Soft Comput (2011) 15:2127–2140
DOI 10.1007/s00500-010-0642-7

FOCUS

Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems

Hui Wang · Zhijian Wu · Shahryar Rahnamayan

Published online: 14 September 2010

Abstract This paper presents a novel algorithm based on generalized opposition-based learning (GOBL) to improve the performance of differential evolution (DE) to solve high-dimensional optimization problems efficiently. The proposed approach, namely GODE, employs similar schemes of opposition-based DE (ODE) for opposition-based population initialization and generation jumping with GOBL. Experiments are conducted to verify the performance of GODE on 19 high-dimensional problems with $D = 50, 100, 200, 500, 1{,}000$. The results confirm that GODE outperforms classical DE, real-coded CHC (crossgenerational elitist selection, heterogeneous recombination, and cataclysmic mutation) and G-CMA-ES (restart covariant matrix evolutionary strategy) on the majority of test problems.

Keywords Differential evolution · Opposition-based DE · Evolutionary computation · Global optimization · High-dimensional optimization · Large-scale optimization

1 Introduction

Many real-world problems may be formulated as optimization problems with variables in continuous domains (continuous optimization problems). In the past decades, different kinds of nature-inspired optimization algorithms have been designed and applied to solve optimization problems, e.g., simulated annealing (SA; Kirkpatrick et al. 1983), evolutionary algorithms (EAs; Bäck 1996), differential evolution (DE; Storn and Price 1997), particle swarm optimization (PSO; Kennedy and Eberhart 1995), ant colony optimization (ACO; Dorigo et al. 1996), estimation of distribution algorithms (EDA; Larranaga and Lozano 2001), etc.

Although these algorithms have shown good optimization performance in solving lower dimensional problems ($D < 100$), many of them suffer from the *curse of dimensionality*, which implies that their performance deteriorates quickly as the dimension of the problem increases. The main reason is that in general the complexity of the problem increases exponentially with its dimension. The majority of evolutionary algorithms lose the power of searching the optima solution when the dimension increases. So, more efficient search strategies are required to explore all the promising regions in a given time budget (Tang et al. 2007).

Opposition-based learning (OBL), introduced by Tizhoosh (2005), is a machine intelligence strategy, which considers current estimate and its opposite estimate at the same time to achieve a better approximation for a current candidate solution. It has been proved (Rahnamayan et al. 2008a) that an opposite candidate solution has a higher chance to be closer to the global optimum solution than a random candidate solution. The idea of OBL has been used to enhance population-based algorithms (Rahnamayan et al. 2006a, b, 2008b; Wang et al. 2007). Opposition-based DE (ODE) is one of these successful applications, which shows excellent search abilities in solving both low-dimensional and high-dimensional problems (Rahnamayan

H. Wang · Z. Wu (✉)
State Key Laboratory of Software Engineering,
Wuhan University, Wuhan 430072, China
e-mail: zjwu9551@sina.com

H. Wang
e-mail: wanghui_cug@yahoo.com.cn

S. Rahnamayan
Faculty of Engineering and Applied Science,
University of Ontario Institute of Technology (UOIT),
2000 Simcoe Street North, Oshawa, ON L1H 7K4, Canada
e-mail: shahryar.rahnamayan@uoit.ca

et al. 2008b; Rahnamayan and Wang 2008). In this paper, we present an enhanced ODE algorithm to solve high-dimensional continuous optimization problems more efficiently. The proposed approach is called GODE, which employs a generalized opposition-based learning (GOBL) concept (Wang et al. 2009a, b).

The rest of the paper is organized as follows. In Sect. 2, the classical DE algorithm is briefly introduced. Section 3 presents some reviews of related work on large-scale global optimization. The GOBL and its analysis are given in Sect. 4. Section 5 gives an implementation of the proposed algorithm, GODE. In Sect. 6, the test suite, parameter settings, results and discussions are presented. Finally, the work is concluded in Sect. 7.

2 Differential evolution

Differential evolution (DE), proposed by Storn and Price (1997), is an effective, robust and simple global optimization algorithm. According to frequently reported experimental studies, DE has shown better performance than many other evolutionary algorithm (EAs) in terms of convergence speed and robustness over several benchmark functions and real-world problems (Vesterstrom and Thomsen 2004).

There are several variants of DE (Storn and Price 1997). According to the suggestions of Herrera et al. (2010a), the *rand/1/exp* strategy shows a better performance to solve high-dimensional problems. Our proposed algorithm is also based on this DE scheme. Let us assume that $X_i(t)(i = 1, 2, \ldots, N_p)$ is the ith individual in population $P(t)$, where N_p is the population size, t is the generation index, and $P(t)$ is the population in the tth generation. The main idea of DE is to generate trial vectors. Mutation and crossover are used to produce new trial vectors, and selection determines which of the vectors will be successfully selected into the next generation.

Mutation: For each vector $X_i(t)$ in Generation t, a mutant vector V is generated by

$$V_i(t) = X_{i_1}(t) + F(X_{i_2}(t) - X_{i_3}(t)), \quad i \neq i_1 \neq i_2 \neq i_3, \tag{1}$$

where $i = 1, 2, \ldots, N_p$ and i_1, i_2, and i_3 are mutually different random integer indices within $[1, N_p]$. The population size N_p should be satisfied $N_p \geq 4$ because i, i_1, i_2, and i_3 are different. $F \in [0, 2]$ is a real number that controls the amplification of the difference vector $(X_{i_2}(t) - X_{i_3}(t))$.

Crossover: Like genetic algorithms, DE also employs a crossover operator to build trial vectors $(U_i(t) = \{U_{i,1}(t), U_{i,2}(t), \ldots, U_{i,D}(t)\})$ by recombining two different vectors. In this paper, we use the *rand/1/exp* strategy to generate the trial vectors.

Selection: A greedy selection mechanism is used as follows:

$$X_i(t) = \begin{cases} U_i(t), & \text{if } f(U_i(t)) \leq f(X_i(t)) \\ X_i(t), & \text{otherwise} \end{cases} \tag{2}$$

Without loss of generality, this paper only considers minimization problem. If, and only if, the trial vector $U_i(t)$ is better than $X_i(t)$, then $X_i(t)$ is set to $U_i(t)$; otherwise, the $X_i(t)$ remains unchanged.

3 Related works

In the past several years, the research on solving high-dimensional optimization problems has attracted much attention. Some excellent works have been proposed. In this section, a brief overview of these approaches is presented.

Yang et al. (2008) proposed a multilevel cooperative coevolution algorithm based on self-adaptive neighborhood search DE (SaNSDE) to solve large-scale problems. Hsieh et al. (2008) presented an efficient population utilization strategy for PSO (EPUS-PSO) to manage the population size. Brest et al. (2008) introduced a population size reduction mechanism into self-adaptive DE, where the population size decreases during the evolutionary process. Tseng and Chen (2008) presented multiple trajectory search (MTS) by using multiple agents to search the solution space concurrently. Zhao et al. (2008) used dynamic multi-swarm PSO with local search (DMS-PSO) for large-scale problems. Rahnamayan and Wang (2008) presented an experimental study of opposition-based DE (ODE; Rahnamayan et al. 2008b) on large-scale problems. The reported results show that ODE significantly improves the performance of standard DE. Wang and Li (2008) proposed a univariate EDA (LSEDA-gl) by sampling under mixed Gaussian and lévy probability distribution. Rahnamayan and Wang (2009) introduced an effective population initialization mechanism when dealing with large-scale search spaces.

Molina et al. (2009) presented a memetic algorithm by employing MTS and local search chains to deal with large-scale problems. García-Martínez and Lozano (2009) proposed a continuous variable neighborhood search algorithm based on evolutionary metaheuristic components. Muelas et al. (2009) used a local search mechanism to improve the solutions obtained by DE. Duarte and Marti (2009) presented an adaptive memory procedure based on scatter search and Tabu search to guide search in solving large-scale problems. Wang et al. (2009b) used an enhanced ODE based on generalized opposition-based learning (GODE) to solve scalable benchmark functions. Gardeus et al. (2009) proposed an enhanced unidimensional search (EUS) by employing an intensification mechanism.

