Adaptive Differential Evolution with Variable Population Size for Solving High-Dimensional Problems

Hui Wang*†, Shahryar Rahnamayan‡ and Zhijian Wu*
*State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China
Email: wang_cug@yahoo.com.cn, zjwu9551@sina.com
†School of Information Engineering, Nanchang Institute of Technology, Nanchang 330099, China
‡Faculty of Engineering and Applied Science, University of Ontario Institute of Technology (UOIT)
2000 Simcoe Street North, Oshawa, ON L1H 7K4, Canada
Email: shahryar.rahnamayan@uoit.ca

***Abstract*—In this paper, we present a novel Differential Evolution (DE) algorithm to solve high-dimensional global optimization problems effectively. The proposed approach, called DEVP, employs a variable population size mechanism, which adjusts population size adaptively. Experiments are conducted to verify the performance of DEVP on 19 high-dimensional global optimization problems with dimensions 50, 100, 200, 500 and 1000. The simulation results show that DEVP outperforms classical DE, CHC (Crossgenerational elitist selection, Heterogeneous recombination, and Cataclysmic mutation), G-CMA-ES (Restart Covariant Matrix Evolutionary Strategy) and GODE (Generalized Opposition-Based DE) on the majority of test problems.**

***Index Terms*—Differential Evolution (DE), variable population size, global optimization, large-scale, high-dimensional.**

I. INTRODUCTION

Many real world problems can be converted into optimization problems. Therefore, optimization has been an active area of research for several decades. As the complexity of problems increase (such as high dimensions), traditional optimization algorithms struggle to achieve satisfactory solutions, and better algorithms are always required. An unconstrained optimization problem can be formulated as a minimization problem as follows:

$$\text{Min } f(x),$$

where $x = [x_1, x_2, \ldots, x_D]$ and D is the dimension size of the problem.

In the past decades, different kinds of nature-inspired algorithms have been designed and applied to solve optimization problems. Simulated Annealing (SA) [1], Evolutionary Algorithms (EAs) [2], Differential Evolution (DE) [3], Particle Swarm Optimization (PSO) [4], Ant Colony Optimization (ACO) [5], and Estimation of Distribution Algorithms (EDA) [6] are some examples among many others. However, these algorithms suffer from the *curse of dimensionality*. It 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 [7].

The DE algorithm, proposed by Storn and Price [3], is an effective, robust, and simple global optimization algorithm. According to frequently reported experimental studies, DE has shown better performance than many other evolutionary algorithms in terms of convergence speed and robustness over several benchmark functions and real-world problems [8]. Therefore, some researchers applied DE and its variants to solve some challenging problems, such as large-scale global optimization problems [9]. Yang et al. proposed a multilevel cooperative co-evolution algorithm (MLCC) based on self-adaptive neighborhood search DE to solve large-scale problems [10]. The presented results showed that MLCC could achieve promising solutions. Brest et al. introduced a population size reduction mechanism into self-adaptive DE, where the population size decreases during the evolutionary process [11]. Recently, Brest and Maučec proposed another version of jDEdynNP-F by employing three new mutation schemes [12]. Rahnamayan and Wang presented a experimental study of opposition-based DE (ODE) [13] on large-scale problems [14]. The reported results show that ODE significantly improves the performance of standard DE. Muelas et al. [15] used a local search mechanism to improve the solutions obtained by DE. Wang et al. [16], [17] used an enhanced ODE based on generalized opposition-based learning (GODE) to solve scalable benchmark functions. Zhao et al. [18] combined self-adaptive DE and multiple trajectory search (MTS) for large-scale optimization, which incorporates DE/current-to-$p$best [19] mutation strategy and hybridized with modified MTS. Wang et al. [20] proposed a sequential DE enhanced by neighborhood search (SDENS), which is based on global and local mutation [21].

In this paper, we present a novel DE algorithm to solve high-dimensional global optimization problems. The proposed approach, called DEVP, employs a variable population size mechanism, in which the population size is variable in terms of the search status of current population. In order to verify the performance of DEVP, we test it on 19 large-scale global optimization problems with $D = 50$, 100, 200, 500 and 1000. Experimental results show that DEVP outperforms DE, CHC, G-CMA-ES, and GODE on the majority of test problems.

The rest of the paper is organized as follows. In Section II,, the DE algorithm is briefly introduced. Section III describes our proposed approach, DEVP. In Section IV, the test suite, parameter settings, results and discussions are presented. Finally, the work is concluded in Section V.

II. A Brief Review of Differential Evolution

There are several variants of DE [3]. According to suggestions of [22], the *rand/1/exp* strategy shows a better performance to solve some 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

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

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 satisfies $N_p \geq 4$ 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:

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

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.

