International Journal of Computational Intelligence and Applications
Vol. 10, No. 1 (2011) 35–65

DOI: 10.1142/S1469026811002982

Imperial College Press
www.icpress.co.uk

A GENERALIZED HYBRID GENERATION SCHEME OF DIFFERENTIAL EVOLUTION FOR GLOBAL NUMERICAL OPTIMIZATION*

WENYIN GONG[†,‡,¶], ZHIHUA CAI[†,‖], LIYUAN JIA[§] and HUI LI[†]

[†]*School of Computer Science*
China University of Geosciences
Wuhan 430074, P. R. China

[‡]*State Key Laboratory of Software Engineering*
Wuhan University, 430072, P. R. China

[§]*Department of Computer Science*
Hunan City University
Yiyang Hunan 413000, P. R. China
[¶]*cug11100304@yahoo.com.cn*
[‖]*zhcai@cug.edu.dn*

Received 8 August 2010
Revised 16 December 2010

Differential evolution (DE) is a simple yet powerful evolutionary algorithm for global numerical optimization over continuous domain, which has been widely used in many areas. Although DE is good at exploring the search space, it is slow at the exploitation of the solutions. To alleviate this drawback, in this paper, we propose a generalized hybrid generation scheme, which attempts to enhance the exploitation and accelerate the convergence velocity of the original DE algorithm. In the hybrid generation scheme the operator with powerful exploitation is hybridized with the original DE operator. In addition, a self-adaptive exploitation factor is introduced to control the frequency of the exploitation operation. In order to evaluate the performance of our proposed generation scheme, two operators, the migration operator of biogeography-based optimization and the "DE/best/1" mutation operator, are employed as the exploitation operator. Moreover, 23 benchmark functions (including 10 test functions provided by CEC2005 special session) are chosen from the literature as the test suite. Experimental results confirm that the new hybrid generation scheme is able to enhance the exploitation of the original DE algorithm and speed up its convergence rate.

Keywords: Differential evolution; global numerical optimization; exploitation; exploration; self-adaptive control parameter.

*This paper is based on "Hybrid Differential Evolution for Global Numerical Optimization" by L. Jia, L. Li, W. Gong and L. Huang, which appeared in Proceedings of RSKT 2010, Springer-Verlag. 2010.10, LNAI 6401, Beijing, China, pp. 560–567.

1. Introduction

Differential Evolution (DE), proposed by Storn and Price,[45] is a simple yet powerful population-based, direct search algorithm with the generation-and-test feature for global optimization using real-valued parameters. In DE, the mutation operator is based on the distribution of solutions in the current population. New candidates are created by combining the parent solution and the mutant. A candidate replaces the parent only if it has a better fitness value. In Ref. 45, Price and Storn gave the working principle of DE with single mutation strategy. Later on, they suggested 10 different mutation strategies of DE.[46,37] Among DE's advantages are its simple structure, ease of use, speed and robustness. Due to these advantages, it has many real-world applications, such as data mining,[1,9] IIR design,[10] neural network training,[18] pattern recognition, digital filter design, engineering design, etc.[37,16,8] Most recently, DE has also been used for the global permutation-based combinatorial optimization problems.[36]

Although the DE algorithm has been widely used in many fields, it has been shown to have certain weaknesses, especially if the global optimum should be located using a limited number of fitness function evaluations (NFFEs). In addition, DE is good at exploring the search space and locating the region of global minimum, but it is slow at the exploitation of the solutions.[33] To remedy this drawback, this paper proposes a new hybrid generation scheme, which attempts to balance the exploration and exploitation of DE. The proposed hybrid generation scheme, which is based on the binomial recombination operator of DE, is hybridized with the exploitative operation. Thus, the new hybrid generation scheme is able to enhance the exploitation of the original DE algorithm. Additionally, a self-adaptive exploitation factor, η, is introduced to control the frequency of the exploitation operation.

The main aim of this paper is to present an alternative generation scheme of DE. This generation scheme attempts to enhance the exploitation and accelerate the convergence rate of the original DE algorithm. To verify the expectation of our proposed generation scheme, two operators, the migration operator of biogeography-based optimization (BBO)[43] and the "DE/best/1" mutation operator, are employed as the exploitation operator. Moreover, 23 benchmark functions (including 10 new test functions provided by CEC2005 special session[47]) are chosen from the literature as the test functions. Experimental results confirm that the new hybrid generation scheme is able to enhance the exploitation and accelerate the convergence speed of the original DE algorithm. To increase the experiment confidence, DE with the new generation scheme is also compared with the original DE algorithm, the adaptive hill-climbing SPX-based DE method,[33] and the opposition-based DE method.[40] The results show that our approach performs better, or at least comparably, in terms of the quality of the final solutions and the convergence rate.

The rest of this paper is organized as follows. Section 2 briefly describes the DE and BBO algorithms. In Sec. 3, some related works of DE are presented. Our proposed approach is presented in detail in Sec. 4. In Sec. 5, we verify our approach through 23 benchmark functions. Moreover, the experimental results are compared with several other DE approaches. The last section, Sec. 6, is devoted to conclusions and future work.

2. DE and BBO

In this work, we consider the following global numerical optimization problem:

$$\text{Minimize } f(\vec{x}), \quad \vec{x} \in S \tag{1}$$

where $S \subseteq R^D$ is a compact set, $\vec{x} = [x_1, \ldots, x_D]^{\mathrm{T}}$, and D is the dimension of the decision variables. Generally speaking, for each variable x_i it satisfies a constrained boundary

$$L_i \leq x_i \leq U_i, \quad i = 1, \ldots, D \tag{2}$$

In global numerical optimization problems, the major challenge is that an algorithm may be trapped in the local optima of the objective function. This issue is particularly challenging when the dimension is high. Recently, using the Evolutionary Computation (EC)[4] to solve the global optimization has been very active, producing different kinds of EC for optimization in the continuous domain, such as genetic algorithms,[3,24] evolutionary programming,[52] evolution strategy,[13] particle swarm optimization,[28] immune clonal algorithm,[20] differential evolution,[45] etc.

2.1. *Differential evolution*

The DE algorithm[45] is a simple evolutionary algorithm (EA)[4] for global numerical optimization. It creates new candidate solutions by combining the parent individual and several other individuals of the same population. A candidate replaces the parent only if it has a better fitness value. This is a rather greedy selection scheme that often outperforms traditional EAs. The pseudo-code of the original DE algorithm is shown in Algorithm 1. D is the number of decision variables. NP is the population size. F is the mutation scaling factor. CR is the probability of crossover operator. $x_{i,j}$ is the jth variable of the solution $\vec{x}_i$. $\vec{u}_i$ is the offspring. rndint$(1, D)$ is a uniformly distributed random integer number between 1 and D. rndreal$_j[0, 1)$ is a uniformly distributed random real number in [0, 1); it is generated anew for each value of j. Many schemes to create a candidate are available. In Algorithm 1, the classic "DE/rand/1/bin" scheme is adopted (see lines 6–13 of Algorithm 1). More details on "DE/rand/1/bin" and other DE schemes can be found in Refs. 46 and 37.

Algorithm 1. DE with DE/rand/1/bin scheme.

```
1: Generate the initial population
2: Evaluate the fitness for each individual
3: while The halting criterion is not satisfied do
4:    for i = 1 to NP do
5:       Select uniform randomly r1 ≠ r2 ≠ r3 ≠ i
6.       j_rand = rndint(1, D)
7:       for j = 1 to D do
8:          if rndreal_j[0, 1) < CR or j == j_rand then
9:             u_{i,j} = v_{i,j} = x_{r1,j} + F · (x_{r2,j} − x_{r3,j})
10:         else
11:            u_{i,j} = x_i, j
12:         end if
13:      end for
14:   end for
15:   for i − 1 to NP do
16:      Evaluate the offspring u_i,
17:      if u_i is better than x_i then
18:            Replace x_i, with u_i
19:      end if
20:   end for
21: end while
```

From Algorithm 1, we can see that there are only three control parameters in this algorithm. These are NP, F, and CR. As for the terminal conditions, we can either fix the maximum NFFEs Max_NFFEs or the precision of a desired solution VTR (value to reach).

In the original DE algorithm, many schemes have been proposed[46,37] that use different learning strategies and/or recombination operations in the reproduction stage. In order to distinguish among its schemes, the notation "DE/a/b/c" is used, where "DE" denotes the DE algorithm; "a" specifies the vector to be mutated (which can be random or the best vector); "b" is the number of difference vectors used; and "c" denotes the crossover scheme, binomial or exponential. In line 9 of Algorithm 1, the mutation strategy is called "DE/rand/1", which is a classic strategy of DE.[37] Other well-known mutation strategies are listed as follows.

(i) "DE/best/1":

$$\vec{v}_i = \vec{x}_{\text{best}} + F(\vec{x}_{r1} - \vec{x}_{r2}). \tag{3}$$

(ii) "DE/best/2":

$$\vec{v}_i = \vec{x}_{\text{best}} + F(\vec{x}_{r1} - \vec{x}_{r2}) + F(\vec{x}_{r3} - \vec{x}_{r4}). \tag{4}$$

(iii) "DE/rand/2":

$$\vec{v}_i = \vec{x}_{r1} + F(\vec{x}_{r2} - \vec{x}_{r3}) + F(\vec{x}_{r4} - \vec{x}_{r5}). \tag{5}$$

(iv) "DE/current-to-best/1":

$$\vec{v}_i = \vec{x}_i + F(\vec{x}_{\text{best}} - \vec{x}_i) + F(\vec{x}_{r1} - \vec{x}_{r2}), \tag{6}$$

where $\vec{x}_{\text{best}}$ represents the best individual in the current generation, $r1, \ldots, r5 \in \{1, \ldots, \text{NP}\}$, and $r1 \neq r2 \neq r3 \neq r4 \neq r5 \neq i$.

2.2. *Biogeography-based optimization*

Since we will use the migration operator of BBO as the exploitation operator in this work, the BBO algorithm will briefly be introduced in this section. BBO[43] is a new population-based, biogeography inspired global optimization algorithm. BBO is a generalization of biogeography to EA. In BBO, it is modeled after the immigration and emigration of species between islands in search of more friendly habitats. The islands represent the solutions and they are ranked by their island suitability index (ISI), where a higher ISI signifies a superior fitness value. The islands are comprised of solution features named suitability index variables (SIV), equivalent to GA's genes.

In BBO, the kth ($k = 1, \ldots, \text{NP}$) individual has its own immigration rate λ_k and emigration rate μ_k. A good solution has higher λ_k and lower μ_k, vice versa. The immigration rate and the emigration rate are functions of the number of species in the habitat. They can be calculated as follows:

$$\lambda_k = I\left(1 - \frac{k}{n}\right), \tag{7}$$

$$\mu_k = E\left(\frac{k}{n}\right), \tag{8}$$

where I is the maximum possible immigration rate; E is the maximum possible emigration rate; k is the number of species of the kth individual; and $n = \text{NP}$ is the maximum number of species. Note that Eqs. (8) and (9) are just one method for calculating λ_k and μ_k. There are other different options to assign them based on different specie models.[43]

There are two main operators in BBO, the migration and the mutation. One option for implementing the migration operator can be described in Algorithm 2,[a] where rndreal$(0, 1)$ is a uniformly distributed random real number in $(0, 1)$. With the migration operator, BBO can share the information among solutions. Poor solutions tend to accept more useful information from good solutions. This makes BBO good at exploiting the information in the current population. More details about the two operators can be found in Ref. 43 and in the Matlab code.[44]

[a] Since the mutation operator of BBO is not used in our approach, we do not describe it here. Interested readers can refer to Refs. 43 and 44.