4 Generalized opposition-based learning

4.1 Opposition-based learning

Opposition-based Learning (OBL; Tizhoosh 2005) is a new concept in computational intelligence, and has been proven to be an effective concept to enhance various optimization approaches (Rahnamayan et al. 2006a, b, 2008b; Wang et al. 2007). When evaluating a solution x to a given problem, simultaneously computing its opposite solution will provide another chance for finding a candidate solution closer to the global optimum (Rahnamayan et al. 2006a).

Opposite number (Rahnamayan et al. 2006a): Let $x \in [a, b]$ be a real number. The opposite of x is defined by:

$$x^* = a + b - x. \tag{3}$$

Similarly, the definition is generalized to higher dimensions as follows.

Opposite point (Rahnamayan et al. 2006a): Let $X = (x_1, x_2, \ldots, x_D)$ be a point in a D-dimensional space, where $x_1, x_2, \ldots, x_D \in R$ and $x_j \in [a_j, b_j]$, $j \in 1, 2, \ldots, D$. The opposite point $X^* = (x_1^*, x_2^*, \ldots, x_D^*)$ is defined by:

$$x_j^* = a_j + b_j - x_j. \tag{4}$$

By applying the definition of opposite point, the opposition-based optimization can be defined as follows.

Opposition-based optimization (Rahnamayan et al. 2006a): Let $X = (x_1, x_2, \ldots, x_D)$ be a point in a D-dimensional space (i.e., a candidate solution). Assume $f(X)$ is a fitness function, which is used to evaluate the candidate's fitness. According to the definition of the opposite point, $X^* = (x_1^*, x_2^*, \ldots, x_D^*)$ is the opposite of $X = (x_1, x_2, \ldots, x_D)$. If $f(X^*)$ is better than $f(X)$, then update X with X^*; otherwise keep the current point X. Hence, the current point and its opposite point are evaluated simultaneously to continue with the fitter one.

4.2 The concept of GOBL

Based on the concept of OBL, we proposed a generalized OBL as follows (Wang et al. 2009a). Let x be a solution in the current search space S, $x \in [a, b]$. The new solution x^* in the opposite space S^* is defined by (Wang et al. 2009a):

$$x^* = \Delta - x, \tag{5}$$

where Δ is a computable value and $x^* \in [\Delta - b, \Delta - a]$. It is obvious that the differences between the current search space S and the opposite search space S^* are the center positions of search space. Because the size of search range (indicates the size of interval boundaries) of S and S^* are $b - a$, and the center of current search space moves from $\frac{a+b}{2}$ to $\frac{2\Delta - a - b}{2}$ after using GOBL.

Similarly, the definition of GOBL is generalized to a D-dimensional search space as follows.

$$x_j^* = \Delta - x_j, \tag{6}$$

where $j = 1, 2, \ldots, D$.

By applying the GOBL, we not only evaluate the current candidate x, but also calculate its transformed candidate x^*. This will provide more chance of finding candidate solutions closer to the global optimum.

However, the GOBL could not be suitable for all kinds of optimization problems. For instance, the transformed candidate may jump away from the global optimum when solving multimodal problems. To avoid this case, a new elite selection mechanism based on population is used after the transformation, in which N_p (population size) fittest candidates will survive in the next generation. The elite selection mechanism is described in Fig. 1. Assume that the current population $P(t)$ has three candidates ($N_p = 3$), x_1, x_2 and x_3, where t is the index of generations. According to the concept of GOBL, we get three transformed candidates x_1^*, x_2^* and x_3^* in population $GOP(t)$. Then, we select three fittest candidates from $P(t)$ and $GOP(t)$ as a new population $P'(t)$. It can be seen from Fig. 1, x_1, x_2^* and x_3^* are three new members in $P'(t)$. In this case, the transformation conducted on x_1 did not provide another chance of finding a better candidate solution. With the help of the elite selection mechanism, x_1^* is eliminated in the next generation.

4.3 Four different schemes of GOBL

Let $\Delta = k(a + b)$, where k is a real number. The new GOBL model is defined by:

$$x^* = k(a + b) - x. \tag{7}$$

Let us consider four typical GOBL schemes with different values of k as follows.

1. $k = 0$ (symmetrical solutions in GOBL, GOBL-SS). The GOBL-SS model is defined by

   $$x^* = -x, \tag{8}$$

   where $x \in [a, b]$ and $x^* \in [-b, -a]$. The current solution x and opposite solution x^* are on the symmetry of origin (0).

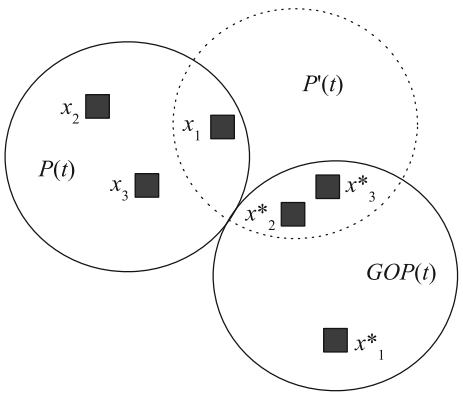

Fig. 1 The elite selection mechanism based on population

2. $k = \frac{1}{2}$ (symmetrical interval in GOBL, GOBL-SI). The GOBL-SI model is defined by

$$x^* = \frac{a+b}{2} - x, \tag{9}$$

where $x \in [a, b]$ and $x^* \in \left[-\frac{b-a}{2}, \frac{b-a}{2}\right]$. The interval of the opposite space is on the symmetry of origin.

3. $k = 1$ (opposition-based learning, OBL).
When $k = 1$, the GOBL model is equal to equation (3), where $x \in [a, b]$ and $x^* \in [a, b]$.

4. $k = rand(0, 1)$ (Random GOBL, GOBL-R)

The GOBL-SI model is defined by

$$x^* = k(a+b) - x, \tag{10}$$

where k is a random number within $[0, 1], x \in [a, b]$ and $x^* \in [k(a+b) - b, k(a+b) - a]$. The center of the opposite space is at a random position between $-\frac{a+b}{2}$ and $\frac{a+b}{2}$.

For a given problem, it is possible that the opposite candidate may jump out of the box-constraint $[X_{\min}, X_{\max}]$. When this happens, the GOBL will be invalid, because the opposite candidate is infeasible. To avoid this case, the opposite candidate is assigned to a random value as follows.

$$x^* = rand(a, b), \quad \text{If } x^* \notin [X_{\min}, X_{\max}] \tag{11}$$

where $rand(a, b)$ is a random number within $[a, b]$, and $[a, b]$ is the interval boundaries of current search space.

To illustrate the mechanism of GOBL clearly, Figs. 2 and 3 present the visualization of the four different GOBL models. By the suggestion of our previous study (Wang et al. 2009a), the GOBL-R surpasses the other three GOBL models. So the proposed approach GODE is also based on the GOBL with random k in this paper.

4.4 GOBL-based optimization

By staying within variables' interval static boundaries, we would jump outside of the already shrunken search space and the knowledge of the current converged search space would be lost. Hence, we calculate opposite particles by using dynamically updated interval boundaries $[a_j(t), b_j(t)]$ as follows (Rahnamayan et al. 2008b).