III. DE with Variable Population Size (DEVP)

The population size, N_p as well as other two control parameters, Cr and F greatly affect the performance of DE [23]. Choosing an appropriate population size highly affects the quality of the obtained solution and the efficiency of the search process. The value of N_p should not be too small in order to avoid stagnation and to provide sufficient exploration. Larger N_p increases the chance of searching better candidate solutions, but a larger population implies a larger number of function evaluations which retards the convergence rate.

According to newly reported results [24], a reasonable choice of the population size is between $N_p = 3 \times D$ and $N_p = 8 \times D$, where N_p must be at least 4 for *DE/rand/1* and 5 for *DE/best/2*, to ensure that DE will have enough mutually different vectors. However, these empirical parameter studies were conducted on several simple (low dimensions, $D \leq 20$) benchmark functions. As dimension increases, these empirical settings may not work.

In this paper, we propose an adaptive DE algorithm (DEVP) by employing a variable population size (VP) mechanism, in which the population size is variable, because the VP method can increase or decrease the number of individuals according to the search status of current population. If the fitness value of the best individual $Best$ does not improve in m generations (m is a predefined number), we consider that the population maybe trapped in local minima. So, that is a right time to add new individuals to the current population and break the stagnation. If the fitness value of the best individual $Best$ improves more than once in m generations, we consider that the current population is large enough to find better candidate solutions. To accelerate the convergence speed, we can reduce the population size by deleting the worst individual from the current population. The implementation of the VP mechanism is described as follows.

- If the fitness value of the $Best$ does not improve in m generations, then a new individual X^* is generated by:

  $$X^* = Best + F(X_{i_4} - X_{i_5}), \qquad (3)$$

  where $Best$ is the best individual found so far, X_{i_4} and X_{i_5} are two different individuals, i_4 and i_5 are two different random integers within $\{1, 2, \ldots, N_p\}$, and F is the same used as in equation (1).
  If the current population size is less than max_N_p (the maximum value of population size), then the new individual X^* will be added into the current population, and $N_p = N_p + 1$. If the current population size is up to the maximum value (N_p==max_N_p), then select a fitter individual between X^* and the worst individual X_w in current population as the new X_w.
- If the fitness value of the $Best$ improves more than once in m generation, and if N_p is more than min_N_p (the minimum value of population size), then the worst individual X_w in current population will be deleted, and $N_p = N_p - 1$.

To illustrate the mechanism of the variable population size, Fig. 1 presents the changes of population size N_p achieved by DEVP on function F_2 (Schwefel's Problem 2.21) with $D = 500$ provided by CEC-2008 [7]. The initial population size $N_p = 60$, the minimum population size $min\_N_p = 50$, and the maximum population size $max\_N_p = 100$. These

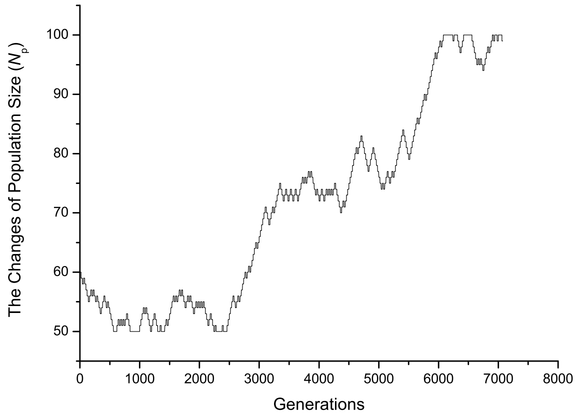


Fig. 1. The changes of population size achieved by DEVP when solving Schwefel's Problem 2.21 with $D = 500$.

parameter settings are based on our empirical studies. From Fig. 1, it can be seen that the VP mechanism dynamically adjusts the population size throughout the evolutionary process. The parameter m may affect the performance of the algorithm. According to our empirical studies, m is set to 20 in this paper.

The framework of DEVP is shown in Algorithm 1, where $P(t)$ is the current population, X_i is the ith individual in current population, $Best$ is the best individual found so far, X_w is the worst individual in current population, X^* is a newly generated individual, N_p is the population size, min_N_p is the minimum population size, max_N_p is the maximum population size, D is the dimension size, $CR \in (0, 1)$ 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 fitness evaluations.

