Differential Evolution with a Dimensional Mutation Strategy for Global Optimization

Jing Guan
China Ship Development and Design Center
Wuhan, China, 430064
Email: g_jing0414@163.com

***Abstract*—Differential evolution (DE) is an efficient means of solving the global optimization problems. In the classical and adaptive DE algorithms, each individual has the same value of F, the amplification factor of the difference vector, in all dimensions. However, some researchers' works showed that population may have different characteristics of converging in different dimensions. Individuals may be very similar to each other in some dimensions, but they may have obvious difference in other dimensions. In this paper, a dimensional mutation strategy is proposed for DE. In this new mutation strategy, each individual has different values of F in different dimensions. This new mutation strategy was implied into jDE algorithm and tested on the CEC05 functions. The experimental results suggested that the dimensional mutation can make a better performance of the jDE algorithm.**

I. Introduction

Differential evolution (DE) [1] is a simple but efficient evolutionary algorithm (EA) for solving global optimization problems. Because of its good convergence property and easiness on implementation, DE has been one of the most widely used EAs and applied in solving many real-world problems.

DE generates new solutions through mutating individuals with the difference vector among several individuals of the population. In DE, there are several mutation strategies. Different mutation strategies seem suitable to solve different kinds of problems [2], [3]. Besides the mutation strategy, there are three other parameters: the amplification factor—F, the crossover probability—CR, and the population size—NP. The performance of DE is high dependent on the mutation strategies and the values of the three control parameters [4]–[6]. For a given problem, it is difficult for users to find a good choice of the four elements in DE. In order to find a good choice, the users need to spend plenty of time fine-tuning the parameters in DE.Therefore,several adaptive DE algorithms were proposed to reduce the sensitivity of DE to these parameters.

FADE, a fuzzy adaptive DE proposed by Liu and Lampinen [7], adapts the control parameters F and CR by fuzzy logic controllers. Qin and Suganthan proposed a self-adaptive DE (SaDE), in which the two control parameters F and CR are adapted according to the historic successful information. DESAP [8]–[10] evolves F, CR and NP with individuals. F, CR and NP can become better as individuals become better. Brest et al. [11] proposed an adaptive DE, called jDE, in which different individuals have different control parameters F and CR. The value of F and CR are randomly changed during the evolution. jDE-2 [12] is a further extended version of jDE. Besides adapting F and CR, jDE-2 can adapt the choice of two mutation strategies. JADE [13] is a very competitive DE, in which F and CR are evolved according to their historic successful records.

In these adaptive DE algorithms, the amplification factor of the difference vector—F is adapted at the level of population or individual. An individual has a same value of F in all dimensions. It was shown in [14] that the population may have different convergence characteristics in different dimensions. Individuals may be very similar to each other in some dimensions, but they may have obvious difference in other dimensions. If an individual has different values of F in different dimensions, DE may have a better performance. Based on the work in [15], a dimensional mutation strategy is studied in this paper.

This paper is structured as follows. Section II introduces the jDE algorithm and the dimensional mutation strategy proposed in this paper. Section III provides the experimental study on the improved jDE algorithm. Section IV provides several final remarks.

II. DE with Improved Scheme of Adapting F

In this section, the method of adapting F in the jDE algorithm [11] is introduced, and a dimensional mutation strategy is proposed.

A. Adaptation of F in jDE

At the G-th generation, with respect to an individual $\mathbf{x}_{i,G}$, a mutant vector $\mathbf{v}_{i,G}$, called target vector, is generated by the mutation operation of DE. For each target vector $\mathbf{x}_{i,G}$ with D dimensions, its associated mutant vector $\mathbf{v}_{i,G} = \{v_{i,1,G}, v_{i,2,G}, \ldots, v_{i,D,G}\}$ can be generated via a certain mutation strategy.

In the jDE algorithm, besides the decision variables, each individual has the parameters F and CR. Fig. 1 presents the encoding of individuals in the jDE algorithm. F and CR can be evolved with individuals. However, F and CR are adapted at the individual level. The diversity of F and CR can improve the performance of DE. In the original jDE, the mutation strategy is rand/1/bin, which is described as follows.

$$\mathbf{v}_{i,G} = \mathbf{x}_{r_1,G} + F_{i,G} \cdot (\mathbf{x}_{r_2,G} - \mathbf{x}_{r_3,G}). \quad (1)$$

978-1-5090-0623-6/16/$31.00

$x_{1,1,G}$	$x_{1,2,G}$	...	$x_{1,D,G}$	$F_{1,G}$	$CR_{1,G}$
$x_{2,1,G}$	$x_{2,2,G}$	...	$x_{2,D,G}$	$F_{2,G}$	$CR_{2,G}$
...	...	...	...	...	...
$x_{NP,1,G}$	$x_{NP,2,G}$	...	$x_{NP,D,G}$	$F_{NP,G}$	$CR_{NP,G}$

Fig. 1. The encoding of individuals in the jDE algorithm with NP individuals.

