where the role of IWPSO, CCOPSO and DE is benchmark algorithm. The parameter settings of all algorithms are shown in table 6. For IWPSO and CCPSO, the velocity is initialized with zero. In our experiment, each benchmark function in BBOB2009 has 50 variables. For all comparative algorithms, we use 16 particles in every dimension (convenient for grouping), so there are 800 particles at one run in total. Every algorithm runs 20 times (the first 20 instances of every benchmark function) with 5000 generations to obtain the minimum of the benchmark functions from BBOB2009. Total number of function evaluations of every run is 4 million. The best and standard deviation of fitness errors in 20 runs are listed in table 7 and 8 respectively, where the best value is emphasized in boldface. From table 7, best fitness error in 20-run among different algorithms, we can draw several conclusions in terms of precision.

Firstly, for separable functions $f_1 - f_5$, DE exhibits the best performance. P-QUATRE and QUATRE exhibit comparative performance. CCPSO and IWPSO exhibits worse performance than other algorithms. DE outperforms other algorithms on functions $f_3 - f_5$,



(a) BBOB2009 function1 comparison

(b) BBOB2009 function2 comparison

(c) BBOB2009 function3 comparison

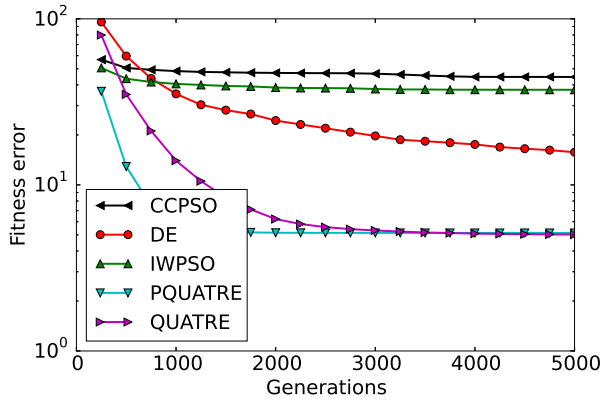
(d) BBOB2009 function4 comparison

(e) BBOB2009 function5 comparison

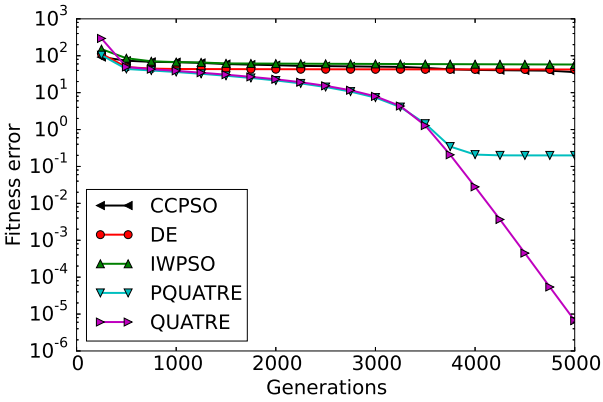
(f) BBOB2009 function6 comparison

FIGURE 2. BBOB2009 benchmark functions($f_1 - f_6$) comparison of the best of fitness errors

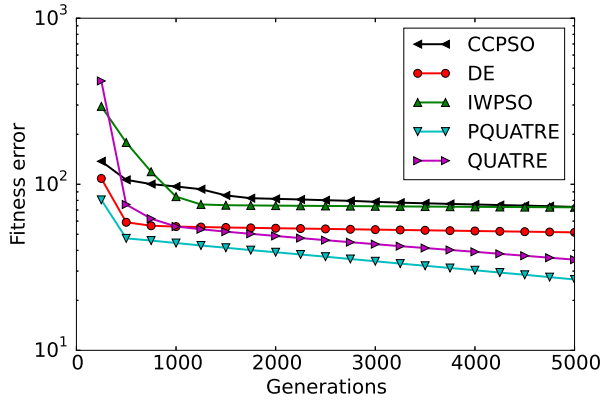
while P-QUATRE exhibits comparative performance on functions $f_4 - f_5$. QUATRE and P-QUATRE outperforms other peers on functions f_1 and f_2, in which QUATRE and P-QUATRE finds the global optima. Secondly, as for lowly or moderately conditional functions $f_6 - f_9$, P-QUATRE shows the best performance. P-QUATRE is superior to other comparative ones on functions $f_6 - f_8$, while P-QUATRE is inferior to QUATRE on function f_9 where P-QUATRE shows comparative performance. Thirdly, for highly conditional and unimodal functions $f_{10} - f_{14}$, P-QUATRE shows the best performance. QUATRE shows the second best performance. P-QUATRE is superior to other peers