Algorithm 1: Differential Evolution with Variable Population Size (DEVP)

```
1  Randomly initialize each individual $X_i$ in the population $P(t)$;
2  Calculate the fitness value of each $X_i$;
3  FEs = $N_p$;
4  while FEs ≤ MAX_FEs do
       /* Execute the classical DE                              */
5      for $i = 1$ to $N_p$ do
6          Randomly select 3 parents $X_{i_1}$, $X_{i_2}$ and $X_{i_3}$ from $P$, where
           $i \neq i_1 \neq i_2 \neq i_3$ ;
7          $V_i = X_{i_1}, + F(X_{i_2} - X_{i_3})$;
8          $U_i = X_i$;
9          for $j = 0$; $rand_j() < CR$ && $j < D$; $j + +$ do
10             $U_{i,j} = V_{i,j}$;
11         end
12         Calculate the fitness value of $U_i$;
13         FEs++;
14         if $f(U_i) \leq f(X_i)$ then
15             $X_i = U_i$
16         end
17         else
18             $X_i = X_i$;
19         end
20     end
       /* Variable population size mechanism                    */
21     if The $Best$ does not improve in $m$ generations then
22         Create a new individual $X^*$ according to equation (3);
23         Calculate the fitness value of $X^*$;
24         FEs++;
           /* Increase the population size                      */
25         if $N_p < max\_N_p$ then
26             $N_p = N_p + 1$;
27             $X_{N_p} = X^*$;
28         end
29         else
30             if $f(X^*) < f(X_w)$ then
31                 $X_w = X^*$;
32             end
33         end
34     end
35     if The $Best$ improves more than once in $m$ generations then
           /* Decrease the population size                      */
36         if $N_p > min\_N_p$ then
37             Delete the worst individual $X_w$ in current population;
38             $N_p = N_p - 1$;
39         end
40     end
41 end
```

IV. Experimental Verifications

A. Benchmark Functions

There are 19 scalable continuous functions used for the following experiments. Functions $F_1 - F_6$ were chosen from the first six functions provided by CEC 2008 Special Session and Competition on Large Scale Global Optimization [7]. Functions $F_7 - F_{11}$ were proposed for ISDA 2009 Workshop on Evolutionary Algorithms and other Metaheuristics for Continuous Optimization Problems–A Scalability Test [25]. The rest eight functions $F_{12} - F_{19}$ are hybrid composition functions built by combining two functions belonging to the set functions $F_1 - F_{11}$. The detailed descriptions of $F_1 - F_{19}$ can be found in [26]. In this paper, we focus on investigating the optimization performance of DEVP on problems with $D = 50$, 100, 200, 500 and 1000.

B. Parameter Settings and Involved Algorithms

Experiments were conducted to compare five algorithms including DE, CHC, G-CMA-ES, GODE and the proposed DEVP algorithm on the mentioned test suite. The algorithms and parameters settings are described as follows.

- **DE**: The classical DE model without parameter adaptation, as the parent algorithm, is considered for the performance comparisons. The crossover operator applied is *rand/1/exp*. According to the latest report in [22], the results obtained on the test suite by using *rand/1/exp* scheme are clearly better than the ones obtained by employing the *rand/1/bin* scheme. In DE, the F, Cr and N_p were fixed to 0.5, 0.9 and 60, respectively.
- **CHC**: The main idea of the CHC algorithm (Crossgenerational elitist selection, Heterogeneous recombination, and Cataclysmic mutation) [27] focuses on the combination of a high pressure selection strategy and several components including a strong diversity. The initial threshold is set at $\frac{L}{4}$, where L is the length of the string ($L = 20 * D$ in the experiments). When no offspring are inserted into the new population the threshold is reduced by 1. When the population converges or the search stops making progress (i.e., the difference threshold has dropped to zero and no new offspring are being generated which are better than any members of the parent population), the population is

re-initialized. The restart population consists of random individuals except for one instance of the best individual found so far. Finally, a real-parameter crossover operator, called BLX-α [28], is considered to substitute the original crossover operator. The instance of CHC uses BLX-α with $\alpha = 0.5$.

- **G-CMA-ES**: The G-CMA-ES is a restart Covariant Matrix Evolutionary Strategy (CMA-ES) with Increasing Population Size [29]. It detects premature convergence and launches a restart strategy that doubles the population size on each restart; by increasing the population size the search characteristic becomes more global after each restart, which empowers the operation of the CMA-ES on multimodal functions. The parameter settings of G-CMA-ES were the ones suggested by Auger and Hansen [29]. The initial solution is uniform randomly chosen from the domain and the initial distribution size (σ) is a third of the domain size.
- **GODE**: For GODE [17], we use the following parameter settings. The F, Cr and N_p were fixed to 0.5, 0.9 and 60, respectively. The probability of opposition is set to 0.05. Like DE, the *rand/1/exp* strategy is employed.
- **DEVP**: For common parameters between DEVP and DE, the same values are used. In DEVP, the *rand/1/exp* was also employed. The parameters F and Cr were fixed to 0.5 and 0.9, respectively. For the initial population size (N_p), the minimum population size (min_N_p), and the maximum population size (max_N_p), we used empirical settings in this paper, $N_p = 60$, $m = 20$, $min\_N_p = 50$ and $max\_N_p = 100$.