Algorithm 2. Habitat migration of BBO.

```
for i = 1 to NP do
    Select x_i with probability ∝ λ_i
    if rndreal(0, 1) < λ_i then
        for j = 1 to NP do
            Select x_j with probability ∝ μ_j
            if rndreal(0, 1) < μ_j then
                Randomly select a variable σ from x_j
                Replace the corresponding variable in x_j with σ
            end if
        end for
    end if
end for
```

3. Related Work to DE

Several previous researches pointed out that there are three main drawbacks of the original DE algorithm. First, the parameters of DE are problem dependent and the choice of them is often critical for the performance of DE.[19,29] Second, choosing the best among different mutation strategies available for DE is also not easy for a specific problem.[39,38] Third, DE is good at exploring the search space and locating the region of global minimum, but it is slow at the exploitation of the solution.[33] Due to these drawbacks, many researchers are now working on the improvement of DE, and many variants are presented.

Adapting the DE's control parameters is one possible improvement. Zaharie[53] proposed a parameter adaptation for DE based on the idea of controlling the population diversity. Zaharie also implemented a multi-population approach of the proposed adaptive DE. Liu and Lampinen[29] proposed a Fuzzy Adaptive DE (FADE), which employs fuzzy logic controllers to adapt the search parameters for the mutation operation and crossover operation. Das *et al.*[11] proposed two variants of DE, DERSF and DETVSF, that use varying scaling factors. Brest *et al.*[6] proposed self-adapting control parameter settings of DE (jDE). Their proposed approach encodes the F and CR parameters into the chromosome and uses a self-adaptive control mechanism to change them. Based on jDE, Brest *et al.* proposed an improved self-adaptive DE (jDE-2).[5] The jDE-2 algorithm uses two strategies "DE/rand/1/bin" and "DE/current-to-best/1/bin" as the candidate strategies. Each strategy has its own control parameters F and CR. These parameters are also encoded into the chromosome like jDE. In Ref. 5, detailed performance comparisons of some adaptive or self-adaptive DE algorithms on the benchmark functions were studied. Salman *et al.*[41] proposed a self-adaptive DE (SDE) algorithm that eliminates the need for manual tuning of control parameters. In SDE, the mutation scaling factor

F is self-adapted by a mutation strategy similar to the mutation operator of DE. The crossover rate CR in Ref. 41 is generated for each individual from a normal distribution $N(0.5, 0.15)$. Nobakhti and Wang[32] proposed a Randomized Adaptive DE (RADE) method, where a simple randomized self-adaptive scheme was proposed for the mutation scaling factor F. Neri and Tirronen[31] proposed a scale factor local search DE (SFLSDE). SFLSDE is a DE-based memetic algorithm[35] which used, within the self-adaptive scheme proposed in Ref. 6, two local search algorithms to detect a value of the scaling factor F corresponding to an offspring with a high performance. Teo[49] presented a DE algorithm with a dynamic population sizing strategy, where the population size is self-adapting. Through five De Jong's test functions, they showed that DE with self-adaptive populations produced highly competitive results compared with the original DE algorithm. Brest and Maucec[7] proposed an improved DE method, where the population size is gradually reduced. The authors concluded that their approach improves efficiency and robustness of DE.

As mentioned above, there are many generation schemes of DE, however choosing the best among different mutation strategies available for DE is not easy for a specific problem.[39,38] To have a better choice of DE's strategies, Feoktistov and Janaqi[17] introduced a generalization of DE's strategies. Their approach led to a new universal formula of differentiation. In Ref. 25, Iorio and Li proposed a rotation-invariant strategy "DE/current-to-rand/1" to solve the rotated multi-objective optimization problems. Qin and Suganthan[39] proposed a self-adaptive DE algorithm. The aim of their work was to allow DE to switch between two schemes: "DE/rand/1/bin" and "DE/best/2/bin" and also to adapt the F and CR values. Mezura-Montes *et al.*[30] presented an empirical comparison of the generation schemes of DE. Ali and Fatti[2] proposed a point generation scheme that uses an approximation to the probability distribution of trial points in DE. Recently, Qin *et al.*[38] extended their previous work.[39] In their proposed SaDE approach, four mutation strategies were adopted. Different CR values were also used for different strategies. Their proposed algorithm outperformed the original DE algorithm and some others compared adaptive/self-adaptive DE variants.[38]

Hybridization with other different algorithms is another direction for the improvement of DE. Fan and Lampinen[15] proposed a modified version of DE, where the trigonometric mutation was embedded into the DE algorithm. The modification made the algorithm to get a better trade-off between the convergence rate and the robustness.[15] Zhang and Xie[54] proposed a DEPSO hybrid to solve optimization problems by incorporating the "DE/rand/1/bin" mutation operator inside the PSO algorithm. Sun *et al.*[48] proposed a new hybrid algorithm based on a combination of DE with Estimation of Distribution Algorithm (EDA).[27] This technique uses a probability model to determine promising regions in order to focus the search process on those areas. Gong *et al.*[21] employed the two level orthogonal crossover to improve the performance of DE. Noman and Iba[34] proposed fittest individual refinement, a crossover-based local search DE method to solve the high dimensional

problems. Based on their previous work,[34] they proposed an adaptive hill-climbing SPX-based DE (DEahcSPX).[33] In DEahcSPX, the SPX technique[50] is integrated into DE to solve the optimization problems by adaptively adjusting the length of the search, using a hill-climbing heuristic. Through the experiments, they showed that the proposed new version of DE performs better, or at least comparably, to classic DE algorithm. Kaelo and Ali[26] adopted the attraction–repulsion concept of electromagnetism-like algorithm to boost the mutation operation of DE. Wang *et al.*[51] proposed a dynamic clustering-based DE for global optimization, where a hierarchical clustering method is dynamically incorporated in DE. Rahnamayan *et al.*[40] proposed an opposition-based DE (ODE) which employs opposition-based learning to generate initial population and also for generation jumping. Through a comprehensive set of benchmark functions they showed that replacing the random initialization with the opposition-based population initialization in DE can accelerate the convergence speed.

4. Our Proposed Hybrid Generation Scheme

From the literature reviewed, we can observe that there are many DE variants to remedy some pitfalls of DE mentioned above. In this study, we will propose a novel hybrid generation scheme to enhance the exploitation of the original DE algorithm. Our approach, which is different from the above-mentioned DE variants, is presented in detail as follows.

4.1. *Motivations*

The DE's behavior is determined by the trade-off between the exploration and exploitation. As stated in Ref. 33, DE is good at exploring the search space, but it is slow at the exploitation of the solutions. Recently, hybridization of EAs is getting more popular due to their capabilities in handling several real world problems.[23] Thus, hybridization of DE with another algorithm, which has powerful exploitation, might balance the exploration and exploitation of the DE algorithm. Based on this consideration, we propose a new hybrid generation scheme, where the operator with powerful exploitation is hybridized with the binomial recombination operator of the original DE algorithm.

As we briefly described in Sec. 2.2, the migration operator of BBO has good exploitation; it can share the useful information among individuals. Since the BBO algorithm is biogeography theoretical support, we will first hybridize the migration operator with DE to verify the performance of our proposed hybrid generation scheme. Moreover, the "DE/best/1" mutation strategy usually has fast convergence speed and favors the exploitation of the best individual in the current population. We will also hybridize this strategy with other DE strategy to test the performance of the hybrid generation scheme.

4.2. *Hybrid generation scheme*

The new hybrid generation scheme of DE is based on the binomial recombination operator of the original DE algorithm. The trial vector $\vec{u}_i$ is possibly composed of three parts: the mutant variables generated by the DE mutation, variables generated by the exploitative operation, and the variables inherited from the target vector$\vec{x}_i$. The pseudo-code of the hybrid generation scheme is described in Algorithm 3.

From Algorithm 3, we can observe that:

(i) The proposed hybrid generation scheme is able to enhance the exploitation of DE because of the exploitative operation.
(ii) It is a generalized scheme. For example, in line 6 of Algorithm 3, it is the explorative operation. The mutation strategy of DE, which is good at exploring the search space, can be used here, such as "DE/rand/1", "DE/rand/2", and so on. In line 8, the operator with powerful exploitation can be employed, such as the migration operator of BBO, "DE/best/1", etc.
(iii) Compared with the original DE generation scheme, our proposed scheme is also very simple and easy to implement.
(iv) The hybrid generation scheme maintains the main property of the original DE generation scheme. The crossover rate CR mainly controls the whole generation scheme. If the CR value is higher, the explorative operation plays a more important role, and the exploitative operation has less influence to the offspring. Especially, when CR is close to 1.0, the generation scheme is more rotationally invariant,[37] and hence it can be used to solve the parameter-dependent problems.

Algorithm 3. The new hybrid generation scheme.

1: for $i = 1$ to NP do
2: Select uniform randomly $r_1 \neq r_2 \neq r_3 \neq i$
3: $j_{\text{rand}} = \text{rndint}(1, D)$
4: for $j = 1$ to D do
5: if $\text{rndreal}_j[0, 1) < \text{CR}$ or $j == j_{\text{rand}}$ then
6: $u_{i,j}$ is generated by the mutation strategy of DE {explorative operation}
7: else if $\text{rndreal}_j[0, 1) < \eta$ then
8: $u_{i,j}$ is generated by the operator with powerful exploitation {exploitative operation}
9: else
10: $u_{i,j} = x_{i,j}$
11: end if
12: end for
13: end for

(v) An additional parameter, i.e., the exploitation factor η, is introduced to control the frequency of the exploitative operation. Setting properly of this parameter can make the algorithm get a well-found trade-off between exploration and exploitation. In this work, we will present a self-adaptive strategy to control this parameter next.

4.3. *Self-adaptive exploitation factor*

In our proposed hybrid generation scheme shown in Algorithm 3, we introduced an additional parameter, i.e., exploitation factor η. The parameter is used to control the frequency of the exploitative operation. If η is set higher, it may lead to over-exploitation. The algorithm may trap into the local optimum. Contrarily, if η is lower, it may result in under exploitation. The algorithm cannot exploit the solutions sufficiently.

Self-adaptation[14] is highly beneficial for adjusting control parameters, especially when done without any user interaction.[5] Thus, in this study, we propose a self-adaptive strategy to control the exploitation factor η. This strategy is similar to the parameter control method proposed in Ref. 6. In our proposed method, η is encoded into the chromosome. The ith individual X_i is represented as follows.

$$X_i = \langle \vec{x}_i, \eta_i \rangle = \langle x_1, \ldots, x_D, \eta_i \rangle, \tag{9}$$

where η_i is the exploitation factor of the ith individual; and it is initially randomly generated between 0 and 1. The new parameter is calculated as:

$$\eta_i = \begin{cases} \text{rndreal}[0, \text{dynamic\_gen}], & \text{if rndreal}[0,1] < \delta \\ \eta_i, & \text{otherwise,} \end{cases} \tag{10}$$

where rndreal$[a, b]$ is a uniform random value generated in $[a, b]$. δ indicates the probability to adjust the exploitation factor η_i. It is similar to the parameters τ_1 and τ_2 used in Ref. 6. The effect of δ will be empirically discussed in Sec. 5.6. dynamic_gen is calculated as:

$$\text{dynamic\_gen} = \frac{\text{gen}}{\text{Max\_gen}}, \tag{11}$$

where gen is the current generation, and Max_gen is the maximum generation. The reason of calculating $\eta_i = \text{rndreal}[0, \text{dynamic\_gen}]$ is that at the beginning of the evolutionary process the useful information of the population is less, and hence, exploiting more information might degrade the performance of the algorithm, especially for the multi-modal functions. As the evolution progresses, the population contains more and more useful information, thus the probability of the exploitative operation needs to be increased to utilize the useful information.