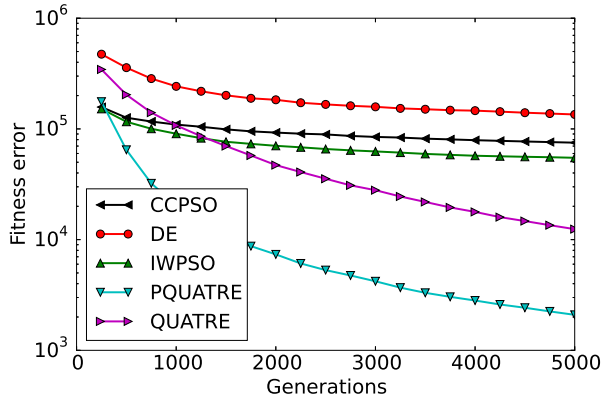
(a) BBOB2009 function7 comparison



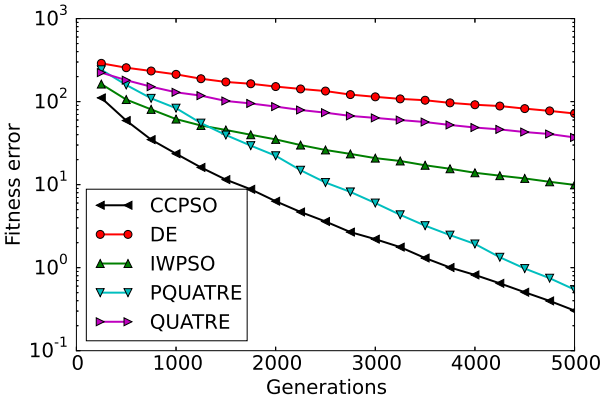
(b) BBOB2009 function8 comparison



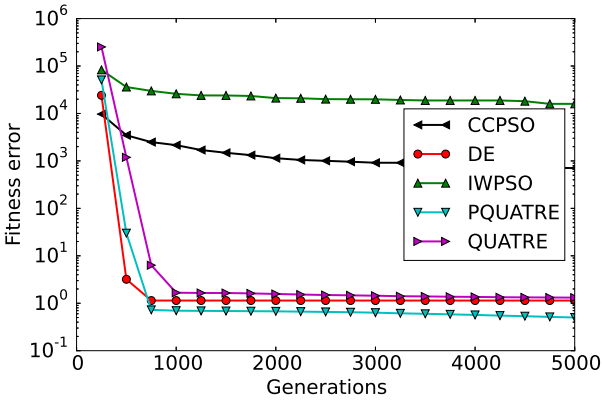
(c) BBOB2009 function9 comparison



(d) BBOB2009 function10 comparison



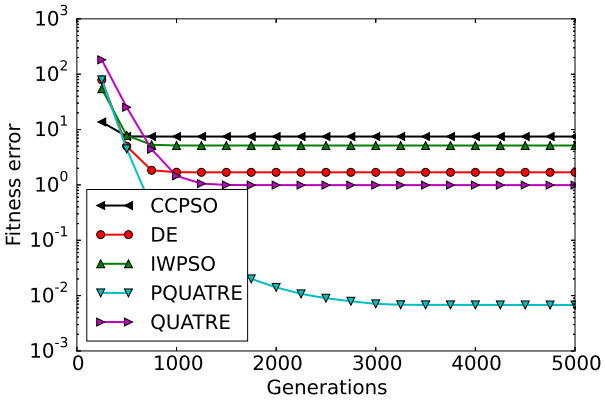
(e) BBOB2009 function11 comparison



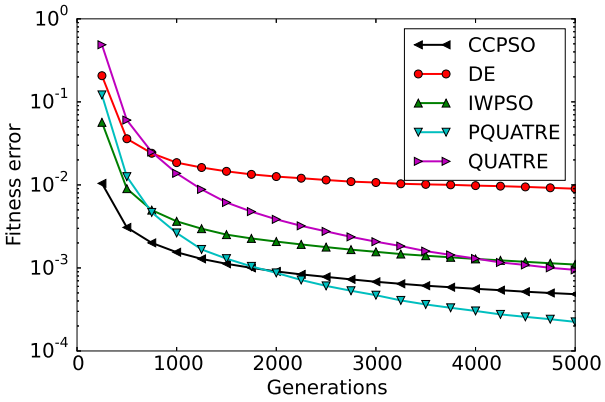
(f) BBOB2009 function12 comparison

FIGURE 3. BBOB2009 benchmark functions($f_7 - f_{12}$) comparison of the best of fitness errors

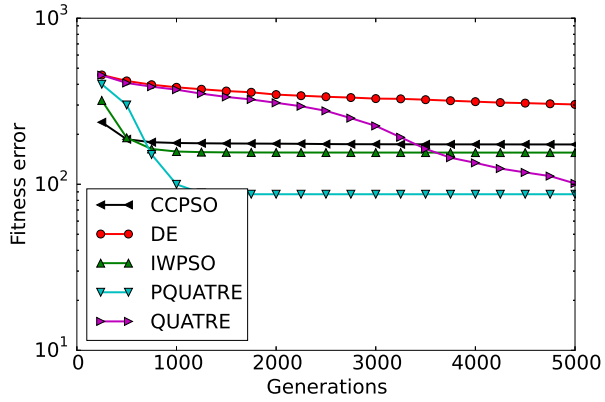
on functions f_{10}, f_{13} and f_{14}. DE shows better performance than other comparative ones on function f_{12}. CCPSO shows the best performance on function f_{11}. Fourthly, as for multi-modal functions of strong global structure $f_{15} - f_{19}$, QUATRE shows the best performance. QUATRE is superior to other peers on functions $f_{16} - f_{18}$, P-QUATRE shows better performance than other algorithms on function f_{15} and CCPSO exhibits the best performance on f_{19}. QUATRE shows comparative performance on functions $f_{16} - f_{19}$. Finally, for multi-modal functions of weak global structure $f_{20} - f_{24}$, P-QUATRE shows the best performance. P-QUATRE shows best performance on functions $f_{21} - f_{24}$, while