The indices r_1, r_2, r_3 are different integers randomly generated within the range $[1, NP]$. r_1, r_2, r_3 are also different from index i. These indices are randomly generated for each mutant vector. In Eq. (1), different individuals have different values of F. An individual has the same value of F in all dimensions. In the jDE algorithm, the control parameters $F_{i,G+1}$ is adapted as follows:

$$F_{i,G+1} = \begin{cases} F_l + rand_1 \cdot F_u & \text{if } rand_2 < \tau_1 \quad (2a) \\ F_{i,G} & \text{otherwise} \quad (2b) \end{cases}$$

$rand_1$ and $rand_2$ are uniformly distributed real numbers within the range $[0, 1]$. τ_1 represents a probability to adjust F. F_l=0.1, F_u=0.9, and τ_1=0.1 is recommended for a general problem [11]. It can seen in Eq. (2) that the value range of F is [0.1,1.0].

It was shown in [14] that the population may have different convergence characteristics at different dimensions. Take the mutation operation of jDE for example. In Fig. 2, the value of target vector $\mathbf{x}_{r_1}$ is same with the global optimum $\mathbf{x}_{best}$ in the second dimension, while it is far away from $\mathbf{x}_{best}$ in the first dimension. Suppose that $\mathbf{v}_i$ is the mutant vector generated by jDE for $\mathbf{x}_{r_1}$. Comparing the distance of $\mathbf{x}_{r_1}$ to the global optimum before and after the mutation at the dimensional level, it can be seen that although $\mathbf{v}_i$ moves closer to the global optimum in the first dimension, it moves away from the global optimum in the second dimension. This results from the same value of F in all dimensions. If the value of F at the first dimension is large and the value of F at the second dimension is small, the algorithm would be able to exploit the global optimum more efficiently.

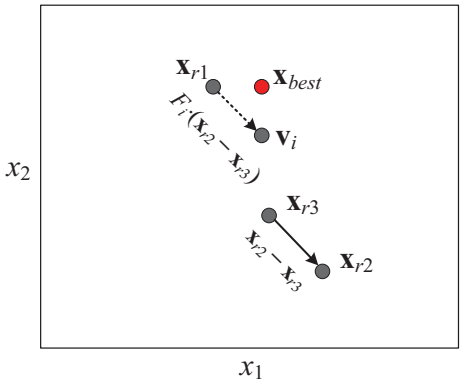


Fig. 2. The mutation operation of jDE.

B. Dimensional-Level Adaptation of F

To adapt the values of F at the dimensional level, the encoding of individuals in jDE is developed. Each individual has D values of F. That is an individual can have different values of F in different dimensions (see Fig. 3). Compared with the encoding shown in Fig. 1, the new encoding of individuals applies F at the levels of individual and dimension. The mutation in jDE (see Eq. (1)) is changed as follows:

$$\mathbf{v}_{i,G} = \mathbf{x}_{r_1,G} + \mathbf{F}_{i,G} \cdot (\mathbf{x}_{r_2,G} - \mathbf{x}_{r_3,G}), \quad (3)$$

where $\mathbf{F}_{i,G} = \{F_{i,1,G}, F_{i,2,G}, \ldots, F_{i,D,G}\}$. In Eq. (3), for an individual, the values of F are independent in different dimensions. The different values of $F_{i,j,G}$ can result in different changes of $x_{r_1,j,G}$ in different dimensions.

$x_{1,1,G}$	$x_{1,2,G}$	...	$x_{1,D,G}$	$F_{1,1,G}$	...	$F_{1,D,G}$	$CR_{1,G}$
$x_{2,1,G}$	$x_{2,2,G}$	...	$x_{2,D,G}$	$F_{2,1,G}$	...	$F_{2,D,G}$	$CR_{2,G}$
...	...	...	...	...	...	...	...
$x_{NP,1,G}$	$x_{NP,2,G}$	...	$x_{NP,D,G}$	$F_{NP,1,G}$	...	$F_{NP,D,G}$	$CR_{NP,G}$

Fig. 3. The new individual encoding.

Fig. 4 illustrates the improvement of the dimensional-level scheme of F. In Fig. 2, the potential mutation space for $\mathbf{x}_{r1}$ is just on the direction of the mutant vector $\mathbf{v}_i$. Compared with Fig. 2, the potential mutation space in the proposed scheme is much larger than that of the jDE algorithm, where is shown as the filled square in Fig. 4. This mutation scheme would improve the exploration ability for each individual as it explores an area rather than on one direction in the search space. This would be particularly helpful to explore the global optimum in high-dimensional search space.