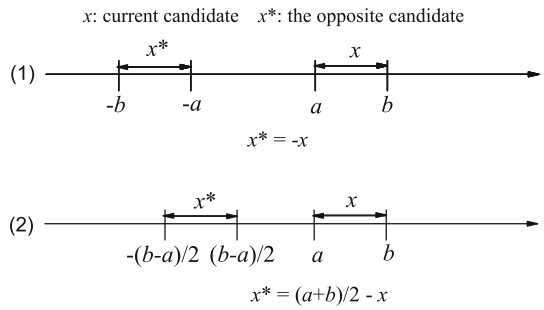

Fig. 2 (*1*) The GOBL-SS model, (*2*) the GOBL-SI model

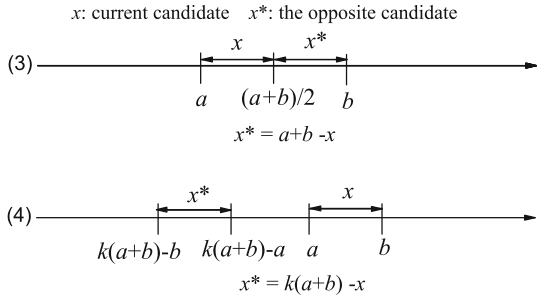

Fig. 3 (*3*) The OBL model, (*4*) the GOBL-R model

$$X^*_{i,j} = k[a_j(t) + b_j(t)] - X_{i,j} \tag{12}$$

$$a_j(t) = \min(X_{i,j}(t)), \; b_j(t) = \max(X_{i,j}(t)) \tag{13}$$

$$X^*_{i,j} = rand(a_j(t), b_j(t)), \text{ If } X^*_{i,j} \notin [X_{min}, X_{max}] \tag{14}$$

$$i = 1, 2, \ldots, N_p, \; j = 1, 2, \ldots, D, k = rand(0, 1),$$

where $X_{i,j}$ is the jth vector of the ith candidate in the population, $X^*_{i,j}$ is the opposite candidate of $X_{i,j}$, $a_j(t)$ and $b_j(t)$ are the minimum and maximum values of the jth dimension in the current search space, respectively, $rand(a_j(t), b_j(t))$ is a random number within $[a_j(t), b_j(t)]$, $[X_{min}, X_{max}]$ is the box-constraint, N_p is the population size, $rand(0,1)$ is a random number within [0,1], k is generated anew in each generation (i.e., the same value of k is used for the whole population), and $t = 1, 2, \ldots$, indicates the generations.

5 Generalized opposition-based differential evolution

In our previous work (Wang et al. 2009a), GOBL was applied to PSO and the experimental results showed that the GOBL with random k works better than the other three GOBL models in many benchmark problems. So, the proposed approach GODE is also based on the random GOBL model in this paper.

Like ODE (Rahnamayan et al. 2008b), the GODE uses similar procedures for opposition-based population initialization and dynamic opposition with GOBL. The parent DE is chosen as a parent algorithm and the proposed GOBL model is embedded in DE to improve its performance. However, the embedded strategy of GODE is different from ODE [See Table 3 in Rahnamayan et al. (2008b)]. In the ODE, the opposition is regarded as a jump or mutation strategy. GODE considers the opposition as another search mechanism besides DE, so GODE alternately executes GOBL and DE. In the ODE, the opposition occurs with a probability, and the parent DE executes every generation. But in the GODE, if the probability of opposition p_o is satisfied, then execute the GOBL; otherwise execute the classical DE.

The framework of GODE is shown in Algorithm 1, where P is the current population, GOP is the opposite

population after using GOBL, P_i is the ith individual in P, GOP_i is the ith individual in GOP, k is a random number within [0, 1], p_o is the probability of GOBL, N_p is the population size, D is the dimension size, $a_j(t), b_j(t)$ is the interval boundaries of current population, $rand(a_j(t), b_j(t))$ is a random number within $[a_j(t), b_j(t)], CR \in (0, 1)$ is the predefined crossover probability, $rand_j()$ is a random number within [0,1], FEs is the number of fitness evaluations, and MAX_FEs is the maximum number of evaluations.

Algorithm 1: The GODE Algorithm

```
1  Randomly initialize each individual in population P;
2  FEs = N_p;
3  Update the dynamic interval boundaries [a_j(t), b_j(t)] in P
   according to equation (13);
4  double k = rand(0, 1);
5  for i = 1 to N_p do
6      for j = 1 to D do
7          GOP_{i,j} = k[a_j(t) + b_j(t)] − P_{i,j};
8          if GOP_{i,j} is out of the box-constraint then
9              GOP_{i,j} = rand(a_j(t), b_j(t));
10         end
11     end
12     Calculate the fitness value of GOP_i;
13     FEs++;
14 end
15 Select N_p fittest individuals from {P, GOP} as an initial
   population P;
16 while FEs ≤ MAX_FEs do
17     if rand(0, 1) ≤ p_o then
           /* Conduct GOBL                                   */
18         Update the dynamic interval boundaries [a_j(t), b_j(t)]
           in P according to equation (13);
19         k = rand(0, 1);
20         for i = 1 to N_p do
21             for j = 1 to D do
22                 GOP_{i,j} = k[a_j(t) + b_j(t)] − P_{i,j};
23                 if GOP_{i,j} is out of the definition domain
                   then
24                     GOP_{i,j} = rand(a_j(t), b_j(t));
25                 end
26             end
27             Calculate the fitness value of GOP_i;
28             FEs++;
29         end
30         Select N_p fittest individuals from {P, GOP} as
           current population P;
31     end
32     else
           /* Execute the classical DE (rand/1/exp)          */
33         for i = 1 to N_p do
34             Randomly select 3 parents P_{i_1}, P_{i_2} and P_{i_3}
               from P, where i ≠ i_1 ≠ i_2 ≠ i_3 ;
35             V_i = P_{i_1} + F(P_{i_2} − P_{i_3});
36             U_i = P_i;
37             for j = 0; rand_j() < CR && j < D; j++ do
38                 U_{i,j} = V_{i,j};
39             end
40             Calculate the fitness value of U_i;
41             FEs++;
42             if f(U_i) ≤ f(P_i) then
43                 P_i' = U_i
44             end
45             else
46                 P_i' = P_i;
47             end
48         end
49         P = P';
50     end
51 end
```

6 Experimental studies

6.1 Experimental setup

There are 19 high-dimensional continuous optimization functions used for the following experiments. Functions F1–F6 were chosen from the first six functions provided by CEC 2008 Special Session and Competition on Large Scale Global Optimization (Tang et al. 2007). Functions F7–F11 were proposed for ISDA 2009 Workshop on Evolutionary Algorithms and other Metaheuristics for Continuous Optimization Problems—A Scalability Test (Herrera and Lozano 2009). The rest of the eight functions, F12–F19, are hybrid composition functions built by combining two functions belonging to the set functions F1–F11. The specific descriptions of F1–F19 can be found in Herrera et al. (2010b). In this paper, we focus on investigating the optimization performance of GODE on problems with and $D = 50, 100, 200, 500, 1{,}000$.

Experiments were conducted to compare four algorithms including DE (Storn and Price 1997), real-coded CHC (Eshelman and Schaffer 1993), G-CMA-ES (Auger and Hansen 2005) and the proposed GODE algorithm on the test suite. The parameter settings of DE, CHC and G-CMA-ES are described in (Herrera et al. 2010a). For GODE, the parameters N_p, F, CR, and p_o are fixed to 60, 0.5, 0.9 and 0.05, respectively. By the suggestions of Herrera et al. (2010a), the *rand/1/exp* scheme is employed in GODE.

In the experiments, and each algorithm is run 25 times for each test function. The maximum number of fitness evaluations MAX_FEs is 5,000**D*. Each run stops when the maximum number of evaluations is achieved. Throughout the experiments, the *Best*, *Median*, *Worst* and *Mean* error values over 25 runs are recorded (for a solution x, the error measure is defined as $F(x) - F(op)$, where op is the global optimum of the function). All the results below 1E–14 have been approximated to 0.0.