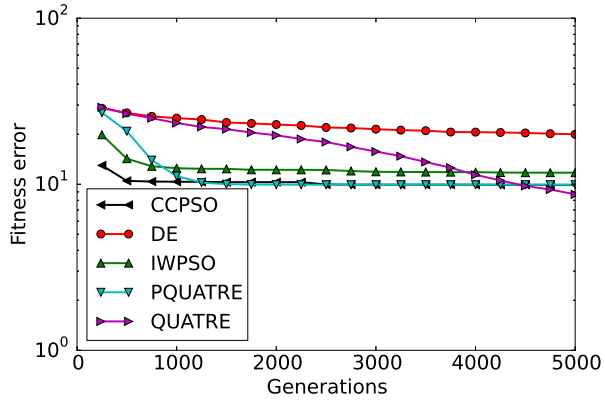
(a) BBOB2009 function13 comparison



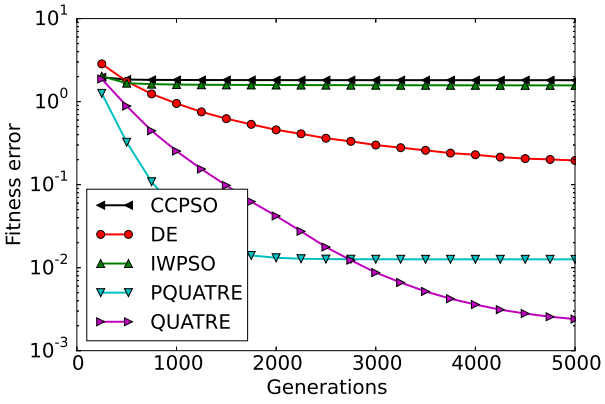
(b) BBOB2009 function14 comparison



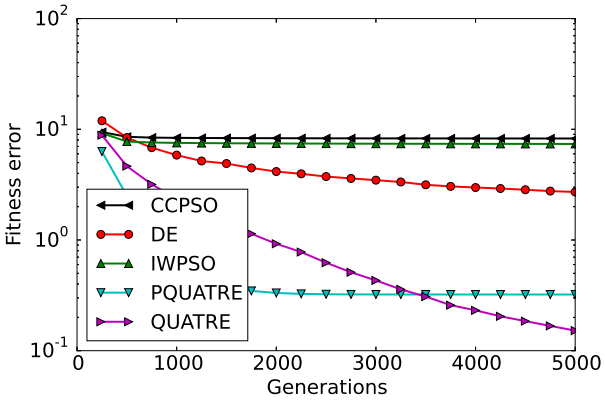
(c) BBOB2009 function15 comparison



(d) BBOB2009 function16 comparison



(e) BBOB2009 function17 comparison

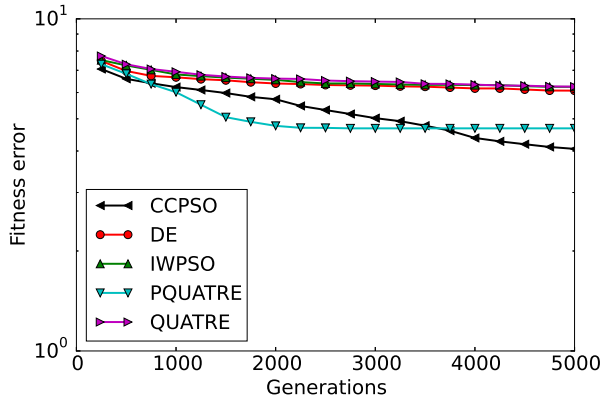


(f) BBOB2009 function18 comparison

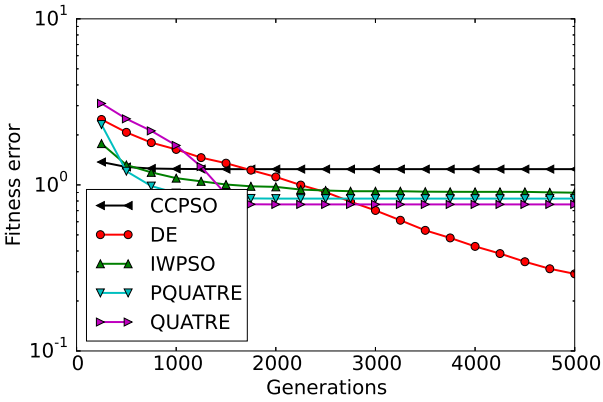
FIGURE 4. BBOB2009 benchmark functions($f_{13} - f_{18}$) comparison of the best of fitness errors

QUATRE shows same performance as P-QUATRE on functions $f_{21} - f_{22}$. There is an interesting phenomenon that all algorithms find the same local optima on function f_{22}.