4.4. *Handling the boundary constraint of variables*

After using the generation scheme to generate a new solution, if one or more of the variables in the new solution are outside their boundaries, i.e., $x_i \notin [L_i, U_i]$, the

Algorithm 4. DE with the proposed hybrid generation scheme.

Generate the initial population
Evaluate the fitness for each individual
Initialize the current generation counter gen = 1
while The halting criterion is not satisfied do
  dynamic-gen = $\frac{\text{gen}}{\text{Max\_gen}}$
  for $i = 1$ to NP do
    Select uniform randomly $r_1 \neq r_2 \neq r_3 \neq i$
    $j_{\text{rand}} = \text{rndint}(1, D)$
    if $\text{rndreal}[0, 1] < \delta$ then
      $\eta_i' = \text{rndreal}[0, \text{dynamic\_gen}]$
    else
      $\eta_i' = \eta_i$
    end if
    for $j = 1$ to D do
      if $\text{rndreal}_j[0, 1) < \text{CR}$ or $j == j_{\text{rand}}$ then
        $u_{i,j}$ is generated by the mutation strategy of DE {explorative operation}
      else if $\text{rndreal}_j[0, 1) < \eta_j'$ then
        $u_{i,j}$ is generated by the operator with powerful exploitation {exploitative operation}
      else
        $u_{i,j} = x_{i,j}$
      end if
    end for
  end for
  for $i = 1$ to NP do
    Evaluate the offspring $\vec{u}_i$
    if $\vec{u}_i$ is better than $\vec{x}_i$ then
      Replace $\vec{x}_i$ with $\vec{u}_i$
      $\eta_i = \eta_i'$
    end if
  end for
  gen = gen + 1
end while

following repair rule is applied:

$$x_i = \begin{cases} L_i + \text{rndreal}_i[0,1] \times (U_i - L_i) & \text{if } x_i < L_i \\ U_i - \text{rndreal}_i[0,1] \times (U_i - L_i) & \text{if } x_i > U_i \end{cases}, \tag{12}$$

where $\text{rndreal}_i[0,1]$ is the uniform random variable from $[0,1]$ in the ith dimension.

4.5. *DE with the hybrid generation scheme*

By incorporating the above-mentioned hybrid generation scheme and the self-adaptive exploitation factor into the original DE framework shown in Algorithm 1, an improved DE algorithm is developed. The pseudo-code of this algorithm is described in Algorithm 4, where η_i' is the exploitation factor of the offspring. Based on the exploitation factor η, the algorithm can self-adaptively control the exploitative operation.

5. Experimental Results and Analysis

In this section, we verify the performance of our proposed hybrid generation scheme. To verify the performance we first choose the migration operator of BBO[43] as the exploitation operator; and the algorithm is referred to as DE-BBO. The reason is that the BBO algorithm is well supported theoretically. We will also use the "DE/best/1" mutation as the exploitation operator to test the performance of the new generation scheme in Section 5.7; and the algorithm is referred to as DE–DE. In addition, 23 functions are selected from the literature as the benchmark problems.

5.1. *Test functions*

In this work, we have carried out different experiments using a test suite, consisting of 23 unconstrained single-objective benchmark functions with different characteristics chosen from the literature. All of the functions are minimization and scalable problems. The first 13 functions, f01–f13, are chosen from Ref. 52. The rest of the 10 functions, F01–F10, are the new test functions provided by the CEC2005 special session.[47] Since we do not make any changes to these problems, we only briefly describe them in Table 1. More details can be found in Refs. 52 and 47.

Functions f01–f04 are unimodal. The generalized Rosenbrock's function f05 is a multimodal function when $D > 3$.[42] Function f06 is the step function, which has one minimum and is discontinuous. Function f07 is a noisy quartic function. Functions f08–f13 are multimodal functions where the number of local minima increases exponentially with the problem dimension. They appear to be the most difficult class of problems for many optimization algorithms. Functions F01–F05 are unimodal. Functions F06–F10 are multi-modal. Functions F01 and F09 are

Table 1. Benchmark functions used in our experimental studies.

Prob	Name	D	S	Optimal
f01	Sphere Model	Scalable	$[-100, 100]^D$	0
f02	Schwefel's Problem 2.22	Scalable	$[-10, 10]^D$	0
f03	Schwefel's Problem 1.2	Scalable	$[-100, 100]^D$	0
f04	Schwefel's Problem 2.21	Scalable	$[-100, 100]^D$	0
f05	Generalized Rosenbrock's Functions	Scalable	$[-30, 30]^D$	0
f06	Step Function	Scalable	$[-100, 100]^D$	0
f07	Quartic Function	Scalable	$[-1.28, 1.28]^D$	0
f08	Generalized Schwefel's Problem 2.26	Scalable	$[-500, 500]^D$	$-418.9829 \times D$
f09	Generalized Rastrigin's Function	Scalable	$[-5.12, 5.12]^D$	0
f10	Ackley's Function	Scalable	$[-32, 32]^D$	0
f11	Generalized Griewank Function	Scalable	$[-600, 600]^D$	0
f12	Generalized Penalized Function 1	Scalable	$[-50, 50]^D$	0
f13	Generalized Penalized Function 2	Scalable	$[-50, 50]^D$	0
F01	Shifted Sphere Function	Scalable	$[-100, 100]^D$	0
F02	Shifted Schwefel's Problem 1.2	Scalable	$[-100, 100]^D$	0
F03	Shifted Rotated High Conditioned Elliptic Function	Scalable	$[-100, 100]^D$	0
F04	Shifted Schwefel's Problem 1.2 with Noise in Fitness	Scalable	$[-100, 100]^D$	0
F05	Schwefel's Problem 2.6 with Global Optimum on Bounds	Scalable	$[-100, 100]^D$	0
F06	Shifted Rosenbrock's Function	Scalable	$[-100, 100]^D$	0
F07	Shifted Rotated Griewank's Function without Bounds	Scalable	R	0
F08	Shifted Rotated Ackley's Function with Global Optimum on Bounds	Scalable	$[-32, 32]^D$	0
F09	Shifted Rastrigin's Function	Scalable	$[-5, 5]^D$	0
F10	Shifted Rotated Rastrigin's Function	Scalable	$[-5, 5]^D$	0

separable, and the remaining eight functions are non-separable. The shifted and/or rotated features make these 10 functions very difficult to solve.

5.2. *Experimental setup*

In this work, we will mainly use the DE-BBO algorithm to evaluate the performance of our proposed hybrid generation scheme. For DE-BBO, we have chosen a reasonable set of value and have not made any effort in finding the best parameter settings. For all experiments, we use the following parameters unless a change is mentioned. When DE and DE-BBO adopt the self-adaptive control parameter proposed in Ref. 6 for CR and F values, they are referred to as jDE and jDE-BBO, respectively.

— Dimension of each function: $D = 30$;
— Population size: NP $= 100$[33,52,6,40];
— Maximum immigration rate: $I = 1.0$[43];
— Maximum emigration rate: $E = 1.0$[43];
— Probability of adjusting η: $\delta = 0.1$ (discussed in Sec. 5.6);
— DE mutation scheme: DE/rand/1/bin[45,33,29,49,40];

— Value to reach: VTR $= 10^{-8}$,[47,40] except for f07 of VTR $= 10^{-2}$;
— Max_NFFEs[b]: For f01, f06, f10, f12, and f13, Max_NFFEs = 150,000; for f03–f05, Max_NFFEs = 500, 000; for f02 and f11, Max_NFFEs = 200,000; for f07–f09 and F01–F10, Max_NFFEs = 300,000.

Moreover, in our experiments, each function is optimized over 50 independent runs. We also use the same set of initial random populations to evaluate different algorithms in a similar way done in Ref. 33. All the algorithms are implemented in standard C++.

5.3. *Performance criteria*

Five performance criteria are selected from the literature[47,40] to evaluate the performance of the algorithms. These criteria are described as follows.

— **Error**[47]**:** The error of a solution $\vec{x}$ is defined as $f(\vec{x} - \vec{x}^*)$, where $\vec{x}^*$ is the global minimum of the function. The minimum error is recorded when the Max_NFFEs is reached in 50 runs. The average and standard deviation of the error values are calculated as well.
— **NFFEs**[47]**:** The number of fitness function evaluations (NFFEs) is also recorded when the VTR is reached. The average and standard deviation of the NFFEs values are calculated.
— **Number of successful runs (SR)**[47]**:** The number of successful runs is recorded when the VTR is reached before the Max NFFEs condition terminates the trial.
— **Convergence graphs**[47]**:** The convergence graphs show the average error performance of the best solution over the total runs, in the respective experiments.
— **Acceleration rate (AR)**[40]**:** This criterion is used to compare the convergence speeds between our approach and other algorithms. It is defined as follows: $\text{AR} = \text{NFFEs}_{\text{other}}/\text{NFFEs}_{\text{our}}$, where AR > 1 indicates our approach is faster than its competitor.

5.4. *Influence of different CR values*

As mentioned in Sec. 4.2, the crossover rate CR plays an important role in the performance of our proposed hybrid generation scheme. In this section, we verify the influence of different CR values (CR $= 0.1, 0.5, 0.9$) to DE and DE-BBO. For DE and DE-BBO, the scaling factor $F = \text{rndreal}(0.1, 1.0)$.[37,8] Other parameters used for DE-BBO and DE are the same as described in Sec. 5.2. To save space, only 13 functions, f01–f13, are chosen to compare the results. The results are shown in Table 2. Additionally, the results of jDE and jDE-BBO are also reported in this table. The best results are highlighted in **boldface**.

[b]TheMax_NFFEs for functions f01–f13 are set as in Ref. 52, except for f05, f08, and f09; they are less than the values in Ref. 52. For functions F01–F10, the Max_NFFEs are set as in Ref. 47.

Table 2. Influence of different CR values and the self-adaptive control parameter for DE and DE-BBO at $D = 30$. Hereafter, a result with **boldface** means better value found.