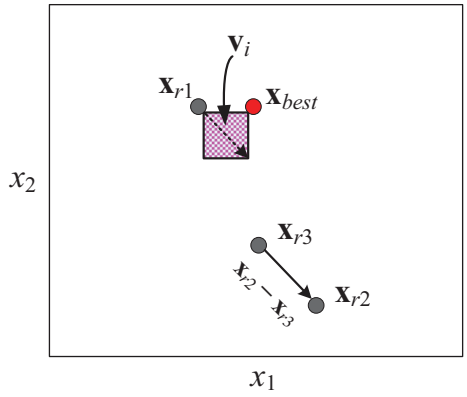


Fig. 4. The improved mutation operation in 2-dimensional space.

The vector of control parameter $\mathbf{F}_{i,G+1}$ is adapted in the following way.

$$\mathbf{F}_{i,G+1} = \begin{cases} \mathbf{F}'_{i,G+1} & \text{if } rand_1 < \tau_1 \quad (4a) \\ \{0.5, \ldots, 0.5\} & \text{otherwise} \quad (4b) \end{cases}$$

where

$$\mathbf{F}'_{i,G+1} = \{\theta_j | \theta_j = F_l + rand_2 \cdot F_u, j = 1, 2, \ldots, D\}. \quad (5)$$

In Eq. (5), θ_j is the value of F in the j-th dimension and generated as jDE [11] (see Eq. (2a)). $rand_1$, $rand_2$ and τ_1 keep same in jDE [11] (see Eq. (2)). Compared with Eq. (2), the diversity of the values of F is improved by the new adaptation. Therefore, the exploration ability of DE algorithms is enhanced.

C. jDE with Dimensional-level of Adaptation

The modified jDE algorithm, called Dim-jDE, applies the new adaptation scheme for F (see Section II-B). Algorithm 1 presents the pseudo code of Dim-jDE. Compared with the original jDE algorithm, the Dim-jDE algorithm adapts F at the levels of individual dimension. Although the proposed scheme runs in $O(D \times NP)$, it does not increase the fitness evaluations compared with the jDE. In addition, it is triggered with a small probability. The control parameter $CR_{i,G+1}$ is adapted as follows, which is the same as jDE [11].

$$CR_{i,G+1} = \begin{cases} rand_3 & \text{if } rand_4 < \tau_2 \\ CR_{i,G} & \text{otherwise} \end{cases} \quad (6)$$

$rand_3, rand_4$ are uniformly distributed random numbers within [0,1]. The value of τ_2 is set to 0.1 [11]. The new CR takes a value from [0,1].

If the value of $u_{i,j}$ is out of the search range after the mutation, $u_{i,j}$ is mapped into the search range as follows.

$$u_{i,j} = \begin{cases} (F_{max} \cdot x_{up,j} - x_{low,j} + u_{i,j})/F_{max} & \text{if } u_{i,j} < x_{low,j} \\ (F_{max} \cdot x_{low,j} - x_{up,j} + u_{i,j})/F_{max} & \text{if } u_{i,j} > x_{up,j} \end{cases} \quad (7)$$

where F_{max} is the maximum value of $F_{i,j,G}$; $x_{low,j}$ and $x_{up,j}$ are the low and up bounds. In this paper, for all algorithms, this mapping method is used to handle the case that the value of $u_{i,j}$ is out of the search range. For the jDE and Dim-jDE algorithms, F_{max}=1.

Algorithm 1 Dim-jDE Algorithm

```
Generate uniform randomly the initial population P_0;
Evaluate the fitness for each individual in P_0;
G = 1;
while the stop criterion is not satisfied do
  for each individual x_i ∈ P_G do
    Select uniform randomly r_1 ≠ r_2 ≠ r_3 ≠ i;
    j_rand=rndint(1,D);
    for j = 1 to D do
      if rnd(0,1)< CR_i or j == j_rand then
        u_{i,j} = x_{r_1,j} + F_{i,j} · (x_{r_2,j} − x_{r_3,j});
      else
        u_{i,j} = x_{i,j};
      end if
    end for
    if u_{i,j} ∉ [x_{low,j}, x_{up,j}] then
      Use Eq. (7) to map u_{i,j} to be in the search range
      [x_{low,j}, x_{up,j}];
    end if
    Evaluate the offspring u_i;
  end for
  for each individual x_i ∈ P_G do
    if u_i is not worse than x_i then
      x_i = u_i;
    end if
  end for
  Adapt CR_i using Eq. (6);
  Adapt F_{i,j} using Eq. (4);
  G = G + 1;
end while
```

III. Experimental Study

A. The Performance of Dim-jDE

1) Benchmark Functions: The algorithms are tested on a set of 25 IEEE CEC05 functions [16] with dimensionality of D=30 and D=50. Functions f_7 and f_{25} have no boundary constraint. For the two functions, if the individuals are beyond of the initial setting bounds, the individuals are not re-mapped by Eq. (7) for all algorithms.