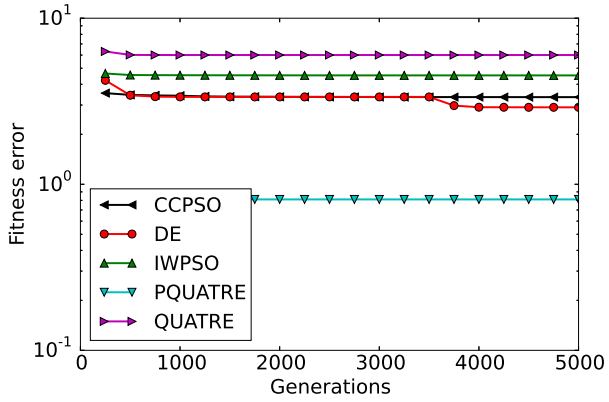
From table 8, standard deviation of fitness error in 20-runs among different algorithms, we can draw several conclusions in terms of stability. Firstly, for separable functions $f_1 - f_5$, DE shows the best stability. P-QUATRE shows comparative stability. DE beats other comparative algorithms on functions $f_3 - f_5$. P-QUATRE is superior to other peers on functions f_1, f_2 and shows comparative stability on functions $f_3 - f_5$, in which P-QUATRE shows the second best stability. Secondly, as for lowly or moderately



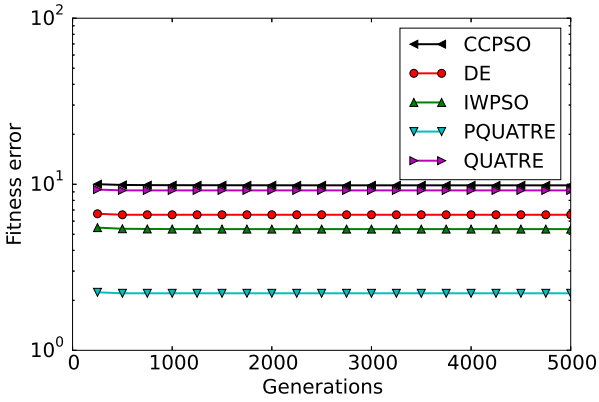
(a) BBOB2009 function19 comparison



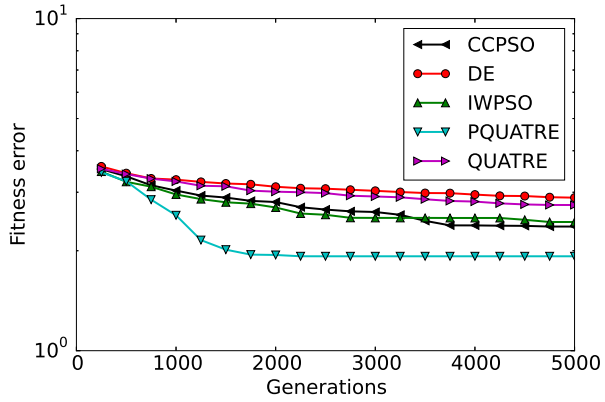
(b) BBOB2009 function20 comparison



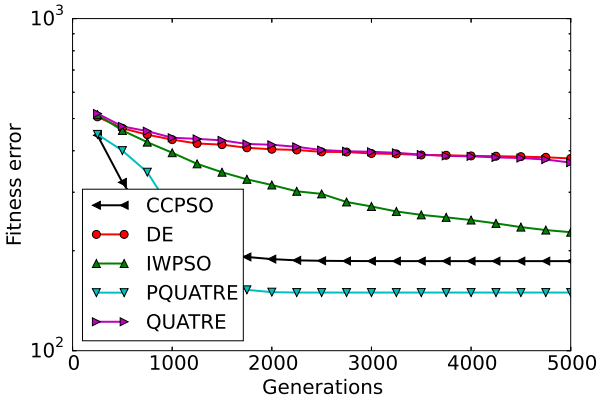
(c) BBOB2009 function21 comparison



(d) BBOB2009 function22 comparison



(e) BBOB2009 function23 comparison



(f) BBOB2009 function24 comparison

FIGURE 5. BBOB2009 benchmark functions($f_{19} - f_{24}$) comparison of the best of fitness errors

conditional functions $f_6 - f_9$, P-QUATRE shows the best stability. P-QUATRE is superior to other comparative algorithms in terms of stability on functions f_6, f_7 and f_9, while inferior to QUATRE on function f_8 where QUATRE shows the best stability. Thirdly, for highly conditional and unimodal functions $f_{10} - f_{14}$, P-QUATRE shows the best stability. P-QUATRE is superior to other comparative algorithms in terms of stability on functions $f_{10}, f_{12} - f_{14}$, while inferior to CCPSO on function f_{11} where P-QUATRE shows the second best stability. Fourthly, as for multi-modal functions of strong global structure $f_{15} - f_{19}$, DE shows the best stability and QUATRE shows the second best stability. DE is

TABLE 4. Comparison results of best fitness error of 10-runs for Parallel QUATRE with different coefficient g values