6.2 Results

Table 1 shows the results of GODE on functions F1–F19 with dimension $D = 50, 100, 200, 500, 1{,}000$. As seen, GODE obtains promising performance on the majority of test functions on each dimension. But GODE fails to solve F3 and its hybrid composition functions F13 and F17. It also has some difficulties when solving F4 for D = 500, 1,000, while it achieves good results for smaller dimensionality. For functions F8, GODE could hardly find reasonable solutions on each dimension. When the dimension increases to 1,000, GODE fails to solve F7 and F15. In our test, the fitness values of these two functions are larger than the maximum value (10^{308}) that double precision float

Table 1 Experimental results of GODE for $D = 50$, 100, 200, 500, 1,000

		$D = 50$	$D = 100$	$D = 200$	$D = 500$	$D = 1{,}000$
Best		0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Median	F1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Worst		0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Mean		0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Best		2.08E−02	1.83E+00	1.05E+01	4.12E+01	8.62E+01
Median	F2	3.58E−02	2.96E+00	1.46E+01	5.96E+01	8.91E+01
Worst		1.30E+00	8.25E+00	2.44E+01	6.87E+01	9.61E+01
Mean		2.57E−01	3.65E+00	1.53E+01	5.81E+01	9.02E+01
Best		2.89E+01	7.90E+01	1.79E+02	4.73E+02	9.68E+02
Median	F3	3.07E+01	8.12E+01	1.80E+02	4.76E+02	9.70E+02
Worst		3.20E+01	8.55E+01	1.82E+02	4.78E+02	9.72E+02
Mean		3.06E+01	8.14E+01	1.80E+02	4.76E+02	9.70E+02
Best		0.00E+00	2.13E−14	1.14E−13	8.30E−13	9.22E−03
Median	F4	0.00E+00	8.17E−14	4.26E−13	3.83E−12	1.08E+00
Worst		6.29E−13	2.22E−13	1.01E−12	1.92E−02	2.06E+00
Mean		1.05E−13	8.32E−14	4.17E−13	1.62E−03	1.03E+00
Best		0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Median	F5	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Worst		0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Mean		0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Best		1.12E−14	2.55E−14	5.03E−14	1.36E−13	2.85E−13
Median	F6	1.12E−14	2.55E−14	5.39E−14	1.43E−13	2.88E−13
Worst		1.48E−14	2.90E−14	5.74E−14	1.50E−13	2.92E−13
Mean		1.24E−14	2.60E−14	5.45E−14	1.43E−13	2.88E−13
Best		0.00E+00	0.00E+00	0.00E+00	0.00E+00	INF
Median	F7	0.00E+00	0.00E+00	0.00E+00	0.00E+00	INF
Worst		0.00E+00	0.00E+00	0.00E+00	0.00E+00	INF
Mean		0.00E+00	0.00E+00	0.00E+00	0.00E+00	INF
Best		9.03E−02	5.67E+01	1.87E+03	3.67E+04	1.78E+05
Median	F8	1.46E−01	7.57E+01	2.09E+03	4.00E+04	1.89E+05
Worst		3.51E−01	1.04E+02	2.36E+03	4.33E+04	1.91E+05
Mean		1.67E−01	7.53E+01	2.10E+03	3.93E+04	1.86E+05
Best		6.04E−06	1.04E−05	2.73E−05	7.00E−05	1.54E−04
Median	F9	7.06E−06	1.49E−05	3.31E−05	8.03E−05	1.71E−04
Worst		1.24E−05	1.92E−05	3.84E−05	8.33E−05	1.81E−04
Mean		7.77E−06	1.46E−05	3.23E−05	7.84E−05	1.70E−04
Best		0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Median	F10	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Worst		0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Mean		0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Best		5.65E−06	1.21E−05	2.85E−05	7.66E−05	1.67E−04
Median	F11	6.52E−06	1.64E−05	3.17E−05	8.25E−05	1.72E−04
Worst		7.44E−06	1.89E−05	3.35E−05	9.28E−05	1.84E−04
Mean		6.44E−06	1.58E−05	3.12E−05	8.25E−05	1.73E−04
Best		1.14E−13	5.34E−12	8.65E−11	6.17E−10	1.70E−09
Median	F12	1.39E−13	8.17E−12	1.16E−10	8.04E−10	1.93E−09
Worst		1.52E−13	9.31E−12	1.54E−10	8.71E−10	2.05E−09
Mean		1.33E−13	7.57E−12	1.20E−10	7.39E−10	1.87E−09
Best		2.44E+01	6.20E+01	1.37E+02	3.58E+02	7.30E+02
Median	F13	2.61E+01	6.29E+01	1.37E+02	3.59E+02	7.31E+02
Worst		2.67E+01	6.49E+01	1.39E+02	3.61E+02	7.32E+02
Mean		2.55E+01	6.32E+01	1.38E+02	3.59E+02	7.31E+02
Best		3.48E−09	2.93E−08	5.72E−08	1.45E−07	2.63E−07
Median	F14	6.50E−09	3.88E−08	6.78E−08	2.06E−07	9.95E−01
Worst		1.04E−08	6.05E−08	9.95E−01	9.95E−01	1.73E+00
Mean		6.24E−09	4.13E−08	8.17E−02	7.67E−02	6.06E−01
Best		0.00E+00	0.00E+00	0.00E+00	0.00E+00	INF
Median	F15	0.00E+00	0.00E+00	0.00E+00	0.00E+00	INF
Worst		0.00E+00	0.00E+00	0.00E+00	0.00E+00	INF
Mean		0.00E+00	0.00E+00	0.00E+00	0.00E+00	INF
Best		1.24E−10	2.62E−10	7.61E−10	1.95E−09	4.32E−09
Median	F16	1.51E−10	3.95E−10	1.00E−09	2.20E−09	4.64E−09
Worst		2.38E−10	4.94E−10	1.12E−09	2.60E−09	4.88E−09
Mean		1.57E−10	3.75E−10	9.54E−10	2.24E−09	4.59E−09
Best		6.07E−01	2.45E−02	3.59E+01	1.11E+02	2.346E+02
Median	F17	1.02E+00	1.29E+01	3.79E+01	1.12E+02	2.358E+02
Worst		2.05E+00	1.45E+01	3.86E+01	1.13E+02	2.363E+02
Mean		1.17E+00	1.11E+01	3.74E+01	1.12E+02	2.357E+02
Best		2.09E−07	9.30E−07	1.72E−06	4.86E−06	9.72E−06
Median	F18	3.05E−07	1.08E−06	2.01E−06	5.14E−06	1.14E−05
Worst		3.78E−07	1.36E−06	2.14E−06	5.19E−06	6.73E−05
Mean		2.97E−07	1.11E−06	1.91E−06	5.06E−06	3.29E−05
Best		0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Median	F19	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Worst		0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Mean		0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00

number can represent. This problem can be solved by using higher precision data types, such as "long double" in C/C++. Although we used "long double" to represent the fitness value, GODE was implemented in Microsoft Visual C++ 6.0, which uses the same size of bytes (8 bytes) to represent "double" and "long double". So, we did not list the results of these two functions for $D = 1000$.

6.3 Comparison of GODE with DE, CHC and G-CMA-ES

This section investigates the performance comparison of GODE with DE, CHC and G-CMA-ES for $D = 50$, 100, 200, 500, 1,000 (the results of G-CMA-ES for $D = 1{,}000$ are not included due to the large computation time for runs for some functions). Tables 2, 3, 4, 5 and 6 present the average error values of the above four algorithms on $D = 50$, 100, 200, 500, 1,000, respectively. The best results among the four (three for $D = 1{,}000$) algorithms are shown in *bold*.