	CR = 0.1		CR = 0.5	
Prob	DE	DE-BBO	DE	DE-BBO
f01	1.35E−17 ± 4.55E−18 (50)	**5.78E−32 ± 5.10E−32** (50)†	8.02E−19 ± 3.77E−19 (50)	**2.12E−23 ± 1.28E−23** (50)†
f02	1.89E−15 ± 4.30E−16 (50)	**0.00E+00 ± 0.00E+00** (50)†	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)
f03	2.26E+03 ± 4.17E+02 (0)	**1.64E+03 ± 3.54E+02** (0)†	1.42E+03 ± 4.56E+02 (0)	**4.53E+02 ± 2.62E+02** (0)†
f04	1.68E−07 ± 3.05E−08 (0)	**9.09E−16 ± 2.71E−16** (50)†	5.08E−10 ± 3.51E−09 (49)	**9.73E−14 ± 2.36E−13** (50)
f05	2.54E+01 ± 6.93E+00 (0)	**2.47E+01 ± 1.81E+00** (0)	**1.75E+01 ± 7.68E−01** (0)	1.76E+01 ± 7.27E−01 (0)
f06	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)
f07	8.99E−03 ± 1.76E−03 (36)	**5.86E−03 ± 1.30E−03** (50)†	5.57E−03 ± 1.24E−03 (50)	**4.57E−03 ± 1.06E−03** (50)†
f08	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)
f09	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	4.66E+01 ± 6.74E+00 (0)	**3.76E+01 ± 6.61E+00** (0)†
f10	8.50E−10 ± 1.32E−10 (50)	**4.14E−15 ± 0.00E+00** (50)†	2.08E−10 ± 5.02E−11 (50)	**8.63E−13 ± 2.26E−13** (50)†
f11	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)
f12	3.24E−19 ± 1.41E−19 (50)	**1.57E−32 ± 0.00E+00** (50)†	1.85E−19 ± 1.11E−19 (50)	**3.81E−24 ± 3.74E−24** (50)†
f13	6.06E−17 ± 2.43E−17 (50)	**4.23E−31 ± 3.20E−31** (50)†	4.67E−17 ± 2.76E−17 (50)	**1.09E−21 ± 7.42E−22** (50)†
	CR = 0.9		Self-adaptive parameter control	
Prob	DE	DE-BBO	jDE	jDE-BBO
f01	9.00E−20 ± 8.07E−20 (50)	**2.48E−22 ± 2.97E−22** (50)†	1.75E−27 ± 1.57E−27 (50)	**0.00E+00 ± 0.00E+00** (50)†
f02	1.89E−15 ± 9.02E−16 (50)	**3.33E−18 ± 1.74E−17** (50)†	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)
f03	1.40E−11 ± 3.39E−11 (50)	**2.49E−14 ± 4.39E−14** (50)†	4.03E−13 ± 6.20E−13 (50)	**1.42E−15 ± 3.34E−15** (50)†
f04	8.05E+00 ± 3.15E+00 (0)	**7.98E+00 ± 3.70E+00** (0)	**3.44E−14 ± 1.83E−13** (50)	7.81E−07 ± 3.13E−06 (44)
f05	9.27E−01 ± 1.49E+00 (2)	**5.28E−01 ± 1.38E+00** (18)†	9.99E−02 ± 1.23E−01 (1)	**7.98E−02 ± 5.64E−01** (4)
f06	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)
f07	3.66E−03 ± 1.11E−03 (50)	**3.31E−03 ± 1.08E−03** (50)	3.46E−03 ± 9.00E−04 (50)	**2.89E−03 ± 6.88E−04** (50)†
f08	6.10E+02 ± 7.35E+02 (5)	**3.65E+02 ± 3.48E+02** (4)†	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)
f09	1.39E+01 ± 9.20E+00 (0)	**8.22E+00 ± 3.26E+00** (0)†	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)
f10	7.03E−11 ± 3.70E−11 (50)	**4.29E−12 ± 4.16E−12** (50)†	1.54E−14 ± 5.48E−15 (50)	**4.14E−15 ± 0.00E+00** (50)†
f11	**1.04E−03 ± 2.86E−03** (44)	2.46E−03 ± 5.15E−03 (39)	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)
f12	1.74E−20 ± 6.93E−20 (50)	**8.50E−23 ± 3.38E−22** (50)	1.15E−28 ± 1.15E−28 (50)	**1.57E−32 ± 0.00E+00** (50)†
f13	5.48E−16 ± 2.07E−15 (50)	**9.67E−17 ± 4.75E−16** (50)	3.92E−26 ± 5.22E−26 (50)	**1.35E−32 ± 0.00E+00** (50)†

Note: †The value of t with 49 degrees of freedom is significant at $\alpha = 0.05$ by two-tailed test, hereinafter.

Since all algorithms start with the same initial population for each problem over each independent run, we use the paired t-test[22] to compare the results. The t-test is also used by other researchers in evolutionary computation, such as Refs. 52, 6, 38, etc.

From Table 2, we can observe that the CR value is sensitive to different problems for both DE and DE-BBO. For the majority of functions, DE-BBO outperforms DE in terms of the final best error values. The t-test results show that for CR = 0.1, 0.5, 0.9 DE-BBO is significantly better than DE on 8, 7, 7 (out of 13) functions, respectively. When using the self-adaptive control parameter for DE and DE-BBO, both jDE and jDE-BBO are more robust than DE and DE-BBO, respectively. For six functions, jDE-BBO is significantly better than jDE. Only for function f04, jDE is better than jDE-BBO. For other six functions, jDE-BBO obtains similar results to jDE.

Since the self-adaptive control parameter is more robust for both DE and DE-BBO, we only report the results of the self-adaptive control parameter-based DE variants in the following experiments.

5.5. *General performance of jDE-BBO*

In order to show the performance of jDE-BBO, we compare it with the jDE and BBO algorithms. The parameters used for jDE-BBO and jDE are the same as described in Sec. 5.2. The parameters of BBO are set as in Ref. 43, and the mutation operator with $m_{\max} = 0.005$ is also used in our experiments. All functions are conducted for 50 independent runs. Table 3 summarizes the results of jDE-BBO and jDE on all test functions.[c] In addition, some representative convergence graphs of jDE-BBO, jDE, and BBO are shown in Fig. 1.

With respect to the best error values shown in Table 3, it can be seen that jDE-BBO significantly outperforms jDE on nine out of 23 functions. For seven functions (f02, f06, f08, f09, f11, F01, and F09), both jDE-BBO and jDE can obtain the global optimum within the Max_NFFEs. For functions f05, F06, F07, and F10, jDE-BBO is slightly better than jDE. Only for two functions, f04 and F05, jDE is slightly better than jDE-BBO.

With respect to the NFFEs to reach the VTR, Table 3 indicates that jDE-BBO requires fewer NFFEs to reach the VTR for 14 functions. For functions f05 and F07, jDE is slightly faster than jDE-BBO. However, for these two functions, jDE-BBO obtains higher SR values than jDE. For the rest of the seven functions (F02–F06, F08, and F10), both jDE-BBO and jDE cannot reach the VTR within the Max_NFFEs. In addition, for the successful 16 functions the overall average AR value is 1.171, which indicates that jDE-BBO is on average 17.1% faster than jDE for these functions.

[c]For the sake of clarity and brevity, in Table 3 the results of BBO are not reported, because BBO is significantly outperformed by both jDE and jDE-BBO for all test functions. The reader may contact the authors for details.

Table 3. Comparison the performance between jDE-BBO and jDE at $D = 30$. "NA" means not available.

	Best error values						NFFEs				
	jDE			jDE-BBO			jDE		jDE-BBO		
Prob	Mean	Std	SR	Mean	Std	SR	Mean	Std	Mean	Std	AR
f01	1.75E−27	1.57E−27	50	**0.00E+00**	0.00E+00	50†	6.11E+04	1.12E+03	**5.24E+04**	9.54E+02	1.165
f02	0.00E+00	0.00E+00	50	0.00E+00	0.00E+00	50	8.45E+04	1.40E+03	**7.11E+04**	1.13E+03	1.189
f03	4.03E−13	6.20E−13	50	**1.42E−15**	3.34E−15	50†	3.57E+05	1.84E+04	**3.22E+05**	1.61E+04	1.109
f04	**3.44E−14**	1.83E−13	50	7.81E−07	3.13E−06	44	3.09E+05	4.54E+03	**2.41E+05**	8.04E+03	1.282
f05	9.99E−02	1.23E−01	1	**7.98E−02**	5.64E−01	4	**4.81E+05**	0.00E+00	4.83E+05	1.31E+04	*0.996*
f06	0.00E+00	0.00E+00	50	0.00E+00	0.00E+00	50	2.30E+04	7.05E+02	**2.01E+04**	6.68E+02	1.140
f07	3.46E−03	9.00E−04	50	**2.89E−03**	6.88E−04	50†	1.11E+05	2.23E+04	**1.00E+05**	2.38E+04	1.106
f08	0.00E+00	0.00E+00	50	0.00E+00	0.00E+00	50	9.58E+04	2.27E+03	**7.29E+04**	1.87E+03	1.314
f09	0.00E+00	0.00E+00	50	0.00E+00	0.00E+00	50	1.19E+05	4.08E+03	**8.90E+04**	2.05E+03	1.338
f10	1.54E−14	5.48E−15	50	**4.14E−15**	0.00E+00	50†	9.31E+04	1.63E+03	**7.64E+04**	1.32E+03	1.218
f11	0.00E+00	0.00E+00	50	0.00E+00	0.00E+00	50	6.47E+04	3.14E+03	**5.64E+04**	1.76E+03	1.148
f12	1.15E−28	1.15E−28	50	**1.57E−32**	0.00E+00	50†	5.52E+04	1.28E+03	**4.78E+04**	1.09E+03	1.156
f13	3.92E−26	5.22E−26	50	**1.35E−32**	0.00E+00	50†	6.71E+04	1.45E+03	**5.62E+04**	1.27E+03	1.195
F01	0.00E+00	0.00E+00	50	0.00E+00	0.00E+00	50	6.24E+04	1.04E+03	**5.58E+04**	1.03E+03	1.119
F02	8.33E−05	1.28E−04	0	**4.72E−07**	6.96E−07	0†	NA	NA	NA	NA	NA
F03	2.26E+05	1.05E+05	0	**1.77E+05**	1.07E+05	0†	NA	NA	NA	NA	NA
F04	1.06E−04	1.50E−04	0	**5.90E−07**	8.89E−07	0†	NA	NA	NA	NA	NA
F05	**9.44E+02**	3.55E+02	0	9.86E+02	4.07E+02	0	NA	NA	NA	NA	NA
F06	3.34E+01	3.79E+01	0	**1.82E+01**	2.36E+01	0	NA	NA	NA	NA	NA
F07	1.32E−02	1.01E−02	6	**1.05E−02**	8.63E−03	10	**1.89E+05**	1.08E+04	1.95E+05	1.52E+04	*0.973*
F08	2.10E+01	4.61E−02	0	2.10E+01	4.85E−02	0	NA	NA	NA	NA	NA
F09	0.00E+00	0.00E+00	50	0.00E+00	0.00E+00	50	1.12E+05	3.41E+03	**8.75E+04**	2.03E+03	1.284
F10	5.89E+01	8.79E+00	0	**5.84E+01**	1.10E+01	0	NA	NA	NA	NA	NA