D=20	g=2	g=4	g=8	g=16
f_1	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**
f_2	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**
f_3	1.9899e+00	9.9496e-01	**0.0000e+00**	**0.0000e+00**
f_4	2.9849e+00	1.9899e+00	**0.0000e+00**	**0.0000e+00**
f_5	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**
f_6	**3.5527e-15**	**3.5527e-15**	**3.5527e-15**	7.1054e-15
f_7	**1.6543e-02**	2.4292e-02	2.6954e-02	7.6524e-02
f_8	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**
f_9	3.5527e-15	7.1054e-15	**0.0000e+00**	3.1287e-09
f_{10}	9.6523e+00	**5.5705e+00**	2.3038e+01	1.7856e+01
f_{11}	1.4282e-08	**3.7961e-09**	3.0290e-08	1.8519e-07
f_{12}	**3.5934e-04**	1.5040e-03	1.1388e-03	2.5449e-03
f_{13}	2.8913e-02	3.1396e-03	**7.2894e-05**	1.2927e-03
f_{14}	**4.5465e-06**	5.0012e-06	7.4787e-06	6.9238e-06
f_{15}	1.9868e+01	**1.4882e+01**	1.8904e+01	2.0946e+01
f_{16}	1.7161e+00	3.6535e+00	**1.2860e+00**	3.4590e+00
f_{17}	**2.1998e-04**	5.0076e-04	8.8440e-03	1.5962e-02
f_{18}	7.5174e-02	**3.9362e-02**	7.2980e-02	7.2740e-02
f_{19}	3.2014e+00	3.0131e+00	**1.7371e+00**	2.7166e+00
f_{20}	4.1453e-01	2.8623e-01	1.7766e-01	**5.9219e-02**
f_{21}	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**
f_{22}	6.9186e-01	6.9186e-01	**7.1054e-15**	**7.1054e-15**
f_{23}	7.4693e-01	7.8736e-01	**4.4215e-01**	1.0864e+00
f_{24}	5.4065e+01	4.6254e+01	6.8051e+01	**3.1093e+01**

superior to other comparative algorithms on f_{15}, f_{16} and f_{19}. QUATRE bests other peers on functions f_{17}, f_{18}. P-QUATRE also shows comparative stability. On function f_{15}, P-QUATRE beats CCPSO, IWPSO and QUATRE; on function f_{16}, P-QUATRE beats QUATRE; on functions f_{17} and f_{18}, P-QUATRE beats other comparative algorithms except for QUATRE. Finally, for multi-modal functions of weak global structure $f_{20} - f_{24}$, P-QUATRE and QUATRE shows the best stability. P-QUATRE is superior to other peers on functions f_{21} and f_{22}. QUATRE is superior to other comparative algorithms on functions f_{23} and f_{24}. IWPSO beats other peers on function f_{20}.

From figure 2-5, the comparison of the mean of the fitness errors among comparative algorithms, we can draw several conclusions in terms of convergence speed, here slope of lines is used as assessment criteria. Firstly, for separable functions $f_1 - f_5$, CCPSO and P-QUATRE shows the best convergence speed. CCPSO shows the best convergence on functions f_1 and f_2. P-QUATRE is superior to other peers on f_3 and f_4. DE beats other comparative algorithms on function f_5. On functions f_1 and f_2, P-QUATRE has similar convergence speed as IWPSO; P-QUATRE is superior to QUATRE, while inferior to DE and CCPSO. On function f_5, P-QUATRE shows the second best convergence speed. Secondly, as for lowly or moderately conditional functions $f_6 - f_9$, P-QUATRE shows the best convergence speed. On functions f_6, f_7 and f_9, P-QUATRE is superior to other peers. On function f_8, QUATRE is superior to other comparative algorithms and P-QUATRE shows the second best convergence speed. Thirdly, for highly conditional

TABLE 5. Comparison results of best fitness error of 10-runs for Parallel QUATRE with different coefficient d values

D=20	d=0.5	d=0.25	d=0.2	d=0.1	d=0.05
f_1	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**
f_2	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**
f_3	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**
f_4	1.9899e+00	9.9496e-01	**0.0000e+00**	**0.0000e+00**	9.9496e-01
f_5	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**
f_6	7.1054e-15	**3.5527e-15**	**3.5527e-15**	**3.5527e-15**	**3.5527e-15**
f_7	4.0956e-01	3.0530e-02	3.5497e-02	1.2131e-02	**0.0000e+00**
f_8	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**
f_9	1.9371e-09	**7.1054e-15**	5.3291e-14	5.6843e-14	3.5989e-12
f_{10}	1.5816e+01	1.2063e+01	1.9322e+01	**8.5713e+00**	1.5636e+01
f_{11}	1.5192e-07	**2.6978e-08**	8.0807e-08	5.4914e-07	3.1347e-05
f_{12}	5.4743e-03	3.0827e-03	8.5431e-05	3.8987e-03	**8.5009e-05**
f_{13}	5.2233e-03	9.1819e-05	**7.4416e-07**	1.7172e-04	5.0920e-04
f_{14}	**5.6730e-06**	5.8906e-06	6.8364e-06	9.5189e-06	1.1237e-05
f_{15}	2.4217e+01	1.7909e+01	1.8905e+01	1.8229e+01	**1.5919e+01**
f_{16}	3.6311e+00	3.1469e+00	**1.6558e+00**	2.7705e+00	1.7740e+00
f_{17}	2.1474e-02	2.2285e-03	3.5935e-03	2.0879e-03	**3.2275e-04**
f_{18}	2.5436e-01	1.5766e-01	1.8746e-01	3.1804e-02	**1.8360e-02**
f_{19}	3.4253e+00	3.7832e+00	3.6561e+00	2.2683e+00	**1.1473e+00**
f_{20}	3.5176e-01	2.9610e-01	**1.6779e-01**	**1.6779e-01**	2.6649e-01
f_{21}	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**
f_{22}	6.9186e-01	6.9186e-01	**0.0000e+00**	6.9186e-01	6.9186e-01
f_{23}	1.2422e+00	1.0757e+00	**6.6382e-01**	1.0607e+00	9.4269e-01
f_{24}	8.5042e+01	4.2139e+01	3.5404e+01	4.4408e+01	**3.0437e+01**