2) Peer Algorithms: Dim-jDE is compared against DE/rand/1/bin [1] and jDE [11]. For all algorithms and functions, the stop criteria is the maximum number of fitness evaluations (i.e., $10000 \times D$) as suggested in [16]. The parameters' setting of algorithms is as follows.

a) DE/rand/1/bin: F=0.5 and CR=0.9, which is recommended in [1], [3], [17]. NP=100 for D=30 and NP=200 for D=50, which are used in [11], [13], [18].
b) jDE: F_l=0.1, F_u=0.9 and τ_1=τ_2=0.1 as recommended in [11]. NP=100 for D=30 and NP=200 for D=50 as used in [11]. The mutation strategy is DE/rand/1/bin.
c) Dim-jDE: Set the same values of parameters as jDE, except for its population size whose value was set to NP=30.

Note that, we will also compare Dim-jDE to jDE with the same initial population (NP=30) to verify that the improved performance of Dim-jDE is caused by the improved scheme of adapting F.

3) Performance Comparison: Table I and Table II summarize the mean errors over 30 independent runs. b, n, and w is the number of functions on which Dim-jDE performs significantly better, equivalent to and significantly worse than its competitor, respectively.

For the 30-dimensional functions, it can be seen in Table I that Dim-jDE performed best on 7 functions. Dim-jDE significantly outperformed DE/rand/1/bin and jDE with NP=100 on most functions. Dim-jDE and jDE with NP=30 have the same population size, but Dim-jDE performed a little significantly better than jDE due to the dimensional adaptation of F. For the 50-dimensional functions, Dim-jDE performed best on 10 functions and outperformed significantly better than DE/rand/1/bin and jDE with NP=200 (see Table II). This is because a large population size may cause many ineffective moves [14], which would decrease the convergence speed of an algorithm. Dim-jDE performed better than jDE with NP=30 on the 50-dimensional functions. From Table I and Table II, it

TABLE I
MEAN ERRORS ± STANDARD DEVIATIONS ON THE 30-DIMENSIONAL CEC05 FUNCTIONS OVER 30 INDEPENDENT RUNS. THE BEST VALUE IS SHOWN IN THE BOLD FONT.