In the following experiments, each algorithm is run 25 times for each test function. The maximum number of fitness evaluations MAX_FEs is $5000 \times D$. Each run stops when the maximum number of evaluations is achieved. Throughout the experiments, the average of error of the best individual found in the 25 runs was recorded (For a solution x, the error measure is defined as $F(x) - F(op)$, where op is the global optimum of the function). According to the suggestions of [22], all the results below 1E–14 have been approximated to 0.0.

C. Numerical Results

The comparison results among DE, CHC, G-CMA-ES, GODE and DEVP on problems with $D = 50$, $D = 100$, $D = 200$, $D = 500$ and $D = 1000$ are presented in Tables I, II, III, IV and V, respectively (we did not include the results of G-CMA-ES for $D = 1000$ due to the large time for runs on some functions). The best results among the five (four for $D = 1000$) algorithms are shown in **bold**.

From the results, it can be seen that DEVP outperforms DE on other four algorithms (three for $D = 1000$) on 8 functions, F_4, F_6, F_9, F_{11}, F_{12}, F_{14}, F_{16} and F_{18}. For F_1, F_5, F_7 (except for $D = 1000$), F_{10}, F_{15} (except for $D = 1000$) and F_{19}, DE, GODE and DEVP can search the global optimum. For F_2 and F_8 (except for $D = 1000$), only G-CMA-ES can find promising solutions, while other three algorithms

TABLE I
RESULTS ACHIEVED BY DE, CHC, G-CMA-ES, GODE AND DEVP ON $D = 50$.

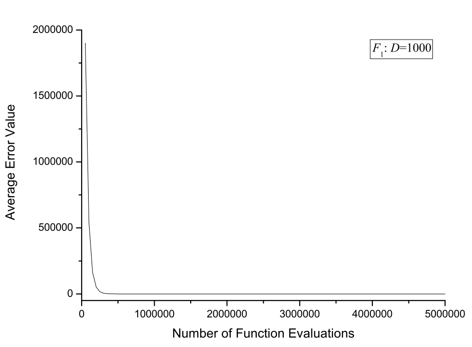
$D = 50$	DE	CHC	G-CMA-ES	GODE	DEVP
Functions	Mean	Mean	Mean	Mean	Mean
F_1	**0.00E+00**	1.67E–11	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_2	3.60E–01	6.19E+01	**2.75E–11**	2.57E–01	1.19E–01
F_3	2.89E+01	1.25E+06	**7.97E–01**	3.06E+01	3.19E+01
F_4	3.98E–02	7.43E+01	1.05E+02	1.05E–13	**0.00E+00**
F_5	**0.00E+00**	1.67E–03	2.96E–04	**0.00E+00**	**0.00E+00**
F_6	1.43E–13	6.15E–07	2.09E+01	1.24E–14	**0.00E+00**
F_7	**0.00E+00**	2.66E–09	1.01E–10	**0.00E+00**	**0.00E+00**
F_8	3.44E+00	2.24E+02	**0.00E+00**	1.67E–01	4.27E–03
F_9	2.73E+02	3.10E+02	1.66E+01	7.77E–06	**0.00E+00**
F_{10}	**0.00E+00**	7.30E+00	6.81E+00	**0.00E+00**	**0.00E+00**
F_{11}	6.23E–05	2.16E+00	3.01E+01	6.44E–06	**0.00E+00**
F_{12}	5.35E–13	9.57E–01	1.88E+02	1.33E–13	**0.00E+00**
F_{13}	**2.45E+01**	2.08E+06	1.97E+02	2.55E+01	2.88E+01
F_{14}	4.16E–08	6.17E+01	1.09E+02	6.24E–09	**1.64E–13**
F_{15}	**0.00E+00**	3.98E–01	9.79E–04	**0.00E+00**	**0.00E+00**
F_{16}	1.56E–09	2.95E–09	4.27E+02	1.57E–10	**1.62E–14**
F_{17}	**7.98E–01**	2.26E+04	6.89E+02	1.17E+00	3.11E+00
F_{18}	1.22E–04	1.58E+01	1.31E+02	2.97E–07	**1.41E–10**
F_{19}	**0.00E+00**	3.59E+02	4.76E+00	**0.00E+00**	**0.00E+00**

TABLE II
RESULTS ACHIEVED BY DE, CHC, G-CMA-ES, GODE AND DEVP ON $D = 100$.