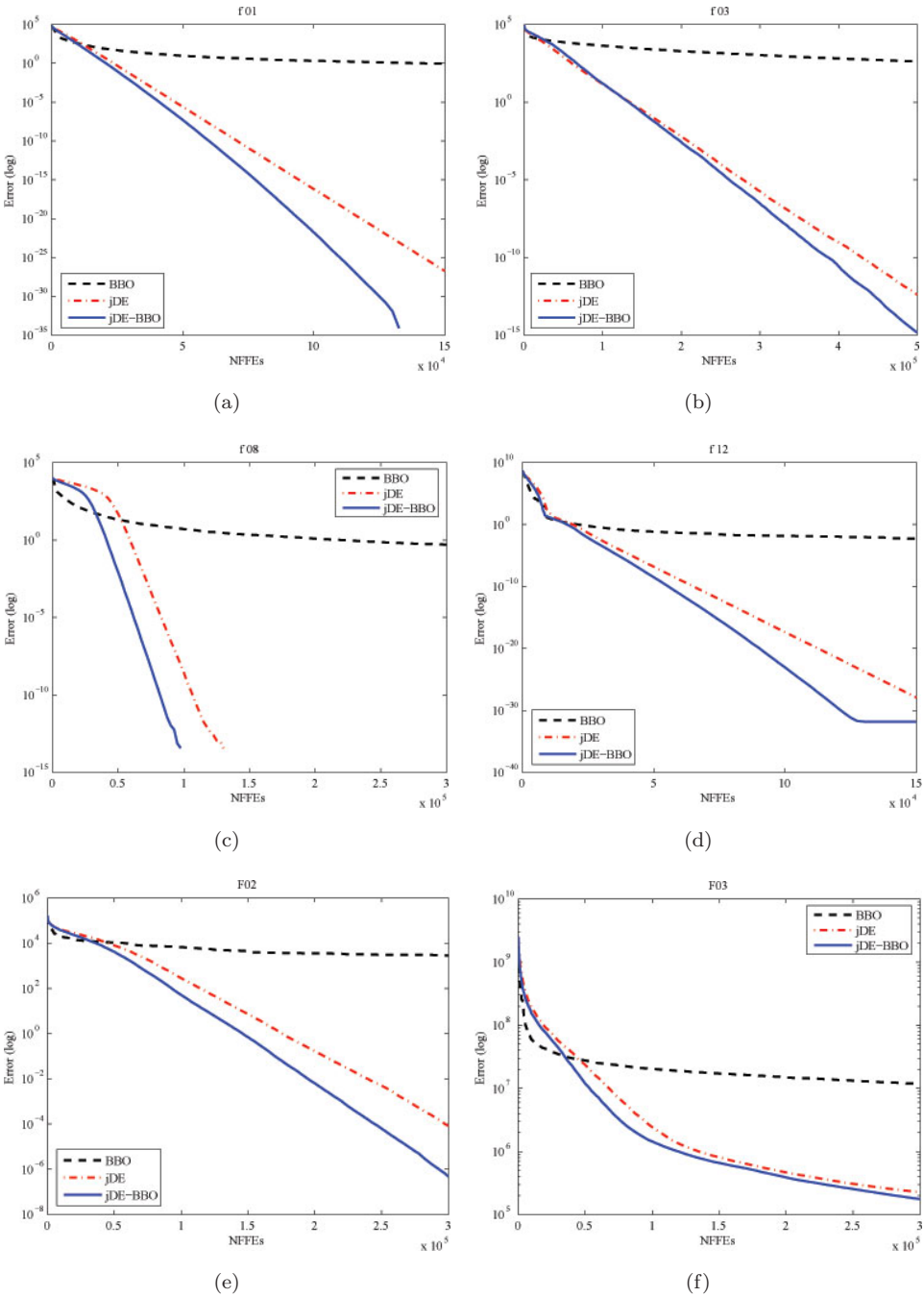
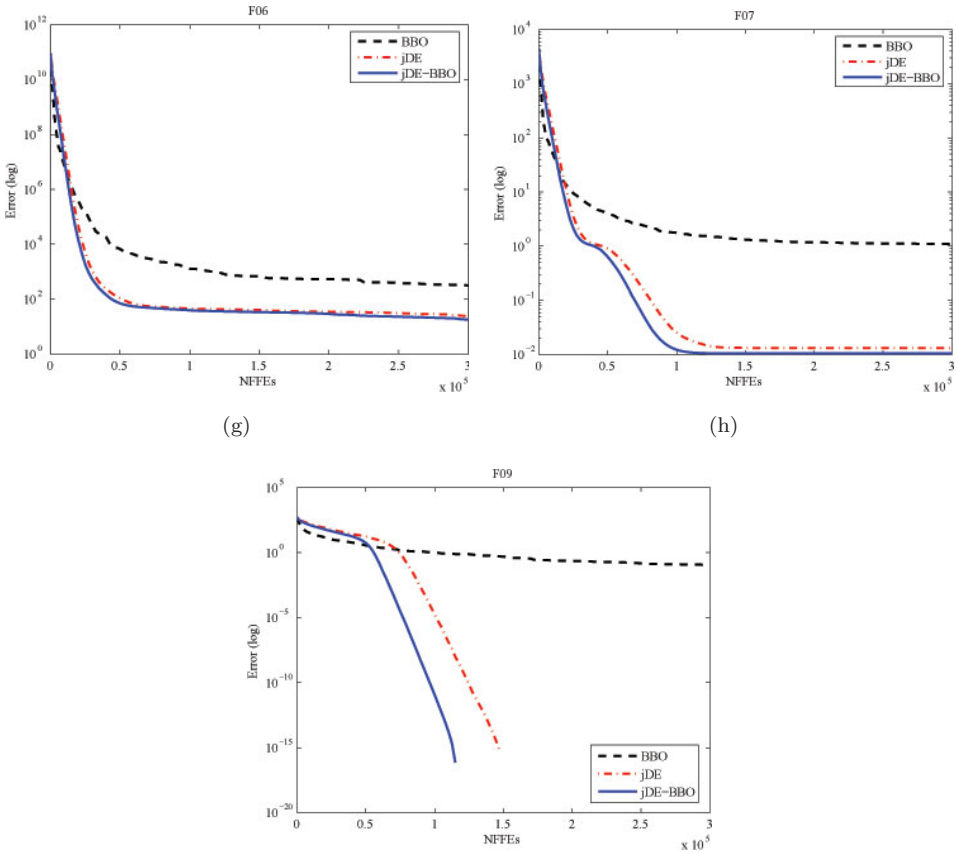


Fig. 1. Mean error curves of BBO, jDE, and jDE-BBO for selected functions. (a) f01, (b) f03, (c) f08, (d) f12, (e) F02, (f) F03, (g) F06, (h) F07, (i) F09.



(g) (h)

(i)

Fig. 1. (*Continued*)

From Fig. 1, it is apparent that in the early stages BBO converges faster than jDE and jDE-BBO, since the migration operator of BBO is good at exploiting the solutions. As the evolution progresses, both jDE and jDE-BBO are faster than BBO. Moreover, jDE-BBO achieves faster convergence rate than jDE.

In general, the performance of jDE-BBO is highly competitive with jDE. Our proposed hybrid generation scheme combined with the migration operator of BBO is able to enhance the exploitation of DE, and hence it can accelerate the original DE algorithm.

5.6. *Effect of different δ values*

Similar to τ_1 and τ_2 used in Ref. 6, the parameter δ is introduced to control the probability of adjusting the exploitation factor η. In this section, we make additional experiments to evaluate the effect of different δ values on the performance

of jDE-BBO. The best error values are reported in Table 4 for $\delta = 0.1$, 0.3, 0.6, and 0.9. All other parameters are kept unchanged as mentioned in Sec. 5.2. In the last column of Table 4, the paired t-tests are used to compare the means of the results between the best and the worst algorithms. Where "+" means the t value of 49 degrees of freedom is significant at $\alpha = 0.05$ by two-tailed test, and "$\approx$" indicates the difference of means is indifferent. In addition, the influence of δ from 0.1 to 1.0 with step size 0.1 is reported for six functions (f03, f08, f09, F03, F06, and F07) in Fig. 2. Note that for functions f03, f08, and f09 the NFFEs are reported, since both jDE-BBO (with different δ values) and jDE can solve these functions.

From Table 4 we can see that only for four functions the statistical results are significantly different between the best and the worst algorithms. For the rest of the 19 out of 23 functions, there is no significant difference. Moreover, Fig. 2 shows that the results are not significantly different for different δ values. In summary, the performance of jDE-BBO is not very sensitive to the choice of the parameter δ, especially for $0.1 \leq \delta \leq 0.4$.

5.7. *Effect of "DE/best/1" exploitative operation*

Since our proposed hybrid generation scheme is a generalized scheme, other operators with powerful exploitation can also be employed as the exploitative operation. In this section, the "DE/best/1" mutation is chosen as the exploitative operator. The algorithm is called jDE-DE. For the sake of simplicity, for "DE/best/1" in Eq. (3), $F = \text{rndreal}[0.1, 1.0]$, $r2$ and $r3$ are the same as in the "DE/rand/1" mutation. Table 5 shows the best error values for all test functions, where jDEbest is the algorithm using the "DE/best/1" mutation in Algorithm1. In the columns 5 and 6 of Table 5, the t-test results are listed, where "+", "−", and "$\approx$" indicate that jDE-DE is significantly better, significantly worse, and indifferent, respectively. The average and standard deviation of NFFEs are shown in Table 6. Table 6 only reports the results when both jDE and jDE-DE achieve the VTR over all 50 runs.

Compared with jDE, jDE-DE is significantly better than jDE for 11 functions in terms of the best error values. jDE-DE is significantly worse than jDE only for function F10. For seven functions, both jDE-DE and jDE can obtain the global optimum over all 50 runs. For the remaining four functions (f04, F05, F07, and F08), jDE-DE provides similar results to jDE. Moreover, from Table 5, we can see that the sum of SR values for jDE-DE is 858, which is larger than that of jDE (707). From Table 6, we can observe that jDE-DE needs less NFFEs than jDE to reach the VTR.

With respect to jDEbest, for 15 functions jDE-DE significantly outperforms jDEbest in terms of the best error values. For four functions (f03, F02, F03, and F04), jDEbest is significantly better than jDE-DE. The four functions are all unimodal functions. For the rest of the four functions, jDE-DE is slightly better than jDE. The sum of SR values of jDEbest is 410 which is less than half of that of jDE-DE (858).

Table 4. Influence of the parameter δ on the performance of jDE-BBO at $D = 30$.