F	Dim-jDE	jDE (NP=30)	jDE (NP=100)	DE/rand/1/bin
f_1	**0.00e+000±0.00e+000**	**0.00e+000±0.00e+000**	1.89e-015±1.04e-014	**0.00e+000±0.00e+000**
f_2	3.60e-008±7.62e-008	**5.99e-009±9.96e-009**‡	6.74e+000±4.08e+000†	1.90e-004±1.96e-004†
f_3	**0.00e+000±0.00e+000**	**0.00e+000±0.00e+000**	1.89e-015±1.04e-014	**0.00e+000±0.00e+000**
f_4	**3.77e-002±7.54e-002**	8.70e+000±4.67e+001†	1.13e+002±7.62e+001†	4.10e-002±3.14e-002
f_5	**3.67e-001±5.91e-001**	4.56e-001±9.53e-001	6.68e+001±2.19e+001†	3.84e-001±4.28e-001
f_6	**1.69e-001±5.58e-001**	2.66e-001±1.01e+000†	1.16e+001±1.01e+000†	3.25e+000±1.36e+000†
f_7	3.29e-004±1.80e-003	3.61e-003±7.25e-003	2.47e-004±1.35e-003‡	**0.00e+000±0.00e+000**‡
f_8	2.10e+001±4.63e-002	2.09e+001±5.42e-002	2.10e+001±3.53e-002	**2.09e+001±5.85e-002**
f_9	2.32e-001±5.01e-001	9.62e-001±1.18e+000†	**1.16e-004±3.81e-004**	1.27e+002±2.04e+001†
f_{10}	1.06e+002±1.12e+001	**4.66e+001±9.95e+000**‡	1.38e+002±8.76e+000†	1.83e+002±1.00e+001†
f_{11}	3.21e+001±1.68e+000	**1.82e+001±7.74e+000**‡	3.35e+001±1.47e+000†	3.91e+001±1.30e+000†
f_{12}	3.17e+004±1.74e+004	**1.84e+003±3.41e+003**‡	3.90e+004±9.72e+003	4.96e+003±4.75e+003‡
f_{13}	1.88e+000±3.43e-001	1.45e+000±5.86e-001‡	2.08e+000±5.28e-001	**1.26e+000±2.94e-001**‡
f_{14}	1.32e+001±1.82e-001	**1.31e+001±2.10e-001**‡	1.32e+001±2.07e-001	1.35e+001±1.36e-001†
f_{15}	2.53e+002±2.58e+002	5.58e+002±2.58e+002†	**1.00e+002±3.92e-013**‡	1.20e+002±1.09e+002
f_{16}	**1.00e+002±2.89e-014**	2.34e+002±2.47e+002†	1.00e+002±4.47e-013†	1.00e+002±4.38e-013†
f_{17}	1.42e+002±1.52e+002	3.69e+002±2.96e+002†	4.55e+002±2.51e+002†	**1.12e+002±6.30e+000**‡
f_{18}	**3.00e+002±6.06e-013**	3.90e+002±2.33e+002	3.00e+002±1.18e-012†	3.00e+002±2.46e-012†
f_{19}	3.00e+002±6.55e-013	**3.00e+002±7.87e-013**‡	3.00e+002±1.06e-012†	3.00e+002±1.51e-012†
f_{20}	4.89e+002±3.20e+002	8.62e+002±2.57e+002†	3.00e+002±4.28e-009	**3.00e+002±2.79e-011**
f_{21}	1.15e+003±1.09e+001	1.16e+003±1.35e+001†	1.15e+003±1.18e+001†	**1.14e+003±6.35e+000**‡
f_{22}	1.15e+003±1.63e+002	1.16e+003±2.70e+001†	1.18e+003±1.45e+001	**9.18e+002±3.49e+002**‡
f_{23}	1.20e+003±1.11e+001	1.21e+003±1.19e+001†	1.19e+003±8.39e+000	**1.19e+003±6.21e+000**‡
f_{24}	1.11e+003±5.47e+000	**1.10e+003±1.14e+001**‡	1.11e+003±4.24e+000†	1.12e+003±2.77e+000†
f_{25}	1.24e+003±9.40e+000	1.17e+003±4.66e+001‡	1.25e+003±1.32e+001†	**1.03e+003±4.25e+002**
$b/n/w$	—	10/6/9	13/10/2	10/8/7

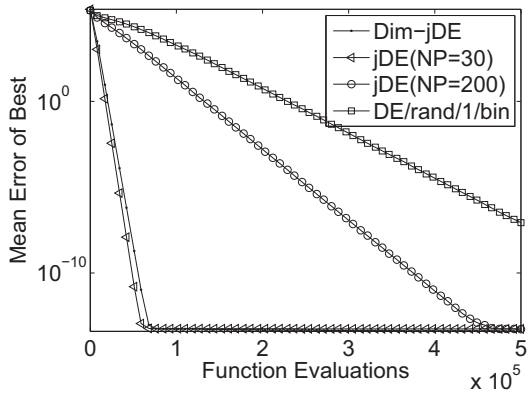
† Dim-jDE performs significantly better than its competitor at a 0.05 level of significance by the Wilcoxon test.
‡ Dim-jDE performs significantly worse than its competitor at a 0.05 level of significance by the Wilcoxon test.

TABLE II
MEAN ERRORS ± STANDARD DEVIATIONS ON THE 50-DIMENSIONAL CEC05 FUNCTIONS OVER 30 INDEPENDENT RUNS. THE BEST VALUE IS SHOWN IN THE BOLD FONT.