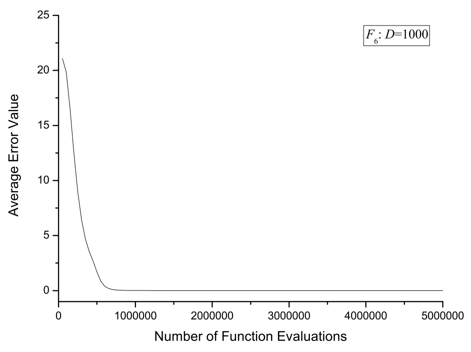
$D = 100$	DE	CHC	G-CMA-ES	GODE	DEVP
Functions	Mean	Mean	Mean	Mean	Mean
F_1	**0.00E+00**	3.56E–11	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_2	4.45E+00	8.58E+01	**1.51E–10**	3.65E+00	3.55E+00
F_3	8.01E+01	4.19E+06	**3.88E+00**	8.14E+01	8.33E+01
F_4	7.96E–02	2.19E+02	2.50E+02	**8.32E–14**	6.73E–02
F_5	**0.00E+00**	3.83E–03	1.58E–03	**0.00E+00**	**0.00E+00**
F_6	3.10E–13	4.10E–07	2.12E+01	2.60E–14	**2.07E–14**
F_7	**0.00E+00**	1.40E–02	4.22E–04	**0.00E+00**	**0.00E+00**
F_8	3.69E+02	1.69E+03	**0.00E+00**	7.53E+01	3.71E+01
F_9	5.06E+02	5.86E+02	1.02E+02	1.46E–05	**0.00E+00**
F_{10}	0.00E+00	3.30E+01	1.66E+01	**0.00E+00**	**0.00E+00**
F_{11}	1.28E–04	7.32E+01	1.64E+02	1.58E–05	**0.00E+00**
F_{12}	5.99E–11	1.03E+01	4.17E+02	7.57E–12	**5.72E–14**
F_{13}	**6.17E+01**	2.70E+06	4.21E+02	6.32E+01	6.22E+01
F_{14}	4.79E–02	1.66E+02	2.55E+02	4.13E–08	**1.23E–12**
F_{15}	**0.00E+00**	8.13E+00	6.30E–01	**0.00E+00**	**0.00E+00**
F_{16}	3.58E–09	2.23E+01	8.59E+02	3.75E–10	**8.55E–14**
F_{17}	1.23E+01	1.47E+05	1.51E+03	1.11E+01	**1.03E+01**
F_{18}	2.98E–04	7.00E+01	3.07E+02	1.11E–06	**5.88E–10**
F_{19}	**0.00E+00**	5.45E+02	2.02E+01	**0.00E+00**	**0.00E+00**

TABLE III
RESULTS ACHIEVED BY DE, CHC, G-CMA-ES, GODE AND DEVP ON $D = 200$.

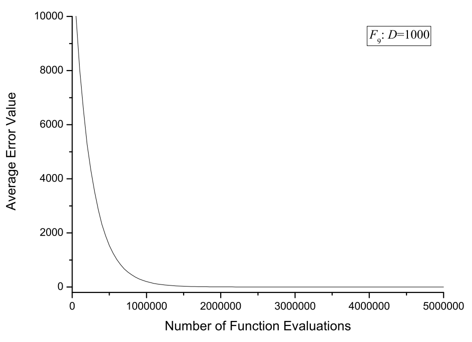
$D = 200$	DE	CHC	G-CMA-ES	GODE	DEVP
Functions	Mean	Mean	Mean	Mean	Mean
F_1	**0.00E+00**	8.34E–01	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_2	1.92E+01	1.03E+02	**1.16E–09**	1.53E+01	2.92E+01
F_3	1.78E+02	2.01E+07	**8.91E+01**	1.80E+02	1.93E+02
F_4	1.27E–01	5.40E+02	6.48E+02	4.17E–13	**0.00E+00**
F_5	**0.00E+00**	8.76E–03	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_6	6.54E–13	1.23E+00	2.14E+01	5.45E–14	**4.91E–14**
F_7	**0.00E+00**	2.59E–01	1.17E–01	**0.00E+00**	**0.00E+00**
F_8	5.53E+03	9.38E+03	**0.00E+00**	2.10E+03	1.21E+04
F_9	1.01E+03	1.19E+03	3.75E+02	3.23E–05	**0.00E+00**
F_{10}	**0.00E+00**	7.13E+01	4.43E+01	**0.00E+00**	**0.00E+00**
F_{11}	2.62E–04	3.85E+02	8.03E+02	3.12E–05	**0.00E+00**
F_{12}	9.76E–10	7.44E+01	9.06E+02	1.20E–10	**1.28E–13**
F_{13}	**1.36E+02**	5.75E+06	9.43E+02	1.38E+02	1.37E+02
F_{14}	1.38E–01	4.29E+02	6.09E+02	8.17E–02	**3.36E–12**
F_{15}	**0.00E+00**	2.14E+01	1.75E+00	**0.00E+00**	**0.00E+00**
F_{16}	7.46E–09	1.60E+02	1.92E+03	9.54E–10	**1.20E–13**
F_{17}	**3.70E+01**	1.75E+05	3.36E+03	3.74E+01	3.94E+01
F_{18}	4.73E–04	2.12E+02	6.89E+02	1.91E–06	**1.67E–09**
F_{19}	**0.00E+00**	2.06E+03	7.52E+02	**0.00E+00**	**0.00E+00**