Prob	$\delta = 0.1$	$\delta = 0.3$	$\delta = 0.6$	$\delta = 0.9$	t-test
f01	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	≈
f02	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	≈
f03	1.42E−15 ± 3.34E−15 (50)	**9.02E−16 ± 1.66E−15** (50)	1.05E−15 ± 1.63E−15 (50)	5.20E−15 ± 1.70E−14 (50)	≈
f04	7.81E−07 ± 3.13E−06 (44)	**7.31E−07 ± 3.63E−06** (41)	2.89E−05 ± 1.28E−04 (36)	4.07E−05 ± 1.83E−04 (32)	≈
f05	**7.98E−02 ± 5.64E−01** (4)	3.19E−01 ± 1.04E+00 (3)	9.84E−01 ± 1.75E+00 (2)	1.07E+00 ± 2.01E+00 (0)	+
f06	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	≈
f07	2.89E−03 ± 6.88E−04 (50)	2.62E−03 ± 5.41E−04 (50)	2.55E−03 ± 5.56E−04 (50)	**2.39E−03 ± 5.76E−04** (50)	+
f08	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	≈
f09	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	≈
f10	4.14E−15 ± 0.00E+00 (50)	4.14E−15 ± 0.00E+00 (50)	4.14E−15 ± 0.00E+00 (50)	4.14E−15 ± 0.00E+00 (50)	≈
f11	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	≈
f12	1.57E−32 ± 0.00E+00 (50)	1.57E−32 ± 0.00E+00 (50)	1.57E−32 ± 0.00E+00 (50)	1.57E−32 ± 0.00E+00 (50)	≈
f13	1.35E−32 ± 0.00E+00 (50)	1.35E−32 ± 0.00E+00 (50)	1.35E−32 ± 0.00E+00 (50)	1.35E−32 ± 0.00E+00 (50)	≈
F01	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	≈
F02	**4.72E−07 ± 6.96E−07** (0)	4.97E−07 ± 6.20E−07 (0)	1.66E−06 ± 5.03E−06 (0)	1.65E−06 ± 4.02E−06 (0)	≈
F03	1.77E+05 ± 1.07E+05 (0)	**1.54E+05 ± 8.33E+04** (0)	1.75E+05 ± 8.34E+04 (0)	1.60E+05 ± 8.68E+04 (0)	≈
F04	**5.90E−07 ± 8.89E−07** (0)	6.64E−07 ± 8.57E−07 (0)	2.20E−06 ± 6.78E−06 (0)	2.00E−06 ± 4.30E−06 (0)	≈
F05	9.86E+02 ± 4.07E+02 (0)	1.03E+03 ± 4.24E+02 (0)	9.79E+02 ± 4.42E+02 (0)	**8.92E+02 ± 3.75E+02** (0)	≈
F06	1.82E+01 ± 2.36E+01 (0)	**1.75E+01 ± 2.21E+01** (0)	2.39E+01 ± 2.58E+01 (0)	2.79E+01 ± 2.57E+01 (0)	+
F07	**1.05E−02 ± 8.63E−03** (10)	1.11E−02 ± 9.87E−03 (11)	1.63E−02 ± 1.03E−02 (3)	1.50E−02 ± 1.18E−02 (9)	+
F08	2.10E+01 ± 4.85E−02 (0)	2.09E+01 ± 5.37E−02 (0)	2.09E+01 ± 5.69E−02 (0)	2.09E+01 ± 4.55E−02 (0)	≈
F09	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	≈
F10	**5.84E+01 ± 1.10E+01** (0)	5.87E+01 ± 1.11E+01 (0)	5.93E+01 ± 1.21E+01 (0)	6.04E+01 ± 1.33E+01 (0)	≈

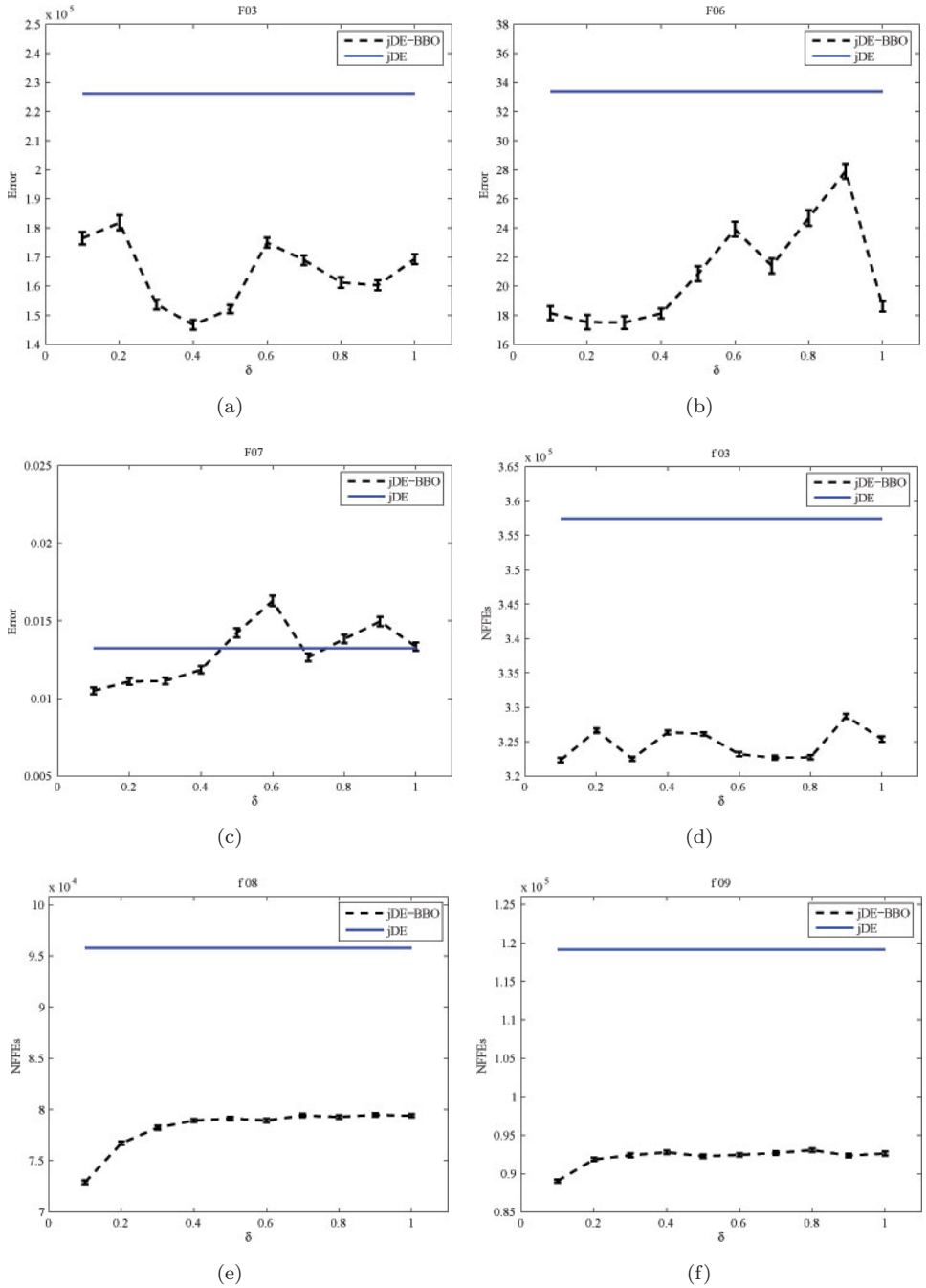


Fig. 2. Influence of jDE-BBO to different δ values for selected functions. (a) f03, (b) f08, (c) f09, (d) F03, (e) F06, (f) F07.

Table 5. Best error values of jDE-DE at $D = 30$, where "1 vs. 3" means "jDE vs. jDE-DE", and "2 vs. 3" means "jDEbest vs. jDE-DE".

Prob	jDE	jDEbest	jDE-DE	t-test: 1 vs. 3	t-test: 2 vs. 3
f01	1.75E−27 ± 1.57E−27 (50)	1.33E−31 ± 6.29E−31 (50)	**2.10E−33 ± 4.55E−33** (50)	+	≈
f02	**0.00E+00 ± 0.00E+00** (50)	6.02E−14 ± 2.76E−13 (50)	**0.00E+00 ± 0.00E+00** (50)	≈	≈
f03	4.03E−13 ± 6.20E−13 (50)	**1.72E−28 ± 1.01E−27** (50)	3.82E−19 ± 1.16E−18 (50)	+	−
f04	3.44E−14 ± 1.83E−13 (50)	1.37E+00 ± 1.43E+00 (0)	**3.56E−16 ± 1.99E−15** (50)	≈	+
f05	9.99E−02 ± 1.23E−01 (1)	1.20E+00 ± 1.85E+00 (35)	**2.38E−12 ± 1.37E−11** (50)	+	≈
f06	**0.00E+00 ± 0.00E+00** (50)	1.56E+01 ± 1.69E+01 (0)	**0.00E+00 ± 0.00E+00** (50)	≈	+
f07	3.46E−03 ± 9.00E−04 (50)	5.16E−03 ± 2.79E−03 (48)	**2.35E−03 ± 5.56E−04** (50)	+	+
f08	**0.00E+00 ± 0.00E+00** (50)	1.67E+03 ± 6.90E+02 (0)	**0.00E+00 ± 0.00E+00** (50)	≈	+
f09	**0.00E+00 ± 0.00E+00** (50)	3.96E+01 ± 1.34E+01 (0)	**0.00E+00 ± 0.00E+00** (50)	≈	+
f10	1.54E−14 ± 5.48E−15 (50)	3.08E+00 ± 7.99E−01 (0)	**4.14E−15 ± 0.00E+00** (50)	+	+
f11	**0.00E+00 ± 0.00E+00** (50)	3.21E−02 ± 3.84E−02 (11)	**0.00E+00 ± 0.00E+00** (50)	≈	+
f12	1.15E−28 ± 1.15E−28 (50)	1.08E+00 ± 1.67E+00 (15)	**1.58E−32 ± 7.30E−34** (50)	+	+
f13	3.92E−26 ± 5.22E−26 (50)	6.87E+00 ± 1.71E+01 (14)	**3.12E−32 ± 7.31E−32** (50)	+	+
F01	**0.00E+00 ± 0.00E+00** (50)	6.39E−28 ± 6.56E−28 (50)	**0.00E+00 ± 0.00E+00** (50)	≈	+
F02	8.33E−05 ± 1.28E−04 (0)	**3.57E−19 ± 1.39E−18** (50)	2.05E−09 ± 4.27E−09 (49)	+	−
F03	2.26E+05 ± 1.05E+05 (0)	**2.53E+04 ± 1.49E+04** (0)	1.37E+05 ± 8.74E+04 (0)	+	−
F04	1.06E−04 ± 1.50E−04 (0)	**4.51E−19 ± 1.77E−18** (50)	2.60E−09 ± 4.75E−09 (47)	+	−
F05	**9.44E+02 ± 3.55E+02** (0)	3.09E+03 ± 6.67E+02 (0)	9.52E+02 ± 5.00E+02 (0)	≈	+
F06	2.43E+01 ± 2.44E+01 (0)	1.83E+00 ± 2.01E+00 (26)	**4.33E−01 ± 1.26E+00** (1)	+	+
F07	1.32E−02 ± 1.01E−02 (6)	1.93E−02 ± 2.01E−02 (11)	**1.29E−02 ± 1.11E−02** (11)	≈	+
F08	2.10E+01 ± 4.61E−02 (0)	2.09E+01 ± 6.32E−02 (0)	2.09E+01 ± 4.69E−02 (0)	≈	≈
F09	**0.00E+00 ± 0.00E+00** (50)	5.47E+01 ± 1.56E+01 (0)	**0.00E+00 ± 0.00E+00** (50)	≈	+
F10	**5.89E+01 ± 8.79E+00** (0)	1.12E+02 ± 3.14E+01 (0)	6.37E+01 ± 1.32E+01 (0)	−	+

Table 6. NFFEs required to achieve the VTR for jDE-DE, where "1 vs. 3" means "jDE vs. jDE-DE", and "2 vs. 3" means "jDEbest vs. jDE-DE".