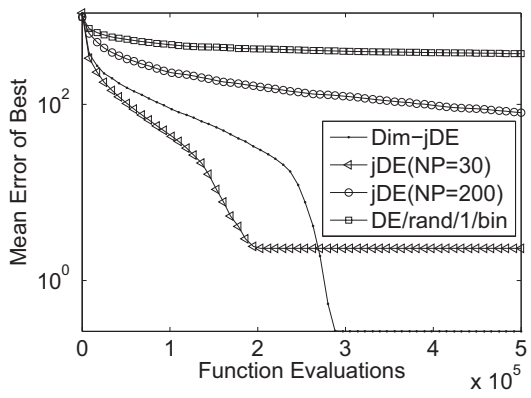
F	Dim-jDE	jDE (NP=30)	jDE (NP=200)	DE/rand/1/bin
f_1	**3.98e-014±2.65e-014**	5.49e-014±1.04e-014†	5.31e-014±1.44e-014	8.59e-008±4.46e-008†
f_2	2.61e-002±2.96e-002	**2.44e-003±2.40e-003**‡	1.18e+004±2.63e+003†	1.03e+004±2.48e+003†
f_3	**4.17e-014±2.56e-014**	5.68e-014±0.00e+000†	7.81e-013±3.63e-013†	3.78e-005±1.97e-005†
f_4	**1.02e+003±4.71e+002**	1.47e+003±1.19e+003	3.46e+004±7.89e+003†	3.40e+004±6.90e+003†
f_5	**1.24e+001±4.47e+000**	3.25e+001±1.13e+001†	4.03e+003±5.28e+002†	6.13e+003±1.14e+003†
f_6	2.07e+001±4.55e+000	**4.53e+000±1.29e+001**‡	4.00e+001±4.12e-001†	4.10e+001±5.43e-001†
f_7	**1.14e+000±3.46e-001**	1.24e+000±3.06e-001	4.90e+000±1.50e+000†	2.42e+000±8.65e-001†
f_8	**2.11e+001±3.91e-002**	2.11e+001±3.08e-002	2.11e+001±3.65e-002	2.11e+001±4.12e-002
f_9	**2.65e-001±5.18e-001**	2.32e+000±1.55e+000†	8.04e+001±5.70e+000†	3.76e+002±1.62e+001†
f_{10}	2.59e+002±2.58e+001	**9.63e+001±1.46e+001**‡	3.30e+002±1.49e+001†	3.92e+002±1.71e+001†
f_{11}	6.10e+001±1.61e+000	**4.82e+001±1.38e+001**‡	6.64e+001±1.69e+000†	7.28e+001±2.03e+000†
f_{12}	2.29e+005±3.27e+004	**7.36e+003±6.06e+003**‡	4.42e+005±5.63e+004†	1.95e+006±2.67e+005†
f_{13}	**2.91e+000±5.29e-001**	5.13e+000±3.44e+000†	4.60e+000±1.04e+000†	4.90e+000±1.29e+000†
f_{14}	2.29e+001±1.93e-001	**2.28e+001±2.19e-001**	2.31e+001±1.93e-001†	2.32e+001±1.39e-001†
f_{15}	6.30e+002±2.32e+002	8.69e+002±9.00e+001†	5.47e+002±2.21e+002	**4.10e+002±3.05e+001**‡
f_{16}	5.29e+002±2.91e+002	8.10e+002±8.73e+001†	**1.37e+002±1.48e+002**‡	1.51e+002±1.14e+002‡
f_{17}	**8.17e+002±9.34e+001**	8.84e+002±7.72e+001†	9.68e+002±1.96e+001†	1.01e+003±1.78e+001†
f_{18}	3.69e+002±2.12e+002	1.04e+003±3.43e+001†	**3.00e+002±1.10e-002**‡	3.01e+002±5.90e-001‡
f_{19}	3.20e+002±1.09e+002	1.02e+003±3.28e+001†	**3.00e+002±1.90e-002**‡	3.01e+002±9.89e-001‡
f_{20}	3.28e+002±1.52e+002	1.03e+003±1.42e+002†	**3.00e+002±1.35e-002**‡	3.01e+002±8.54e-001‡
f_{21}	**1.11e+003±1.18e+001**	1.13e+003±8.03e+000†	1.14e+003±1.28e+001†	1.13e+003±1.52e+001†
f_{22}	1.13e+003±8.24e+000	**1.13e+003±9.10e+000**	1.13e+003±6.84e+000	1.14e+003±6.03e+000†
f_{23}	1.16e+003±1.23e+001	1.16e+003±7.50e+000	1.16e+003±8.91e+000	**1.16e+003±7.40e+000**
f_{24}	1.19e+003±4.66e+000	**1.18e+003±5.10e+000**‡	1.19e+003±3.91e+000†	1.20e+003±3.24e+000†
f_{25}	1.32e+003±8.74e+000	**1.23e+003±2.87e+001**‡	1.33e+003±5.22e+000†	1.35e+003±6.04e+000†
$b/n/w$	—	12/6/7	16/5/4	18/2/5

† Dim-jDE performs significantly better than its competitor at a 0.05 level of significance by the Wilcoxon test.
‡ Dim-jDE performs significantly worse than its competitor at a 0.05 level of significance by the Wilcoxon test.

can be seen that the performance of Dim-jDE is more superior to other peer algorithms as the dimensionality of functions



(a) f_1



(b) f_9

Fig. 5. Convergence graph for the 50-dimensional f_1 and f_9.

becomes higher.