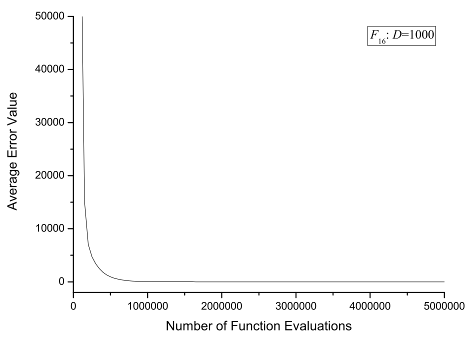
(a) F_1 $(D = 1000)$



(b) F_6 $(D = 1000)$



(c) F_9 $(D = 1000)$



(d) F_{16} $(D = 1000)$

Fig. 2. The average convergence curves of DEVP on F_1, F_6, F_9 and F_{16} with $D = 1000$.

TABLE IV
RESULTS ACHIEVED BY DE, CHC, G-CMA-ES, GODE AND DEVP ON $D = 500$.

$D = 500$ Functions	DE Mean	CHC Mean	G-CMA-ES Mean	GODE Mean	DEVP Mean
F_1	**0.00E+00**	2.84E–12	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_2	5.35E+01	1.29E+02	**3.48E–04**	5.81E+01	7.23E+01
F_3	4.76E+02	1.14E+06	**3.58E+02**	4.76E+02	4.75E+02
F_4	3.20E–01	1.91E+03	2.10E+03	**1.62E–03**	9.22E–03
F_5	**0.00E+00**	6.98E–03	2.96E–04	**0.00E+00**	**0.00E+00**
F_6	1.65E–12	5.16E+00	2.15E+01	1.43E–13	**1.37E–13**
F_7	**0.00E+00**	1.27E–01	7.21E+153	**0.00E+00**	**0.00E+00**
F_8	6.09E+04	7.22E+04	**2.36E–06**	3.93E+04	1.11E+05
F_9	2.52E+03	3.00E+03	1.74E+03	7.84E–05	**0.00E+00**
F_{10}	**0.00E+00**	1.86E+02	1.27E+02	**0.00E+00**	**0.00E+00**
F_{11}	6.76E–04	1.81E+03	4.16E+03	8.25E–05	**0.00E+00**
F_{12}	7.07E–09	4.48E+02	2.58E+03	7.39E–10	**2.65E–13**
F_{13}	**3.59E+02**	3.22E+07	2.87E+03	**3.59E+02**	3.60E+02
F_{14}	1.35E–01	1.46E+03	1.95E+03	7.67E–02	**4.94E–02**
F_{15}	0.00E+00	6.01E+01	2.82E+262	**0.00E+00**	**0.00E+00**
F_{16}	2.04E–08	9.55E+02	5.45E+03	2.24E–09	**4.68E–13**
F_{17}	**1.11E+02**	8.40E+05	9.59E+03	1.12E+02	1.29E+02
F_{18}	1.22E–03	7.32E+02	2.05E+03	5.06E–06	**4.89E–09**
F_{19}	**0.00E+00**	1.76E+03	2.44E+06	**0.00E+00**	**0.00E+00**

TABLE V
RESULTS ACHIEVED BY DE, CHC, GODE AND DEVP ON $D = 1000$.