Prob	jDE	jDEbest	jDE-DE	AR: 1 vs. 3	AR: 2 vs. 3
f01	6.11E+04 ± 1.12E+03	**2.79E+04 ± 1.74E+03**	4.57E+04 ± 8.25E+02	1.34	*0.61*
f02	8.45E+04 ± 1.40E+03	**5.26E+04 ± 6.75E+03**	6.30E+04 ± 9.92E+02	1.34	*0.84*
f03	3.57E+05 ± 1.84E+04	**1.43E+05 ± 1.52E+04**	2.96E+05 ± 1.22E+04	1.21	*0.48*
f04	3.09E+05 ± 4.54E+03	NA ± NA	**2.32E+05 ± 7.61E+03**	1.33	NA
f06	2.30E+04 ± 7.05E+02	NA ± NA	**1.76E+04 ± 8.07E+02**	1.31	NA
f07	1.11E+05 ± 2.23E+04	1.22E+05 ± 6.45E+04 (48)	**7.88E+04 ± 1.76E+04**	1.40	1.54
f08	9.58E+04 ± 2.27E+03	NA ± NA	**7.70E+04 ± 2.31E+03**	1.24	NA
f09	1.19E+05 ± 4.08E+03	NA ± NA	**1.03E+05 ± 4.59E+03**	1.16	NA
f10	9.31E+04 ± 1.63E+03	NA ± NA	**6.48E+04 ± 8.27E+02**	1.44	NA
f11	6.47E+04 ± 3.14E+03	**2.82E+04 ± 1.34E+03** (11)	4.95E+04 ± 2.18E+03	1.31	*0.57*
f12	5.52E+04 ± 1.28E+03	5.04E+04 ± 9.56E+03 (15)	**4.15E+04 ± 1.06E+03**	1.33	1.21
f13	6.71E+04 ± 1.45E+03	8.86E+04 ± 1.79E+04 (14)	**4.87E+04 ± 9.94E+02**	1.38	1.82
F01	6.24E+04 ± 1.04E+03	9.05E+04 ± 1.13E+03	**5.07E+04 ± 1.25E+03**	1.23	1.79

In summary, the jDE-DE algorithm is more robust than jDE and jDEbest. The hybrid generation scheme combined with "DE/best/1" is also able to enhance the exploitation and accelerate the original DE algorithm.

5.8. *Comparison with other DE hybrids*

In this work, the main purpose is to introduce an alternative generation scheme of DE and demonstrate its benefits. In general, many other improvements of DE and other exploitative operators are also likely to be used to incorporate the hybrid generation scheme, such as the self-adaptive parameter control[6] and the strategy adaptation[38] of DE. Thus, the improvement of DE in this work should not be regarded as a competitor to other DE variants. However, to increase the experiment confidence, we compare jDE-BBO and jDE-DE with other DE hybrids in this section.

Since there are many DE variants, we only compare jDE-BBO and jDE-DE with DEahcSPX proposed in Ref. 33 and ODE proposed in Ref. 40. In DEahcSPX, a crossover-based adaptive local search operation is proposed to accelerate DE. The authors concluded that DEahcSPX outperforms the original DE algorithm in items of convergence rate in all experimental studies. In ODE, the opposition-based learning is used for the population initialization and generation jumping. Compared with the original DE algorithm and FADE, ODE performs better in terms of the convergence speed and solution accuracy. All of the parameter settings are the same as mentioned in Sec. 5.2. For DEahcSPX, the number of parents in SPX sets to be $n_p = 3$.[33] For ODE, the jump rate $J_r = 0.3$.[40] In order to make a fair comparison between jDE-BBO and jDE-DE, both DEahcSPX and ODE adopt the self-adaptive parameter control proposed in Ref. 6 for CR and F values. The two algorithms are referred to as jDEahcSPX and jODE, respectively.

The results are described in Table 7.[d] To save space, only the mean values are reported herein. "[a]" indicates the mean NFFEs required to reach the VTR. In the columns 6 through 9 of Table 7, the t-test results are listed, where "+", "−", and "≈" indicate that jDE-BBO or jDE-DE is significantly better, significantly worse, and indifferent, respectively, compared with jDEahcSPX or jODE. When all of the four algorithms achieve the VTR within theMax_NFFEs over all 50 runs, the AR is calculated in the columns 6 through 9.

With respect to jDE-BBO, for 13 functions jDE-BBO, jDEachSPX, and jODE are able to reach the VTR within theMax_NFFEs over all 50 runs. Only for function f07, jDE-BBO converges slower than jDEachSPX and jODE. For 12 functions, jDE-BBO obtains faster convergence rate than jDEachSPX and jODE. On average, jDE-BBO is 19%and 18% faster than jDEachSPX and jODE, respectively. In terms of the best error values for the remaining 10 functions (f04, f05, F02–F08, and

[d]Due to the tight space limitation, in Table 7 we only reported the summary results of all algorithms. For detailed results, the readers can contact the authors.

Table 7. Comparison of the performance of jDE-BBO and jDE-DE with other DE hybrids, where "1 vs. 3" indicates "jDE-BBO vs. jDEahcSPX", "1 vs. 4" indicates "jDE-BBO vs. jODE", "2 vs. 3" means "jDE-DE vs. jDEahcSPX", and "2 vs. 4" means "jDE-DE vs. jODE".

					AR or t-test			
Prob	jDE-BBO	jDE-DE	jDEachSPX	jODE	1 vs. 3	1 vs. 4	2 vs. 3	2 vs. 4
f01	[5.24E+04] (50)	**[4.57E+04]** (50)	[6.16E+04] (50)	[5.83E+04] (50)	1.18	1.11	1.35	1.28
f02	[7.11E+04] (50)	**[6.30E+04]** (50)	[8.58E+04] (50)	[1.05E+05] (50)	1.21	1.48	1.36	1.67
f03	[3.22E+05] (50)	**[2.96E+05]** (50)	[3.89E+05] (50)	[3.74E+05] (50)	1.21	1.16	1.31	1.26
f04	7.81E−07 (44)	**3.56E−16** (50)	1.09E−14 (50)	1.08E−04 (43)	≈	≈	≈	≈
f05	7.98E−02 (4)	**2.38E−12** (50)	1.48E−01 (1)	2.58E+01 (0)	+	+	+	+
f06	[2.01E+04] (50)	**[1.76E+04]** (50)	[2.30E+04] (50)	[2.15E+04] (50)	1.14	1.07	1.31	1.22
f07	[1.00E+05] (50)	[7.88E+04] (50)	[9.64E+04] (50)	**[3.61E+04]** (50)	*0.96*	*0.36*	*1.22*	*0.46*
f08	**[7.29E+04]** (50)	[7.70E+04] (50)	[9.44E+04] (50)	[1.06E+05] (50)	1.30	1.45	1.23	1.37
f09	**[8.90E+04]** (50)	[1.03E+05] (50)	[1.20E+05] (50)	[1.37E+05] (50)	1.34	1.54	1.16	1.33
f10	[7.64E+04] (50)	**[6.48E+04]** (50)	[9.43E+04] (50)	[9.13E+04] (50)	1.23	1.19	1.46	1.41
f11	[5.64E+04] (50)	**[4.95E+04]** (50)	[6.47E+04] (50)	[6.56E+04] (50)	1.15	1.16	1.31	1.32
f12	[4.78E+04] (50)	**[4.15E+04]** (50)	[5.59E+04] (50)	[5.03E+04] (50)	1.17	1.05	1.35	1.21
f13	[5.62E+04] (50)	**[4.87E+04]** (50)	[6.78E+04] (50)	[6.36E+04] (50)	1.21	1.13	1.39	1.31
F01	[5.58E+04] (50)	**[5.07E+04]** (50)	[6.27E+04] (50)	[6.22E+04] (50)	1.12	1.11	1.24	1.23
F02	4.72E−07 (0)	**2.05E−09** (49)	9.06E−05 (0)	2.94E−03 (0)	+	+	+	+
F03	1.77E+05 (0)	**1.37E+05** (0)	2.25E+05 (0)	3.18E+05 (0)	+	+	+	+
F04	5.90E−07 (0)	**2.60E−09** (47)	1.11E−04 (0)	3.72E−03 (0)	+	+	+	+
F05	9.86E+02 (0)	9.52E+02 (0)	**8.55E+02** (0)	8.78E+02 (0)	≈	≈	≈	≈
F06	1.82E+01 (0)	**4.33E−01** (1)	2.24E+01 (0)	5.15E+01 (0)	≈	+	+	+
F07	1.05E−02 (10)	1.29E−02 (11)	**9.65E−03** (13)	1.07E−02 (21)	≈	≈	≈	≈
F08	2.10E+01 (0)	2.09E+01 (0)	2.10E+01 (0)	2.09E+01 (0)	≈	≈	≈	≈
F09	**[8.75E+04]** (50)	[9.44E+04] (50)	[1.13E+05] (50)	[1.36E+05] (50)	1.29	1.55	1.20	1.44
F10	5.84E+01 (0)	6.37E+01 (0)	5.81E+01 (0)	**5.25E+01** (0)	≈	−	−	−

F10), jDE-BBO is significantly better than jDEahcSPX for four functions. For six functions there are no significant differences between jDE-BBO and jDEahcSPX. Compared with jODE, jDE-BBO significantly outperforms jODE for five functions. For function F10, jODE is significantly better than jDE-BBO. For the rest of the four functions, the results are indifferent.

With respect to jDE-DE, a similar conclusion to jDE-BBO can be drawn about the relative performance of jDE-DE, jDEahcSPX, and jODE, i.e., jDE-DE outperforms jDEahcSPX and jODE on the majority of functions. jDE-DE is on average 30% and 27% faster than jDEachSPX and jODE, respectively.

5.9. *Discussion*

The DE algorithm is a simple yet powerful population-based, direct search algorithm for global optimization. It is good at exploring the search space; however it lacks the exploitation of the solutions. Hence, there are many DE variants attempting to remedy this drawback. In this study, we propose a hybrid generation scheme to enhance the exploitation of DE. This improvement is different from the previous DE variants. To demonstrate the benefits of the proposed hybrid generation scheme, a comprehensive set of experiments were conducted. From the experimental results we can summarize that:

(1) Experimental results confirm our expectation that the proposed hybrid generation scheme is able to enhance the exploitation and accelerate the convergence speed of the original DE algorithm. Incorporating the self-adaptive exploitation factor, the hybrid generation scheme makes the original DE algorithm more effective.
(2) The enhancement of the exploitation of the hybrid generation scheme is not influenced by the parameter settings of CR, according to the results shown in Table 2.
(3) The parameter study of δ shows that the influence of δ is not significant, especially for $0.1 \leq \delta \leq 0.4$.
(4) Since the proposed generation scheme is a generalized scheme, two exploitative operators, the migration operator of BBO and the "DE/best/1" mutation, are chosen to test the performance of the scheme. The results indicate that both the operators can enhance the exploitation of DE. Moreover, by looking carefully at the results in Table 7, we can see that jDE-DE can obtain better results on 17 out of 23 functions than jDE-BBO. Hence, we can expect that other exploitative operators may still work well in our proposed scheme.
(5) Compared with other DE variants, the results show that our approach performs better, or at least comparably, in terms of the quality of the final solutions and the convergence rate.
(6) From the previous experiments, we can obtain that for some highly-complex problems (e.g., F05 and F10), the performance of jDE-BBO or jDE-DE is worse than jDE. Our future work will dedicate to improving it.