TABLE 6. Parameters setting of different algorithms

Algorithms	Parameters settings
IWPSO	$c_1 = c_2 = 2.0, iw = 0.5$
CCPSO	$c_1 = c_2 = 2.05, iw = 1, K = 0.7298$
DE	$C_r = 0.1, F = 0.5$
QUATRE	$c = 0.7$
PQUATRE	$c = 0.7, i = 200, d = 0.05, g = 8$

and unimodal functions $f_{10} - f_{14}$, P-QUATRE and CCPSO shows the best convergence speed. P-QUATRE is superior to other peers on function f_{10} and f_{13}. CCPSO is superior to other comparative algorithms on f_{11} and f_{14}. DE shows better convergence speed than other peers on function f_{12}. P-QUATRE shows the second best convergence on functions f_{11}, f_{12} and f_{14}. Fourthly, as for multi-modal functions of strong global structure f_{15}-f_{19}, P-QUATRE shows the best convergence speed. IWPSO, CCPSO and P-QUATRE have similar convergence speed on functions f_{15} and f_{16}. P-QUATRE shows better convergence speed than other algorithms on function f_{17}-f_{19}. Finally, for multi-modal functions of weak global structure $f_{20} - f_{24}$, P-QUATRE shows the best convergence speed. P-QUATRE is superior to other peers on functions f_{20}, f_{23} and f_{24}. There is an interesting

TABLE 7. Comparison results of best fitness error of 20-runs between different algorithms

D=50	CCPSO	DE	IWPSO	P-QUATRE	QUATRE
f1	4.4409e-16	4.4409e-16	4.4409e-16	**0.0000e+00**	**0.0000e+00**
f2	1.7764e-15	1.7764e-15	1.7764e-15	**0.0000e+00**	**0.0000e+00**
f3	4.6763e+01	**0.0000e+00**	1.7909e+01	9.9496e-01	1.9899e+01
f4	6.6662e+01	**2.9849e+00**	2.4874e+01	4.9748e+00	2.0894e+01
f5	1.2820e+02	**3.4106e-13**	9.8014e+01	5.9117e-12	8.6638e-05
f6	1.7982e+00	6.9037e+00	1.1960e+01	**1.0088e-06**	1.8636e+00
f7	2.4645e+01	1.0163e+01	2.0133e+01	**2.0848e+00**	2.3415e+00
f8	1.8045e-01	3.6431e+01	1.8646e+01	**2.4443e-12**	7.1023e-07
f9	3.7399e+01	4.1535e+01	4.4470e+01	2.5405e+01	**2.4377e+01**
f10	1.8794e+04	8.8538e+04	1.3672e+04	**5.6293e+02**	7.9937e+03
f11	**8.7598e-02**	5.3410e+01	5.8673e+00	1.4883e-01	2.6410e+01
f12	1.3804e-02	**4.7342e-07**	3.0198e-02	2.5917e-04	3.3889e-05
f13	5.2816e-02	1.1190e-02	1.7199e-02	**1.2393e-05**	1.1708e-03
f14	3.7611e-04	7.0517e-03	8.6333e-04	**1.5534e-04**	7.3773e-04
f15	7.7607e+01	2.6025e+02	6.4672e+01	**4.6763e+01**	5.6713e+01
f16	4.7877e+00	1.3660e+01	5.4180e+00	4.7696e+00	**1.8209e+00**
f17	5.8246e-01	1.2944e-01	4.2254e-01	3.4006e-03	**1.9279e-04**
f18	3.4435e+00	1.6145e+00	2.6955e+00	8.1219e-02	**2.0560e-02**
f19	**1.8763e+00**	5.7145e+00	5.2009e+00	3.4045e+00	5.7545e+00
f20	9.9114e-01	**9.5070e-02**	7.3814e-01	5.8825e-01	4.5402e-01
f21	2.2204e-16	7.1054e-15	7.1054e-15	**0.0000e+00**	**0.0000e+00**
f22	**6.9186e-01**	**6.9186e-01**	**6.9186e-01**	**6.9186e-01**	**6.9186e-01**
f23	9.0694e-01	2.5276e+00	1.8774e+00	**8.4688e-01**	2.3713e+00
f24	1.0696e+02	3.1879e+02	1.5545e+02	**8.9983e+01**	3.2494e+02