From the results, it can be seen that both DE and GODE achieve promising results in all test cases except for F8, while CHC and G-CMA-ES fall into local optima on the majority of test functions. Compared to DE, GODE achieves better results on nine functions (F4, F6, F8, F9, F11, F12, F14, F16 and F18) in terms of their average error values, while DE performs better on one function F13. For functions

Table 2 Mean error values achieved by DE, CHC, G-CMA-ES and GODE on $D = 50$

F	Mean			
	GODE	DE	CHC	G-CMA-ES
F1	**0.00E+00**	**0.00E+00**	1.67E−11	**0.00E+00**
F2	2.57E−01	3.60E−01	6.19E+01	**2.75E−11**
F3	3.06E+01	2.89E+01	1.25E+06	**7.97E−01**
F4	**1.05E−13**	3.98E−02	7.43E+01	1.05E+02
F5	**0.00E+00**	**0.00E+00**	1.67E−03	2.96E−04
F6	**1.24E−14**	1.43E−13	6.15E−07	2.09E+01
F7	**0.00E+00**	**0.00E+00**	2.66E−09	1.01E−10
F8	1.67E−01	3.44E+00	2.24E+02	**0.00E+00**
F9	**7.77E−06**	2.73E+02	3.10E+02	1.66E+01
F10	**0.00E+00**	**0.00E+00**	7.30E+00	6.81E+00
F11	**6.44E−06**	6.23E−05	2.16E+00	3.01E+01
F12	**1.33E−13**	5.35E−13	9.57E−01	1.88E+02
F13	2.55E+01	**2.45E+01**	2.08E+06	1.97E+02
F14	**6.24E−09**	4.16E−08	6.17E+01	1.09E+02
F15	**0.00E+00**	**0.00E+00**	3.98E−01	9.79E−04
F16	**1.57E−10**	1.56E−09	2.95E−09	4.27E+02
F17	1.17E+00	**7.98E−01**	2.26E+04	6.89E+02
F18	**2.97E−07**	1.22E−04	1.58E+01	1.31E+02
F19	**0.00E+00**	**0.00E+00**	3.59E+02	4.76E+00

The bold value indicates the best mean error value among the four algorithms

Table 3 Mean error values achieved by DE, CHC, G-CMA-ES and GODE on $D = 100$

F	Mean			
	GODE	DE	CHC	G-CMA-ES
F1	**0.00E+00**	**0.00E+00**	3.56E−11	0.00E+00
F2	3.65E+00	4.45E+00	8.58E+01	**1.51E−10**
F3	8.14E+01	8.01E+01	4.19E+06	**3.88E+00**
F4	**8.32E−14**	7.96E−02	2.19E+02	2.50E+02
F5	**0.00E+00**	**0.00E+00**	3.83E−03	1.58E−03
F6	**2.60E−14**	3.10E−13	4.10E−07	2.12E+01
F7	**0.00E+00**	**0.00E+00**	1.40E−02	4.22E−04
F8	7.53E+01	3.69E+02	1.69E+03	**0.00E+00**
F9	**1.46E−05**	5.06E+02	5.86E+02	1.02E+02
F10	**0.00E+00**	**0.00E+00**	3.30E+01	1.66E+01
F11	**1.58E−05**	1.28E−04	7.32E+01	1.64E+02
F12	**7.57E−12**	5.99E−11	1.03E+01	4.17E+02
F13	6.32E+01	**6.17E+01**	2.70E+06	4.21E+02
F14	**4.13E−08**	4.79E−02	1.66E+02	2.55E+02
F15	**0.00E+00**	**0.00E+00**	8.13E+00	6.30E−01
F16	**3.75E−10**	3.58E−09	2.23E+01	8.59E+02
F17	**1.11E+01**	1.23E+01	1.47E+05	1.51E+03
F18	**1.11E−06**	2.98E−04	7.00E+01	3.07E+02
F19	**0.00E+00**	**0.00E+00**	5.45E+02	2.02E+01

The bold value indicates the best mean error value among the four algorithms

Table 4 Mean error values achieved by DE, CHC, G-CMA-ES and GODE on $D = 200$

F	Mean			
	GODE	DE	CHC	G-CMA-ES
F1	**0.00E+00**	**0.00E+00**	8.34E−01	0.00E+00
F2	1.53E+01	1.92E+01	1.03E+02	**1.16E−09**
F3	1.80E+02	1.78E+02	2.01E+07	**8.91E+01**
F4	**4.17E−13**	1.27E−01	5.40E+02	6.48E+02
F5	**0.00E+00**	**0.00E+00**	8.76E−03	**0.00E+00**
F6	**5.45E−14**	6.54E−13	1.23E+00	2.14E+01
F7	**0.00E+00**	**0.00E+00**	2.59E−01	1.17E−01
F8	2.10E+03	5.53E+03	9.38E+03	**0.00E+00**
F9	**3.23E−05**	1.01E+03	1.19E+03	3.75E+02
F10	**0.00E+00**	**0.00E+00**	7.13E+01	4.43E+01
F11	**3.12E−05**	2.62E−04	3.85E+02	8.03E+02
F12	**1.20E−10**	9.76E−10	7.44E+01	9.06E+02
F13	1.38E+02	**1.36E+02**	5.75E+06	9.43E+02
F14	**8.17E−02**	1.38E−01	4.29E+02	6.09E+02
F15	**0.00E+00**	**0.00E+00**	2.14E+01	1.75E+00
F16	**9.54E−10**	7.46E−09	1.60E+02	1.92E+03
F17	3.74E+01	**3.70E+01**	1.75E+05	3.36E+03
F18	**1.91E−06**	4.73E−04	2.12E+02	6.89E+02
F19	**0.00E+00**	**0.00E+00**	2.06E+03	7.52E+02

The bold value indicates the best mean error value among the four algorithms

Table 5 Mean error values achieved by DE, CHC, G-CMA-ES and GODE on $D = 500$

F	Mean			
	GODE	DE	CHC	G-CMA-ES
F1	**0.00E+00**	**0.00E+00**	2.84E−12	0.00E+00
F2	5.81E+01	5.35E+01	1.29E+02	**3.48E−04**
F3	4.76E+02	4.76E+02	1.14E+06	**3.58E+02**
F4	**1.62E−03**	3.20E−01	1.91E+03	2.10E+03
F5	**0.00E+00**	**0.00E+00**	6.98E−03	2.96E−04
F6	**1.43E−13**	1.65E−12	5.16E+00	2.15E+01
F7	**0.00E+00**	**0.00E+00**	1.27E−01	7.21E+153
F8	3.93E+04	6.09E+04	7.22E+04	**2.36E−06**
F9	**7.84E−05**	2.52E+03	3.00E+03	1.74E+03
F10	**0.00E+00**	**0.00E+00**	1.86E+02	1.27E+02
F11	**8.25E−05**	6.76E−04	1.81E+03	4.16E+03
F12	**7.39E−10**	7.07E−09	4.48E+02	2.58E+03
F13	**3.59E+02**	**3.59E+02**	3.22E+07	2.87E+03
F14	**7.67E−02**	1.35E−01	1.46E+03	1.95E+03
F15	**0.00E+00**	**0.00E+00**	6.01E+01	2.82E+262
F16	**2.24E−09**	2.04E−08	9.55E+02	5.45E+03
F17	1.12E+02	**1.11E+02**	8.40E+05	9.59E+03
F18	**5.06E−06**	1.22E−03	7.32E+02	2.05E+03
F19	**0.00E+00**	**0.00E+00**	1.76E+03	2.44E+06

The bold value indicates the best mean error value among the four algorithms

Table 6 Mean error values achieved by DE, CHC, G-CMA-ES and GODE on $D = 1000$

F	Mean		
	GODE	DE	CHC
F1	**0.00E+00**	**0.00E+00**	1.36E−11
F2	9.02E+01	**8.46E+01**	1.44E+02
F3	9.70E+02	**9.69E+02**	8.75E+03
F4	**1.03E+00**	1.44E+00	4.76E+03
F5	**0.00E+00**	**0.00E+00**	7.02E−03
F6	**2.88E−13**	3.29E−12	1.38E+01
F7	INF	**0.00E+00**	3.52E−01
F8	**1.86E+05**	2.46E+05	3.11E+05
F9	**1.70E−04**	5.13E+03	6.11E+03
F10	**0.00E+00**	**0.00E+00**	3.83E+02
F11	**1.73E−04**	1.35E−03	4.82E+03
F12	**1.87E−09**	1.68E−08	1.05E+03
F13	7.31E+02	**7.30E+02**	6.66E+07
F14	**6.06E−01**	6.90E−01	3.62E+03
F15	INF	**0.00E+00**	8.37E+01
F16	**4.59E−09**	4.18E−08	2.32E+03
F17	**2.357E+02**	2.358E+02	2.044E+07
F18	**3.29E−05**	2.37E−03	1.72E+03
F19	**0.00E+00**	**0.00E+00**	4.20E+03

The bold value indicates the best mean error value among the three algorithms

F2, F3 and F17, DE outperforms GODE on some dimensions. Both GODE and DE achieve the same results on six functions: F1, F5, F7, F10, F15 (except for $D = 1,000$) and F19 (except for $D = 1,000$). These comparison results clearly demonstrate that the achieved improvements of GODE are due to usage of the proposed GOBL strategy.