Fig. 5 shows the convergence behaviors of algorithms on two typical 50-dimensional functions (the unimodal function f_1 and the multimodal function f_9) over 30 independent runs. It can be seen in Fig. 5a that Dim-jDE and jDE with NP=30 converge much faster than other algorithms on f_1. This is because a small population size can speed up the convergence of algorithms, while a large population size will slow down the convergence speed. This is also can be seen on the multimodal function f_9 (see Fig. 5b). Although the initial populations of Dim-jDE and jDE with NP=30 are the same, due to the high diversity of F, Dim-jDE can avoid to be trapped in a local optimum, while the population of jDE with NP=30 is trapped in a local optimum.

B. The Advantage of Dim-jDE

The working mechanism of dimensional adaptation of F is studied in this section. Take a 2-dimensional multimodal *Schwefel* function [19] for example.

$$f(\mathbf{x}) = \sum_{i=1}^{D} \left(-x_i \sin\left(\sqrt{|x_i|}\right)\right), x_i \in [-500, 500], \qquad (8)$$

D=2 here. The global optimum of the *Schwefel* function is $\mathbf{x} \approx$\{420.9687,. . .,420.9687\}, which is shown as the pentagram point in the figure.

Fig. 6 shows a typical run of jDE [11] and Dim-jDE (see Algorithm 1) algorithms for the 2-dimensional *Schwefel* function with NP=4 and $MaxFEs$=20000. Note that, the initial population of jDE and Dim-jDE algorithms are the same. In Fig. 6a, for the individual $\mathbf{x}_2$, its trail vector generated by jDE is as follows:

$$\mathbf{v}_2 = \mathbf{x}_1 + F_2 \cdot (\mathbf{x}_4 - \mathbf{x}_3),$$

where F_2=0.287 generated by Eq. (2). Although $\mathbf{v}_2$ is closer to the global optimum than $\mathbf{x}_2$, it is further away from the global optimum than $\mathbf{x}_2$ at the dimension x_2. This is because that for all the dimensions, the values of F are same. If the value of F at the dimension x_2 is very small, $\mathbf{v}_2$ could be more closer to the global optimum. In Fig. 6b, for the individual $\mathbf{x}_1$, its trail vector generated by Dim-jDE is as follows:

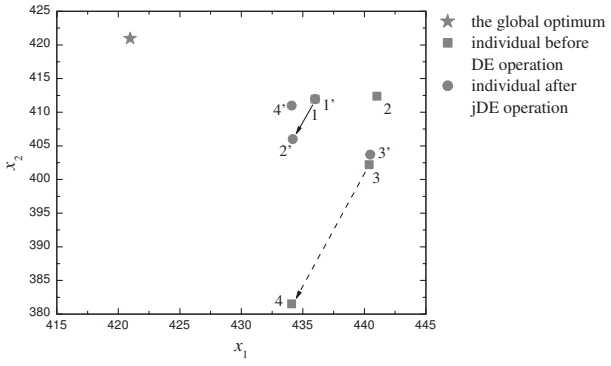
$$\mathbf{v}_1 = \mathbf{x}_4 + \mathbf{F_1} \cdot (\mathbf{x}_2 - \mathbf{x}_3),$$

where $\mathbf{F_1} = \{0.886, 0.523\}$ generated by Eq. (4). The individual, which is shown as the triangle point, is assumed to generate by jDE when the values of F at both dimensions are equal to 0.886. Because for the two dimensions the values of F of Dim-jDE are different, the individual generated by Dim-jDE moves a smaller step than the one generated by jDE at the dimension x_1. Thus, the individual generated by Dim-jDE can be more closer to the global optimum. The diversity of F can improve the explorative capability of Dim-jDE algorithm. In the Fig. 6c, it can be seen that jDE traps in a local optimum, but Dim-jDE can found the global optimum.

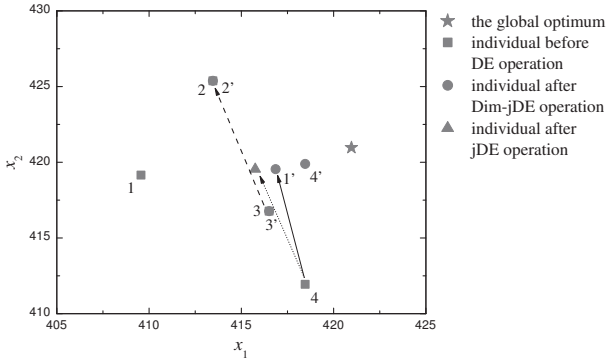
IV. Conclusions

The performance of DE is high dependent on the mutation strategies and the values of the control parameters. Although several adaptive DE algorithms have been proposed, the amplification factor of the difference vector—F is adapted at the levels of population and individual. An individual has a same value of F in all dimensions. The population may have different convergence characteristics in different dimensions. Individuals may be very similar to each other in some dimensions, but they may have obvious difference in other dimensions. In this paper, a dimensional mutation is studied and proposed for DE.