phenomenon that P-QUATRE converge faster than QUATRE on almost all functions, even on unimodal functions $f_{10} - f_{14}$. One reason for this phenomenon is different matrix $(X_{r1} - X_{r2})$ in a subpopulation contains more zero row vector (because 5% bad particles are replaced by the same one particle, the global best particle in neighbor subpopulation, in every communication with neighbor subpopulation) than corresponding different matrix in QUATRE, which enhance the exploitation for the area in which there is current global best particle.

In summary, DE has best performance in terms of precision and stability on separable functions, decent performance in terms of precision on multi-modal functions of weak global structure, best performance in terms of stability on multi-modal functions of strong global structure. P-QUATRE has best performance in terms of precision on lowly or moderately conditional functions, highly conditional and unimodal function and multi-modal functions of weak global structure, decent performance in terms of precision on separable functions, best performance in terms of stability on lowly or moderately conditional functions, highly conditional, unimodal function and multi-modal functions of weak global structure, decent performance in terms of stability on separable functions. QUATRE has best performance in terms of precision on multi-modal functions of strong global structure, decent performance in terms of precision on separable functions and multi-modal functions of weak global structure, decent performance in terms of stability

TABLE 8. Comparison results of standard deviation of fitness error of 20-runs between different algorithms

D=50	CCPSO	DE	IWPSO	P-QUATRE	QUATRE
f_1	2.6958e-14	2.6958e-14	2.6958e-14	**6.3479e-15**	2.7450e-14
f_2	2.0010e-14	2.0010e-14	2.0010e-14	**3.5104e-15**	1.7665e-14
f_3	1.8154e+01	**6.7713e-01**	7.1120e+00	2.1038e+00	5.3752e+00
f_4	3.2665e+01	**1.6547e+00**	1.0838e+01	2.5336e+00	7.5918e+00
f_5	1.6627e+01	**5.3675e-14**	8.3088e+00	1.8849e-12	3.1351e-05
f_6	8.3572e+00	6.3460e+00	4.2284e+01	**1.6883e-05**	5.8590e+00
f_7	1.9414e+01	3.1447e+00	1.3395e+01	**1.8825e+00**	2.2976e+00
f_8	2.4191e+01	2.3921e+00	2.9256e+01	8.9144e-01	**5.3325e-06**
f_9	4.7531e+01	2.6033e+01	5.2823e+01	**6.7179e-01**	2.4183e+01
f_{10}	4.5886e+04	2.8380e+04	3.0239e+04	**1.1392e+03**	3.1610e+03
f_{11}	**1.4178e-01**	1.0165e+01	2.7517e+00	3.0116e-01	7.0742e+00
f_{12}	3.1371e+03	1.8922e+00	7.1288e+04	**7.3090e-01**	4.0950e+00
f_{13}	9.5330e+00	2.2473e+00	9.0837e+00	**1.3485e-02**	1.5416e+00
f_{14}	5.3905e-05	1.0698e-03	1.1919e-04	**3.7814e-05**	1.2050e-04
f_{15}	5.7695e+01	**1.7888e+01**	4.5955e+01	2.1031e+01	2.1690e+01
f_{16}	2.7762e+00	**2.2071e+00**	3.5314e+00	3.5789e+00	5.6536e+00
f_{17}	9.0831e-01	3.2164e-02	1.0011e+00	6.6077e-03	**3.5102e-03**
f_{18}	3.7049e+00	6.3718e-01	3.4507e+00	2.1634e-01	**1.2494e-01**
f_{19}	1.5831e+00	**1.8001e-01**	5.4280e-01	6.3456e-01	2.5922e-01
f_{20}	1.6393e-01	2.6400e-01	**1.0122e-01**	1.3303e-01	1.3103e-01
f_{21}	5.0094e+00	2.5510e+00	5.7605e+00	**8.1358e-01**	9.5553e+00
f_{22}	7.7389e+00	6.7068e+00	6.4461e+00	**2.7059e+00**	1.0896e+01
f_{23}	6.8616e-01	2.0912e-01	3.3776e-01	5.6958e-01	**1.9324e-01**
f_{24}	3.5807e+01	2.8100e+01	6.2837e+01	3.6709e+01	**2.2933e+01**

on separable functions and multi-modal functions of strong global structure. CCPS and IWPSO have poor performance on both all function categories.

5. **Conclusions.** In this paper, we propose Parallel QUATRE algorithm, discuss the parameter settings, and then we use BBOB2009 test suit to validate the global search capacity of the proposed algorithm. Comparisons are made among our algorithm and other well-known algorithms. The comparison results demonstrate that proposed P-QUATRE has excellent performance in general. However the proposed Parallel QUATRE has poor performance than QUATRE on multi-modal functions of strong global structure (P-QUATRE shows better performance on multi-modal functions, considering multi-modal functions of weak global structure), which will be a studied key point in the future. Parallel QUATRE has too many parameters, another studied key point in the future will be studied on how to reduce the number of parameters.