$D = 1000$ Functions	DE Mean	CHC Mean	GODE Mean	DEVP Mean
F_1	**0.00E+00**	1.36E–11	**0.00E+00**	**0.00E+00**
F_2	**8.46E+01**	1.44E+02	9.02E+01	1.02E+02
F_3	**9.69E+02**	8.75E+03	9.70E+02	9.70E+02
F_4	1.44E+00	4.76E+03	1.03E+00	**4.52E–01**
F_5	**0.00E+00**	7.02E–03	**0.00E+00**	**0.00E+00**
F_6	3.29E–12	1.38E+01	2.88E-13	**2.86E–13**
F_7	**0.00E+00**	3.52E–01	INF	INF
F_8	2.46E+05	3.11E+05	**1.86E+05**	4.18E+05
F_9	5.13E+03	6.11E+03	1.70E–04	**0.00E+00**
F_{10}	0.00E+00	3.83E+02	**0.00E+00**	**0.00E+00**
F_{11}	1.35E–03	4.82E+03	1.73E–04	**0.00E+00**
F_{12}	1.68E–08	1.05E+03	1.87E–09	**5.39E–13**
F_{13}	**7.30E+02**	6.66E+07	7.31E+02	7.31E+02
F_{14}	6.90E–01	3.62E+03	6.06E–01	**1.52E–01**
F_{15}	**0.00E+00**	8.37E+01	INF	INF
F_{16}	4.18E–08	2.32E+03	4.59E–09	**9.69E–13**
F_{17}	**2.36E+02**	2.04E+07	2.36E+02	2.43E+02
F_{18}	2.37E–03	1.72E+03	3.29E–05	**1.17E–08**
F_{19}	**0.00E+00**	4.20E+03	**0.00E+00**	**0.00E+00**

fail to solve them. For the rest of three functions, F_3, F_{13} and F_{17}, all the five algorithms can hardly search reasonable results. For F_7 and F_{15}, DEVP as well as GODE cannot solve them when the dimension increases to 1000. The main reason is that the fitness values of these two functions are larger than the maximum value (10^{308}) that double precision float 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, DEVP 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

TABLE VI
AVERAGE RANKINGS OF THE ALGORITHMS WHEN $D = 50$.

Algorithms	Ranking
DEVP	4.18
GODE	3.82
DE	3.39
G-CMA-ES	2.24
CHC	1.37

results of these two functions for $D = 1000$. Compared to DE, DEVP performs better on the majority of test functions. It shows that the achieved improvements of DEVP are due to usage of the proposed variable population size mechanism. The average converge curves on F_1, F_6, F_9 and F_{16} with $D = 1000$ are illustrated in Fig. 2. Due to the page limitations, we only list four figures. As seen, DEVP converges very fast during the evolution.

TABLE VII
AVERAGE RANKINGS OF THE ALGORITHMS WHEN $D = 100$.

Algorithms	Ranking
DEVP	4.29
GODE	3.82
DE	3.29
G-CMA-ES	2.24
CHC	1.37

TABLE VIII
AVERAGE RANKINGS OF THE ALGORITHMS WHEN $D = 200$.

Algorithms	Ranking
DEVP	3.95
GODE	3.84
DE	3.47
G-CMA-ES	2.32
CHC	1.42

TABLE IX
AVERAGE RANKINGS OF THE ALGORITHMS WHEN $D = 500$.

Algorithms	Ranking
DEVP	3.97
GODE	3.92
DE	3.45
G-CMA-ES	2.08
CHC	1.58

TABLE X
AVERAGE RANKINGS OF THE ALGORITHMS WHEN $D = 1000$.

Algorithms	Ranking
DEVP	3.18
GODE	3.03
DE	2.74
CHC	1.06

In order to compare the performance of multiple algorithms on the test suite, non-parametric tests are used in the following experiments. By the suggestions of [30], we use average ranking of Friedman test to compare the performance of DE, CHC, G-CMA-ES, GODE and DEVP on the 19 test functions with different dimensions.

Tables VI-X show the average ranking of DEVP, GODE, DE, CHC, and G-CMA-ES on $D = 50$, 100, 200, 500 and 1000, respectively. These results are calculated by the software MULTIPLETEST package (provided on the website: http://sci2s.ugr.es/sicidm). For each dimension, the performance of the five algorithms (four for $D = 1000$) can be sorted by average ranking into the following order: DEVP, GODE, DE, G-CMA-ES, and CHC. It means that DEVP is the best one among the five algorithms.

V. CONCLUSIONS

This paper presents a novel DE algorithm to improve the performance of DE in solving high-dimensional global optimization problems. The proposed approach used a variable population size mechanism, in which the population size has been changed adaptively. If the fitness value of the best individual found so far does not improve in a predefined number of generations, then we try to increase the population size by adding a new individual to the current population; otherwise we try to decrease the population size by removing the worst individual from the current population. Experimental verifications on 19 high-dimensional problems with $D = 50$, 100, 200, 500 and 1000 show that DEVP outperforms GODE, DE, CHC and G-CMA-ES on the majority of test problems.