G-CMA-ES outperforms GODE on three functions, F2, F3 and F8. Especially for F8, only G-CMA-ES achieves excellent results, while the other three algorithms could hardly search reasonable results. Both GODE and C-CMA-ES obtain the same results for F1 and F5 on $D = 200$. For the rest of the 14 functions, GODE obtains better performance than G-CMA-ES.

Real-coded CHC is the worst algorithms among the four algorithms. It only obtains promising results on F1, which is a simple and unimodal function. For the rest of the functions, the performance of CHC is seriously affected with increasing dimensions. It suggests that CHC is not a good choice for solving large-scale problems.

6.4 Statistical comparisons of different algorithms

Non-parametric tests can be used for comparing algorithms, the results of which represent average values for each problem, in spite of the inexistence of relationships among them. It is encouraged to use non-parametric tests to analyze the results obtained by evolutionary algorithms for continuous optimization problems in multiple problem analysis (García et al. 2009a, b; Luengo et al. 2009).

To compare the performance of multiple algorithms on the test suite, average ranking of Friedman test is conducted according to the suggestions of García et al. (2010). Tables 7, 8, 9, 10 and 11 show the average ranking of GODE, DE, CHC and G-CMA-ES on $D = 50$, 100, 200, 500 and 1,000, respectively. For each dimension, the performance of the four algorithms (three for D = 1,000) can be sorted by average ranking into the following order: GODE, DE, G-CMA-ES and CHC. It means that GODE

Table 7 Average rankings of the algorithms when $D = 50$

Algorithm	Ranking
GODE	3.499999999999999
DE	3.078947368421052
G-CMA-ES	2.052631578947368
CHC	1.368421052631579

Table 8 Average rankings of the algorithms when $D = 100$

Algorithm	Ranking
GODE	3.5526315789473673
DE	3.026315789473684
G-CMA-ES	2.052631578947368
CHC	1.368421052631579

Table 9 Average rankings of the algorithms when $D = 200$

Algorithm	Ranking
GODE	3.473684210526315
DE	3.052631578947368
G-CMA-ES	2.1052631578947367
CHC	1.368421052631579

Table 10 Average rankings of the algorithms when $D = 500$

Algorithm	Ranking
GODE	3.473684210526315
DE	3.1052631578947363
G-CMA-ES	1.8947368421052628
CHC	1.5263157894736836

Table 11 Average rankings of the algorithms when $D = 1000$

Algorithm	Ranking
GODE	2.7058823529411757
DE	2.2941176470588234
CHC	1.0

and CHC are the best and worst ones among the four algorithms, respectively.

To investigate the significant differences between the behavior of two algorithms, we conduct other tests: Nemenyi's, Holm's, Shaffer's and Bergmann-Hommel's. For each test, we calculate the adjusted p values on pairwise comparisons of all algorithms.

Tables 12,13,14,15 and 16 show the results of adjusted p values on and $D = 50$, 100, 200, 500 and 1,000, respectively. The computation of the results is used by the software MULTIPLETEST package (provided on the Web site: http://sci2s.ugr.es/sicidm). Under the null hypothesis, the two algorithms are equivalent. If the null hypothesis is rejected, then the performances of these two algorithms are significantly different. In this paper, we only discuss whether the hypotheses is rejected at the 0.05 level of significance. For $D = 50$, Nemenyi's procedure rejects hypotheses 1–3, while Holm's, Shaffer's and Bergmann-Hommel's procedures reject hypotheses 1–4. For $D = 100$, Nemenyi's, Holm's and Shaffer's procedures reject hypotheses 1–3, while Bergmann-Hommel's procedure rejects hypotheses 1–4. For $D = 200$, Nemenyi's, Holm's and Shaffer's procedures reject hypotheses 1–3, while Bergmann-Hommel's procedures rejects hypotheses 1–4. For $D = 500$ and 1,000, all the four tests reject hypotheses 1–4 and 1–3, respectively.

Besides the above four tests, we also conduct Wilcoxon's test to detect significant differences between the behavior of two algorithms. Table 17 shows the p values of applying Wilcoxon's test between GODE and other three algorithms for $D = 50$, 100, 200, 500 and 1,000. The p values below 0.05 are shown in *bold*. From the results, it can be seen that GODE is significantly better than CHC and G-CMA-ES on each dimension. Though GODE is not statistically better than DE on each dimension, GODE outperforms DE according to the results of average ranking.

6.5 Computational running time

In this section, we investigate the computation running time and the computational time complexity of the proposed algorithm. For each test function, the average running time on the 25 runs are recorded. The computational conditions are listed as follows.

- System: Windows 7 professional edition
- CPU: Intel(R) Core(TM)2 Duo CPU E8500 (3.16GHz)
- RAM: 4G
- Language: C++
- Compiler: Microsoft Visual C++ 6.0
- MAX_FEs: 5000*D (D=50, 100, 200, 500 and 1,000)

The average computation running time of GODE on the test suite is given in Table 18. As seen, the computation time increases dramatically with increasing of dimensions.

Table 12 Adjusted p values for $D = 50$

i	Hypothesis	Unadjusted p	p_{Neme}	p_{Holm}	p_{Shaf}	p_{Berg}
1	GODE vs .CHC	3.59814323610516 95E-7	2.15888594166310 18E-6	2.15888594166310 18E-6	2.15888594166310 18E-6	2.15888594166310 18E-6
2	DE vs .CHC	4.43002824159746 5E-5	2.65801694495847 9E-4	2.21501412079873 27E-4	1.32900847247923 95E-4	1.32900847247923 95E-4
3	GODE vs .G-CMA-ES	5.49182033991498 1E-4	0.00329509220394 8989	0.00219672813596 59926	0.00164754610197 44946	0.00164754610197 44946
4	DE vs .G-CMA-ES	0.01427390724856 5145	0.08564344349139 087	0.04282172174569 5436	0.04282172174569 5436	0.01427390724856 5145
5	CHC vs .G-CMA-ES	0.10235752557227 876	0.61414515343367 26	0.20471505114457 5	0.20471505114457 5	0.20471505114457 5
6	GODE vs .DE	0.31477678305963 647	1.88866069835781 9	0.31477678305963 647	0.31477678305963 647	0.31477678305963 647