6. Conclusion and FutureWork

In this paper, we propose a generalized hybrid generation scheme of DE. The hybrid generation scheme is based on the binomial recombination operator of DE, where an additional exploitative operation is incorporated in the original DE generation scheme. Hence, our proposed hybrid generation scheme has the ability to enhance the exploitation and accelerate the convergence rate of DE. Compared with the original DE generation scheme, the hybrid generation scheme is also very simple and easy to implement. In addition, an exploitation factor η is introduced to control the frequency of the exploitative operation. Through a comprehensive set of experimental verifications of our hybrid generations scheme, the results confirm the enhancement of the exploitation by the scheme. The hybrid generation scheme is able to make the original DE algorithm more effective in terms of the quality of the final results and the reduction of the NFFEs. On the error criterion, our approach can provide the smallest error values on the majority of the functions. On the SR criterion, our approach still obtains the highest overall SR values. With respect to NFFEs, convergence graph, and AR criteria, the results show that our approach converges faster compared with other DE variants considered in this paper. Thus, our proposed hybrid generation scheme can be regarded as an alternative scheme of DE for further research.

Since the proposed generation scheme is a generalized scheme, other exploitation operators (e.g., the SPX operator[50] and the PCX operator[12]) can also be used to enhance the performance of the original DE algorithm. Our future work is required to verify this expectation. Additionally, combination with the constraint-handling techniques is one possible future study using the proposed generation scheme for the constrained problems.

Acknowledgment

This work was partly supported by the Foundation of State Key Lab of Software Engineering under Grant No. SKLSE2010-08-13, the National Natural Science Foundation of China under Grant No. 61075063, and the Research Fund for the Doctoral Program of Higher Education under Grant No. 20090145110007. The authors would like to thank the anonymous reviewers for their constructive comments.

References

1. B. Alatas, E. Akin and A. Karci, MODENAR: Multi-objective differential evolution algorithm for mining numeric association rules, *Appl. Soft Comput.* **8**(1) (Jan. 2008) 646–656.
2. M. M. Ali and L. P. Fatti, A differential free point generation scheme in the differential evolution algorithm, *J. Global Optimization* **35**(4) (2006) 551–572.

3. J. Andre, P. Siarry and T. Dognon, An improvement of the standard genetic algorithm fighting premature convergence in continuous optimization, *Adv. Eng. Softw.* **32**(1) (2000) 49–60.
4. T. Back, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms* (Oxford University Press, Oxford, UK, 1996).
5. J. Brest, B. Boskovic, S. Greiner, V. Zumer and M. S. Maucec, Performance comparison of self-adaptive and adaptive differential evolution algorithms, *Soft Comput.* **11**(7) (May 2007) 617–629.
6. 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.* **10**(6) (Dec. 2006) 646–657.
7. J. Brest and M. S. Maucec, Population size reduction for the differential evolution algorithm, *Appl. Intell.* **29**(3) (2008) 228–247.
8. U. K. Chakraborty, *Advances in Differential Evolution* (Springer-Verlag, Berlin, 2008).
9. S. Das, A. Abraham and A. Konar, Automatic clustering using an improved differential evolution algorithm, *IEEE Trans. Syst. Man Cybern.* **38**(1) (Feb. 2008) 218–237.
10. S. Das and A. Konar, Two-dimensional IIR filter design with modern search heuristics: A comparative study, *Int. J. Comput. Intell. Appl.* **6**(3) (2006) 329–355.
11. S. Das, A. Konar and U. K. Chakraborty, Two improved differential evolution schemes for faster global search, in Hans-Georg Beyer and Una-May O'Reilly (eds.), *Genetic and Evolutionary Computation Conference, GECCO 2005, Proceedings* (ACM, Washington DC, USA, June 25–29, 2005), pp. 991–998.
12. K. Deb, A. Anand and D. Joshi, A computationally efficient evolutionary algorithm for real-parameter optimization, *Evolut. Comput.* **10**(4) (2002) 345–369.
13. C. Ebenau, J. Rottschafer and G. Thierauf, An advanced evolutionary strategy with an adaptive penalty function for mixed-discrete structural optimisation, *Adv. Eng. Softw.* **36**(1) (2005) 29–38.
14. A. E. Eiben, R. Hinterding and Z. Michalewicz, Parameter control in evolutionary algorithms, *IEEE Trans. Evolut. Computat.* **3**(2) (Jul. 1999) 124–141.
15. H.-Y. Fan and J. Lampinen, A trigonometric mutation operation to differential evolution, *J. Global Optimization* **27**(1) (2003) 105–129.
16. V. Feoktistov, *Differential Evolution: In Search of Solutions* (Springer-Verlag, New York, Inc., Secaucus, NJ, USA, 2006).
17. V. Feoktistov and S. Janaqi, *Generalization of the Strategies in Differential Evolution*, Vol. 7 (IEEE Computer Society, Los Alamitos, CA, USA, 2004), p. 165a.
18. X. Z. Gao, X. Wang and S. J. Ovaska, Fusion of clonal selection algorithm and differential evolution method in training cascadeccorrelation neural network, *Neurocomputing* **72**(10–12) (2009) 2483–2490.
19. R. Gaperle, S. D. Muler and P. Koumoutsakos, A parameter study for differential evolution, in *Proc. WSEAS Int. Conf. Advances Intell. Syst. Fuzzy Syst. Evol. Comput.* (2002) 293–298.
20. M. Gong, L. Jiao and X. Zhang, Immune secondary response and clonal selection inspired optimizers, *Neurocomputing* **72**(1–3) (2008) 149–161.
21. W. Gong, Z. Cai and C. X. Ling, ODE: A fast and robust differential evolution based on orthogonal design, in *AI 2006: Advances in Artificial Intelligence, 19th Australian Joint Conference on Artificial Intelligence*, Hobart, Australia, December 4–8, 2006, LNAI 4304 (Springer, Berlin, German, 2006), pp. 709–718.
22. C. H. Goulden, *Methods of Statistical Analysis*, 2nd edn. (Wiley, New York, 1956).

23. C. Grosan, A. Abraham and H. Ishibuchi, *Hybrid Evolutionary Algorithms* (Springer-Verlag, Berlin, Germany, 2009).
24. O. Hrstka and A. Kucerova, Improvements of real coded genetic algorithms based on differential operators preventing premature convergence, *Adv. Eng. Softw.* **35**(3–4) (2004) 237–246.
25. A. W. Iorio and X. Li, Solving rotated multi-objective optimization problems using differential evolution, in *AI 2004: Advances in Artificial Intelligence, Proceedings* (Springer-Verlag, LNAI 3339, 2004), pp. 861–872.
26. P. Kaelo and M. M. Ali, Differential evolution algorithms using hybrid mutation, *Comput. Optim. Appl.* **37**(2) (2007) 231–246.
27. P. Larranaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation* (Springer-Verlag, Berlin, Germany, 2001).
28. J. J. Liang, A. K. Qin, P. N. Suganthan and S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, *IEEE Trans. Evol. Comput.* **10**(3) (2006) 281–295.
29. J. Liu and J. Lampinen, A fuzzy adaptive differential evolution algorithm, *Soft Comput.* **9**(6) (2005) 448–462.
30. E. Mezura-Montes, J. Velazquez-Reyes and C. A. C. Coello, A comparative study of differential evolution variants for global optimization, in Mike Cattolico (ed.), *Genetic and Evolutionary Computation Conference, GECCO 2006, Proceedings* (ACM, Seattle, Washington, USA, July 8–12, 2006), pp. 485–492.
31. F. Neri and V. Tirronen, Scale factor local search in differential evolution, *Memetic Comput.* **2** (2009) 153–171.
32. A. Nobakhti and H. Wang, A simple self-adaptive differential evolution algorithm with application on the alstom gasifier, *Appl. Soft Comput.* **8**(1) (2008) 350–370.
33. N. Noman and H. Iba. Accelerating differential evolution using an adaptive local search, *IEEE Trans. Evol. Comput.* **12**(1) (Feb. 2008) 107–125.
34. N. Noman and H. Iba, Enhancing differential evolution performance with local search for high dimensional function optimization, in Hans-Georg Beyer and Una-May O'Reilly (ed.), *Genetic and Evolutionary Computation Conference, GECCO 2005, Proceedings* (ACM, Washington DC, USA, June 25–29, 2005), pp. 967–974.
35. Y.-S. Ong and A. J. Keane, Meta-lamarckian learning in memetic algorithms, *IEEE Trans. Evol. Comput.* **8**(2) (Apr. 2004) 99–110.
36. G. C. Onwubolu and D. Davendra, *Differential Evolution: A Handbook for Global Permutation-Based Combinatorial Optimization* (Springer-Verlag, Berlin, 2009).
37. K. Price, R. Storn and J. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization* (Springer-Verlag, Berlin, 2005).
38. A. K. Qin, V. L. Huang and P. N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, *IEEE Trans. Evol. Comput.* **13**(2) (Apr. 2009) 398–417.
39. A. K. Qin and P. N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, in *IEEE Congress on Evolutionary Computation (CEC2005)* (IEEE, 2005), pp. 1785–1791.
40. S. Rahnamayan, H. R. Tizhoosh and M. M. A. Salama, Opposition-based differential evolution, *IEEE Trans. Evol. Comput.* **12**(1) (Feb. 2008) 64–79.
41. A. Salman, A. P. Engelbrecht and M. G. H. Omran, Empirical analysis of self-adaptive differential evolution, *European J. Operat. Res.* **183**(2) (2007) 785–804.
42. Y.-W. Shang and Y.-H. Qiu, A note on the extended rosenbrock function, *Evol. Comput.* **14**(1) (2006) 119–126.

43. D. Simon, Biogeography-based optimization, *IEEE Trans. Evol. Comput.* **12**(6) (Dec. 2008) 702–713.
44. D. Simon, The matlab code of biogeography-based optimization, 2008. URL: http://academic.csuohio.edu/simond/bbo/.
45. R. Storn and K. Price, Differential evolution — a simple and efficient heuristic for global optimization over continuous spaces, *J. Global Optim.* **11**(4) (Dec. 1997) 341–359.
46. R. Storn and K. Price, Home page of differential evolution, 2010. URL: http://www.icsi.berkeley.edu/∼storn/code.html.
47. 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 CEC2005 special session on real-parameter optimization, 2005.
48. J. Sun, Q. Zhang and E. P. K. Tsang, DE/EDA: A new evolutionary algorithm for global optimization, *Inform. Sci.* **169**(3-4) (2005) 249–262.
49. J. Teo, Exploring dynamic self-adaptive populations in differential evolution, *Soft Comput.* **10**(8) (2006) 673–686.
50. S. Tsutsui, M. Yamamura and T. Higuchi, Multi-parent recombination with simplex crossover in real coded genetic algorithms, in *Genetic and Evolutionary Computation Conference, GECCO 1999, Proceedings* (Morgan Kaufmann, San Francisco, CA, 1999), pp. 657–664.
51. Y.-J. Wang, J.-S. Zhang and G.-Y. Zhang, A dynamic clustering based differential evolution algorithm for global optimization, *Euro. J. Operat. Res.* **183**(1) (2007) 56–73.
52. X. Yao, Y. Liu and G. Lin, Evolutionary programming made faster, *IEEE Trans. Evol. Comput.* **3**(2) (Jul. 1999) 82–102.
53. D. Zaharie, Control of population diversity and adaptation in differential evolution algorithms, in R. Matousek and P. Osmera (ed.), *Proc. of Mendel 2003, 9th International Conference on Soft Computing* (2003), pp. 41–46.
54. W. Zhang and X. Xie, *DEPSO: Hybrid Particle Swarm with Differential Evolution Operator* (2003) (IEEE Computer Society, Washington, DC, USA, 2003), pp. 3816–3821.