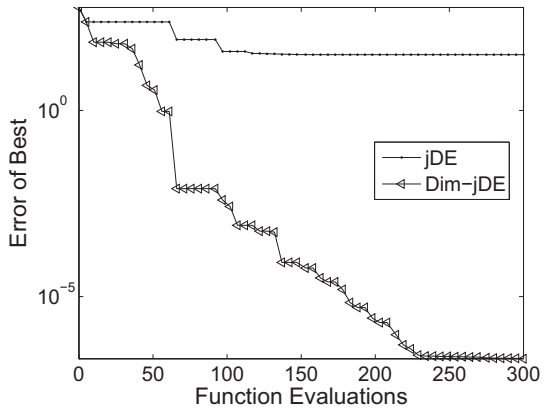
In the dimensional mutation, an individual has D values of F. That is an individual can have different values of F in different dimensions. This new dimensional mutation can enhance the exploration ability of DE. The new adaptation is applied into the jDE algorithm and tested on the CEC05 functions. The experimental results suggest that the dimensional mutation can make a better performance of jDE. The high diversity of F can increase the probability that jDE jumps out of a local optimum.



(a) jDE



(b) Dim-jDE



(c) convergence

Fig. 6. A typical run of jDE and Dim-jDE algorithms on the 2-dimensional *Schwefel* function with NP=4: (a) the mutation of jDE at the 29-th generation, (b) the mutation of Dim-jDE at the 13-rd generation, and (c) the convergence graph.

ACKNOWLEDGEMENT

This work is funded by the project of the National Natural Science Foundation of China (Grand No. 61305086).

REFERENCES

[1] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, December 1997.

[2] V. Feoktistov and S. Janaqi, "Generalization of the strategies in differential evolution," in *Proceedings of the 18th International Parallel and Distributed Processing Symposium*, April 2004, pp. 165–170.

[3] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello, "A comparative study of differential evolution variants for global optimization," in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, ser. GECCO2006. New York, NY, USA: ACM, 2006, pp. 485–492.

[4] R. Gämperle, S. D. Müller, and P. Koumoutsakos, "A parameter study for differential evolution," in *WSEAS Int. Conf. on Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*. Press, 2002, pp. 293–298.

[5] J. Liu and J. Lampinen, "On setting the control parameter of the differential evolution method," in *Proc. 8th Int. Conf. Soft Computing*, 2002, pp. 11–18.

[6] R. Mallipeddi and P. Suganthan, "Empirical study on the effect of population size on differential evolution algorithm," in *IEEE Congress on Evolutionary Computation'08*, june 2008, pp. 3663–3670.

[7] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 9, no. 6, pp. 448–462, 2005.

[8] J. Teo, "Differential evolution with self-adaptive populations," in *Knowledge-Based Intelligent Information and Engineering Systems - 9th International Conference, KES 2005*, Melbourne, Australia, 2005, pp. 1284–1290.

[9] ——, "Exploring dynamic self-adaptive populations in differential evolution," *Soft Comput.: Fusion Found., Methodologies Applicat.*, vol. 10, no. 8, pp. 673–686, 2006.

[10] N. Teng, J. Teo, and M. Hijazi, "Self-adaptive population sizing for a tune-free differential evolution," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 13, pp. 709–724, 2009.

[11] 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 Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[12] J. Brest, B. Boskovic, V. Z. S. Greiner, and M. Maucec, "Performance comparison of self-adaptive and adaptive differential evolution algorithms," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 11, no. 7, pp. 617–629, 2007.

[13] J. Zhang and C. Arthur, "JADE: Adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

[14] M. Yang, C. Li, Z. Cai, and J. Guan, "Differential evolution with auto-enhanced population diversity," *Cybernetics, IEEE Transactions on*, vol. 45, no. 2, pp. 302–315, Feb 2015.

[15] M. Yang, J. Guan, Z. Cai, and C. Li, "A dimensional-level adaptive differential evolutionary algorithm for continuous optimization," in *Proceedings of the 2014 conference companion on Genetic and evolutionary computation companion*. ACM, 2014, pp. 123–124.

[16] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization," Nanyang Technological University, Singapore, Tech. Rep., 2005.

[17] J. Vesterstrom, R. Thomsen, B. R. Center, A. Univ., and Denmark, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in *Proc. IEEE Congr. Evol. Comput.*, ser. CEC2004, 2004, pp. 1980–1987.

[18] A. Qin and P. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2005, pp. 1785–1791.

[19] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.