Table 13 Adjusted p values for $D = 100$

i	Hypothesis	Unadjusted p	p_{Neme}	p_{Holm}	p_{Shaf}	p_{Berg}
1	GODE vs .CHC	1.8408401690471716E-7	1.104504101428303E-6	1.104504101428303E-6	1.104504101428303E-6	1.104504101428303E-6
2	DE vs .CHC	7.552637614373264E-5	4.5315825686239583E-4	3.7763188071866316E-4	2.2657912843119791E-4	2.2657912843119791E-4
3	GODE vs .G-CMA-ES	3.420185079493773E-4	0.0020521110476962637	0.0013680740317975092	0.0010260555238481319	0.0010260555238481319
4	DE vs .G-CMA-ES	0.020091261342747014	0.12054756805648209	0.060273784028241045	0.060273784028241045	0.020091261342747014
5	CHC vs .G-CMA-ES	0.10235752557227876	0.6141451534336726	0.2047150511445575	0.2047150511445575	0.2047150511445575
6	GODE vs .DE	0.2089123817406994	1.2534742904441964	0.2089123817406994	0.2089123817406994	0.2089123817406994

Table 14 Adjusted p values for $D = 200$

i	Hypothesis	Unadjusted p	p_{Neme}	p_{Holm}	p_{Shaf}	p_{Berg}
1	GODE vs .CHC	5.001718578552697E-7	3.001031147131618E-6	3.001031147131618E-6	3.001031147131618E-6	3.001031147131618E-6
2	DE vs .CHC	5.795221529189861E-5	3.4771329175139164E-4	2.8976107645949307E-4	1.7385664587569582E-4	1.7385664587569582E-4
3	GODE vs .G-CMA-ES	0.0010867046009095688	0.006520227605457413	0.004346818403638275	0.0032601138027287067	0.0032601138027287067
4	DE vs .G-CMA-ES	0.02370907646413881	0.14225445878483287	0.07112722939241643	0.07112722939241643	0.02370907646413881
5	CHC vs .G-CMA-ES	0.07854585095119071	0.47127510570714426	0.15709170190238142	0.15709170190238142	0.15709170190238142
6	GODE vs .DE	0.31477678305963647	1.888660698357819	0.31477678305963647	0.31477678305963647	0.31477678305963647

Table 15 Adjusted p values for $D = 500$

i	Hypothesis	Unadjusted p	p_{Neme}	p_{Holm}	p_{Shaf}	p_{Berg}
1	GODE vs .CHC	3.3309837778945975E-6	1.9985902667367586E-5	1.9985902667367586E-5	1.9985902667367586E-5	1.9985902667367586E-5
2	DE vs .CHC	1.634535987808791E-4	9.807215926852747E-4	8.172679939043955E-4	4.903607963426374E-4	4.903607963426374E-4
3	GODE vs .G-CMA-ES	1.6345359878087968E-4	9.80721592685278E-4	8.172679939043955E-4	4.90360796342639E-4	4.90360796342639E-4
4	DE vs .G-CMA-ES	0.0038512913785902596	0.023107748271541557	0.011553874135770779	0.011553874135770779	0.0038512913785902596
5	CHC vs .G-CMA-ES	0.37907971990782696	2.2744783194469615	0.7581594398156539	0.7581594398156539	0.7581594398156539
6	GODE vs .DE	0.3790797199078275	2.274478319446965	0.7581594398156539	0.7581594398156539	0.7581594398156539

Table 16 Adjusted p values for $D = 1000$

i	Hypothesis	Unadjusted p	p_{Neme}	p_{Holm}	p_{Shaf}	p_{Berg}
1	GODE vs .CHC	6.576869920924794E-7	1.973060976277438E-6	1.973060976277438E-6	1.973060976277438E-6	1.973060976277438E-6
2	DE vs .CHC	1.613164203085826E-4	4.8394926092574776E-4	3.226328406171652E-4	1.613164203085826E-4	1.613164203085826E-4
3	GODE vs .DE	0.2299490567942142	0.6898471703826426	0.2299490567942142	0.2299490567942142	0.2299490567942142

Table 17 Wilcoxon test considering functions F1–F19 on each dimension

GODE vs.	p values				
	$D = 50$	$D = 100$	$D = 200$	$D = 500$	$D = 1000$
DE	2.79E−01	8.69E−02	1.96E−01	3.47E−01	2.79E−01
CHC	**1.32E−04**	**1.32E−04**	**1.32E−04**	**1.32E−04**	**2.93E-04**
G-CMA-ES	**3.78E−03**	**4.97E−03**	**1.48E−02**	**8.42E−03**	N/A

The bold value means the p value is below 0.05

Table 18 The average computation running time (in seconds) achieved by GODE

F	$D = 50$	$D = 100$	$D = 200$	$D = 500$	$D = 1000$
F1	0.531	1.42	4.071	18.876	68.577
F2	0.78	2.355	7.691	41.73	159.323
F3	4.009	15.351	60.294	376.351	1479.83
F4	2.277	8.346	31.543	190.632	754.059
F5	2.356	8.627	33.166	200.679	794.26
F6	2.199	7.862	29.765	177.902	702.531
F7	0.827	2.527	8.44	42.245	–
F8	0.561	1.42	4.025	18.813	66.722
F9	3.292	12.417	48.407	293.796	1167.83
F10	1.56	5.601	20.826	123.521	487.236
F11	3.385	12.87	50.045	306.696	1220.03
F12	1.326	4.571	16.536	95.597	373.215
F13	3.932	14.96	58.796	364.885	1435.55
F14	2.574	9.5	36.442	218.525	864.726
F15	1.186	3.994	13.681	78.827	–
F16	2.121	7.379	27.8	165.609	651.894
F17	3.729	13.681	53.18	327.351	1296.32
F18	3.213	11.903	45.864	278.04	1102.28
F19	1.622	5.897	20.452	128.872	470.903

To investigate how the computational running time grows in relation to the number of dimensions, we analyze the computational time complexity of GODE as follows.

Assume that $O(F(D))$ is the computational time complexity of a fitness evaluation function $F(D)$ on dimension D. For GODE, the computational time complexity is $O(D^2) + O(D) \cdot O(F(D))$. So, the computational time complexity of GODE depends on $O(F(D))$. For the 19 test functions used in the experiments, $O(F(D))$ is $O(D)$ except for F8; on this function, $O(F(D)) = O(D^2)$. So the computational time complexity of GODE is $O(D^3)$ for F8 and $O(D^2)$ for the rest of functions.

To verify the above analysis, we use two regression models, cubic ($T = a_0 + a_1 \times D + a_2 \times D^2 + a_3 \times D^3$, where $a_0, a_1, a_2, a_3 > 0$) and quadratic ($T = b_0 + b_1 \times D + b_2 \times D^2$, where $b_0, b_1, b_2 > 0$), to approximate the results of computational running time for F8 and the rest of functions, respectively. To measure how well the regression line approximates the real data points, we calculate the values of "R square", which evaluates the goodness of fit of the line. It represents the percentage variation of the data explained by the fitted line. A value of 1.000 indicates that the regression line perfectly (100%) fits the real data. Table 19 shows the values of "R square". As seen, the regression model perfectly fits the real data for each test function. It confirms that the computational time complexity of GODE is $O(D^3)$ for F8 and $O(D^2)$ for the rest of the functions. The results suggest that GODE is applicable for large-scale problems.

Table 19 Computational time complexity analysis using regression model

F	Regression model	R square
F1	Quadratic	1.000
F2	Quadratic	1.000
F3	Quadratic	1.000
F4	Quadratic	1.000
F5	Quadratic	1.000
F6	Quadratic	1.000
F7	–	–
F8	Cubic	1.000
F9	Quadratic	1.000
F10	Quadratic	1.000
F11	Quadratic	1.000
F12	Quadratic	1.000
F13	Quadratic	1.000
F14	Quadratic	1.000
F15	–	–
F16	Quadratic	1.000
F17	Quadratic	1.000
F18	Quadratic	1.000
F19	Quadratic	1.000

7 Conclusion

In this paper, a novel DE algorithm based on generalized opposition-based learning (GODE) is proposed. The GOBL is an enhanced opposition-based learning, which transforms candidates in current search space to a new search space. By simultaneously evaluating solutions in the current search space and transformed space, we can provide more chance of finding better solutions. A scalability test over 19 high-dimensional continuous optimization problems with and $D = 50$, 100, 200, 500 and 1,000, provided by the current special issue, are conducted.