REFERENCES

[1] J. Kennedy, R. Eberhart, Particle swarm optimization, *Proc. IEEE Int. Conf. Neural Networks*, vol.4, no.2, pp.1942-1948, 1995.

[2] X. S. Yang, A New Metaheuristic Bat-Inspired Algorithm, *Computer Knowledge and Technology*, vol.284, pp.65-74, 2010.

[3] Z. Meng, J. S. Pan, and A. Alelaiwi, A new meta-heuristic ebb-tide-fish-inspired algorithm for traffic navigation, *Telecommunication Systems*, vol.62, no.2, pp.403-415, 2016.

[4] J. S. Pan, Z. Meng, S. C. Chu, H. R. Xu, Monkey King Evolution: an enhanced ebb-tide-fish algorithm for global optimization and its application in vehicle navigation under wireless sensor network environment. *Telecommunication Systems*, vol.65, no.3, pp.351-364, 2017.

[5] Z. Meng, and J. S. Pan, Monkey King Evolution: A new memetic evolutionary algorithm and its application in vehicle fuel consumption optimization, *Knowledge-Based Systems*, vol.97, pp.144-157, 2017.

[6] K. V. Price, Differential evolution vs. the functions of the 2nd, ICEO. *IEEE International Conference on Evolutionary Computation IEEE*, pp.153-157, 1997.

[7] S. Das, P. N. Suganthan, Differential evolution: A survey of the state-of-the-art, *IEEE Transactions on Evolutionary Computation*, vol.15, no.1, pp.4-31, 2011

[8] Z. Meng, J. S. Pan, and L. Kong, Parameters with adaptive learning mechanism (PALM) for the enhancement of differential evolution, *Knowledge-Based Systems*, vol.141, pp.92-112, 2018.

[9] A. K. Qin, V. L. Huang, and P. N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimizatin, *IEEE Transactions on Evolutionary Computation*, vol.13, no.2, pp.398-417, 2009

[10] J. Brest, S. Greiner, B. Boscovic, M. Mernik,and V. Zumer, Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems, *IEEE Transactions on Evolutionary Computation*, vol.10, no.8, pp.646-657, 2006

[11] J. Teo, Exploring dynamic self-adaptive populations in differential evolution, *Soft Computing*, vol.10, no.8, pp.673-686, 2006

[12] J. Liu, J. Lampinen, A fuzzy adaptive differential evolution algorithm, *Soft Computing*, vol.9, no.6, pp.448-462, 2005.

[13] J. S Pan, Z. Meng, H. Xu, and X. Li, QUasi-Affine TRansformation Evolution (QUATRE) algorithm: A new simple and accurate structure for global optimization. *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, Springer International Publishing, 2016.

[14] Z. Meng, and J. S. Pan, QUasi-affine TRansformation Evolutionary (QUATRE) algorithm: A parameter-reduced differential evolution algorithm for optimization problems, *Evolutionary Computation IEEE*, pp.4082-4089, 2016.

[15] Z. Meng, J. S. Pan, and H. Xu, QUasi-Affine TRansformation Evolutionary (QUATRE) algorithm: A cooperative swarm based algorithm for global optimization, *Knowledge-Based Systems*, vol.109, pp.104-121, 2016.

[16] Z. Meng, and J. S. Pan, A Competitive QUasi-Affine TRansformation Evolutionary (C-QUATRE) Algorithm for global optimization, *Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on, IEEE*, 2016

[17] J. S. Pan, Z. Meng, QUATRE algorithm with sort strategy for global optimization in comparison with DE and PSO variants, *The Euro-China Conference on Intelligent Data Analysis and Applications*, Springer, Cham, 2017

[18] Z. Meng, J. S. Pan, and X. Li, The QUasi-Affine TRansformation Evolution (QUATRE) Algorithm: An Overview, *The Euro-China Conference on Intelligent Data Analysis and Applications*, Springer, Cham, 2017.

[19] Z. Meng, and J. S. Pan, QUasi-Affine TRansformation Evolution with External ARchive (QUATRE-EAR): an enhanced structure for differential evolution, *Knowledge-Based Systems*, vol.155, pp.35-53, 2018.

[20] J.F. Chang, et al, A parallel particle swarm optimization algorithm with communication strategies, *Journal of Information Science & Engineering*, vol.21, no.4, pp.809-818, 2005.

[21] N. Hansen, S. Finck, R. Ros, and A. Auger, Real-parameter black-box optimization benchmarking 2009: noiseless functions definitions, [Research Report] RR-6829, 2009

[22] J. J. Liang, et al, Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization, *Technical report 201212*, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, 2013

[23] Y. Shi, and R. Eberhart, A modified particle swarm optimizer, *Advances in Natural Computation*, Springer Berlin Heidelberg, pp.439-439, 1998.

[24] R. C. Eberhart, and Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, *Proceedings of the 2000 IEEE Congress on Evolutionary Computation*,La Jolla, CAIEEE, 2000.