GODE as well as DE obtains promising performance on the test suite (except for F8), while CHC and G-CMA-ES

could hardly obtain reasonable results on many functions. Compared to DE, GODE performs better on the majority of test functions. It demonstrates that the proposed GOBL is helpful to solve these kinds of challenging problems.

Statistical comparisons of GODE with DE, CHC and G-CMA-ES show that GODE and CHC are the best and worst ones among the four algorithms. GODE is significantly better than CHC and G-CMA-ES on each dimension. Though GODE is not significantly better than DE, GODE performs better than DE according to the results of average ranking.

However, GODE is not suitable for all kinds of problems, such as F3 and its hybrid composition functions F13 and F17. GODE also has some difficulties when solving F8, while G-CMA-ES could find the global optimum on this function.

The proposed GOBL can be embedded and investigated on other population-based algorithms. More applications on GOBL will be studied. Moreover, possible directions for our future work also include the investigation of diversity analysis, the weaknesses and limitations of GOBL, the convergence analysis, etc.

Acknowledgments The authors would like to thank Dr. D. Molina for his suggestions in the implementation of DE. The authors also thank the editor and the anonymous reviewers for their detailed and constructive comments that helped us to increase the quality of this work. This work was supported by the National Natural Science Foundation of China (No.: 61070008), and the Jiangxi Province Science & Technology Pillar Program (No.: 2009BHB16400).

References

Auger A, Hansen N (2005) A restart CMA evolution strategy with increasing population size. In: Proceedings of IEEE congress on evolutionary computation, pp 1769–1776

Bäck T (1996) Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford University Publisher, New York

Brest J, Zamuda A, Bošković B, Maučec MS, Žumer V (2008) High-dimensional real-parameter optimization using self-adaptive differential evolution algorithm with population size reduction. In: Proceedings of IEEE congress on evolutionary computation, pp 2032–2039

Dorigo M, Maniezzo V, Colorni A (1996) The ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern Part B Cybern 26:29–41

Duarte A, Marti R (2009) An adaptive memory procedure for continuous optimization. In: Proceedings of international conference on intelligent system design and applications, pp 1085–1089

Eshelman LJ, Schaffer JD (1993) Real-coded genetic algorithm and interval schemata. Found Genet Algorithms 2:187–202

García-Martínez C, Lozano M (2009) Continuous variable neighbourhood search algorithm based on evolutionary metaheuristic components: A scalability test. In: Proceedings of international conference on intelligent system design and applications, pp 1074–1079

García S, Fernández A, Luengo J (2009a) A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. Soft Comput 13:959–977

García S, Molina D, Lozano M, Herrera F (2009b) A study on the use of non-parametric tests for analyzing the evolutionary algorithms behaviour: a case study on the CEC2005 special session on real parameter optimization. J Heuristics 15:617–644

García S, Fernández A, Luengo J, Herrera F (2010) Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power. Inf Sci 180:2044–2064

Gardeux V, Chelouah R, Siarry P, Glover F (2009) Unidimensional search for solving continuous high-dimensional optimization problems. In: Proceedings of international conference on intelligent system design and applications, pp 1096–1101

Herrera F, Lozano M (2009) Workshop for evolutionary algorithms and other metaheuristics for continuous optimization problems—a scalability test, In: Proceedings of international conference on intelligent system design and applications, Pisa, Italy

Herrera F, Lozano M, Molina D (2010a) Components and parameters of DE, real-coded CHC, and G-CMAES. Technical report, University of Granada, Spain

Herrera F, Lozano M, Molina D (2010b) Test suite for the special issue of soft computing on scalability of evolutionary algorithms and other metaheuristics for large scale continuous optimization problems, Technical report, University of Granada, Spain

Hsieh S, Sun T, Liu C, Tsai S (2008) Solving large scale global optimization using improved particle swarm optimizer. In: Proceedings of IEEE congress on evolutionary computation, pp 1777–1784

Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks, pp 1942–1948

Kirkpatrick S, Gelatt CD, Vecchi PM (1983) Optimization by simulated annealing. Science 220:671–680

Larranaga P, Lozano JA (2001) Estimation of distribution algorithms—a new tool for evolutionary computation. Kluwer Academic Publishers, Boston

Luengo J, García S, Herrera F (2009) A study on the use of statistical tests for experimentation with neural networks: analysis of parametric test conditions and non-parametric tests. Expert Syst Appl 36:7798–7808

Molina D, Lozano M,Herrera F (2009) Memetic algorithm with local search chaining for continuous optimization problems: a scalability test. In: Proceedings of international conference on intelligent system design and applications, pp 1068–1073

Muelas S, LaTorre A and Peña J (2009) A memetic differential evolution algorithm for continuous optimization. In: Proceedings of international conference on intelligent system design and applications, pp 1080–1084

Rahnamayan S, Wang GG (2008) Solving large scale optimization problems by opposition-based differential evolution (ODE). Trans Comput 7(10):1792–1804

Rahnamayan S, Wang GG (2009) Toward effective initialization for large-scale search spaces. Trans Syst 8(3):355–367

Rahnamayan S, Tizhoosh HR, Salama MMA (2006a) Opposition-based differential evolution algorithms. In: Proceedings of IEEE congress on evolutionary computation, pp 2010–2017

Rahnamayan S, Tizhoosh HR, Salama MMA (2006b) Opposition-based differential evolution for optimization of noisy problems. In: Proceedings of IEEE congress on evolutionary computation, pp 1865–1872

Rahnamayan S, Tizhoosh HR, Salama MMA (2008a) Opposition versus randomness in soft computing techniques. Elsevier J Appl Soft Comput 8:906–918

Rahnamayan S, Tizhoosh HR, Salama MMA (2008b) Opposition-based differential evolution. IEEE Trans Evol Comput 12(1): 64–79

Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optimiz 11:341–359

Tang K, Yao X, Suganthan PN, Macnish C, Chen Y, Chen C, Yang Z (2007) Benchmark functions for the CEC'2008 special session and competition on high-dimensional real-parameter optimization. Technical Report, Nature Inspired Computation and Applications Laboratory, USTC, China

Tizhoosh HR (2005) Opposition-based learning: a new scheme for machine intelligence. In: Proceedings of international conference on computational intelligence for modeling control and automation, pp 695–701

Tseng L, Chen C (2008) Multiple trajectory search for large scale global optimization. In: Proceedings of IEEE congress on evolutionary computation, pp 3057–3064

Vesterstrom J, Thomsen R (2004) A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In: Proceedings of IEEE congress on evolutionary computation, pp 1980–1987

Wang Y, Li B (2008) A restart univariate estimation of distribution algorithm sampling under mixed Gaussian and Lévy probability distribution. In: Proceedings of IEEE congress on evolutionary computation, pp 3918–3925

Wang H, Liu Y, Zeng SY, Li H, Li CH (2007) Opposition-based particle swarm algorithm with Cauchy mutation. In: Proceedings of IEEE congress on evolutionary computation, pp 4750–4756

Wang H, Wu ZJ, Liu Y, Wang J, Jiang DZ, Chen LL (2009a) Space transformation search: a new evolutionary technique. In: Proceedings of world summit on genetic and evolutionary computation, pp 537–544

Wang H, Wu ZJ, Rahnamayan S, Kang LS (2009b) A scalability test for accelerated DE using generalized opposition-based learning. In: Proceedings of international conference on intelligent system design and applications, pp 1090–1095

Yang Z, Tang K, Yao X (2008) Multilevel cooperative coevolution for large scale optimization. In: Proceedings of IEEE congress on evolutionary computation, pp 1663–1670

Zhao S, Liang J, Suganthan PN, Tasgetiren MF (2008) Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization. In: Proceedings of IEEE congress on evolutionary computation, pp 3846–3853