However, the variable population size mechanism is not always beneficial for improving the performance of DE. For functions F_2, F_3, F_8, F_{13} and F_{17}, DE performs better than DEVP in some dimensions. The effects of parameters, m, min_N_p and max_N_p, the method of generating new individuals will be investigated in our future work.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (No.: 61070008).

REFERENCES

[1] S. Kirkpatrick, C. D. Gelatt, P. M. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.
[2] T. Bäck, Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford University Publisher, New York, 1996.
[3] R. Storn and K. Price, "Differential evolution–A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optimiz.*, vol. 11, pp. 341–359, 1997.
[4] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Networks*, 1995, pp. 1942–1948.
[5] M. Dorigo, V. Maniezzo, A. Colorni, "The ant system: optimization by a colony of cooperating agents," *IEEE Trans Syst Man Cybern Part B Cybern*, vol. 26, pp. 29–41, 1996.
[6] P. Larranaga, J. A. Lozano, Estimation of distribution algorithms-A New Tool for Evolutionary Computation, Kluwer Academic Publishers, Boston, 2001.
[7] K. Tang, X. Yao, P. N. Suganthan C. Macnish, Y. Chen, C. Chen, Z. Yang, "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, 2007. http://nical.ustc.edu.cn/cec08ss.php.
[8] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in *Proc. Congr. Evol. Comput.*, 2004, vol. 2, pp. 1980–1987.

[9] S. Rahnamayan and G.G. Wang, "Toward Effective Initialization for Large- Scale Search Spaces," World Scientific and Engineering Academy and Society, /emphTransactions on Systems, vol. 8, no. 3, pp. 355–367, 2009.

[10] Z. Yang, K. Tang, X. Yao, "Multilevel cooperative coevolution for large scale optimization," in *Proc. Congr. Evol. Comput.*, 2008, pp. 1663–1670.

[11] J. Brest, A. Zamuda, B. Bošković, M. S. Maučec, V. Žumer, "High-dimensional real-parameter optimization using self-adaptive differential evolution algorithm with population size reduction," in *Proc. Congr. Evol. Comput.*, 2008, pp. 2032–2039.

[12] J. Brest, M. S. Maučec, "Self-adaptive differential evolution algorithm using population size reduction and three strategies," *Soft Comput*, 2010. DOI: 10.1007/s00500-010-0644-5

[13] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1 pp. 64–79, 2008.

[14] S. Rahnamayan, G. Gary Wang, Solving large scale optimization problems by opposition-based differential evolution (ODE)," World Scientific and Engineering Academy and Society, in *Transactions on Computers*, vol. 7, no. 10, pp. 1792–1804, 2008.

[15] S. Muelas, A. LaTorre and J. Peña, "A memetic differential evolution algorithm for continuous optimization," *Proceedings of International Conference on Intelligent System Design and Applications*, 2009, pp. 1080–1084.

[16] H. Wang, Z. Wu, S. Rahnamayan and L. Kang, "A scalability test for accelerated DE using generalized opposition-based learning," *Proc. Intelligent System Design and Applications*, 2009, pp. 1090–1095.

[17] H. Wang, Z. Wu and S. Rahnamayan, "Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems," *Soft Computing*. DOI: 10.1007/s00500-010-0642-7.

[18] S. Z. Zhao, P. N. Suganthan, S. Das, "Self-adaptive differential evolution with multi-trajectory search for large scale optimization," *Soft Comput*, 2010. DOI: 10.1007/s00500-010-0645-4

[19] J.Q. Zhang, A. C. Sanderson AC, "JADE: adaptive eifferential evolution with optional external archive," *IEEE Trans on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

[20] H. Wang, Z. Wu and S. Rahnamayan and D. Jiang, "Sequential differential evolution enhanced by neighborhood search for large scale global optimization," *Proc. Congr. Evol. Comput.*, 2010, pp. 4056–4062.

[21] S. Das, A. Abraham, U. Chakraborty and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 526–553, 2009.

[22] F. Herrera, M. Lozano and D. Molina, Components and Parameters of DE, Real-coded CHC, and G-CMAES. *Technical Report*, University of Granada, Spain, 2010.

[23] J. Brest, S. Greiner, G. Bošković, M. Mernik, V. Žumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Transactions on Evoluationary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

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

[25] F. Herrera and M. Lozano, Workshop for evolutionary algorithms and other metaheuristics for continuous optimization problems–A scalability test, *Proceedings of International Conference on Intelligent System Design and Applications*, Pisa, Italy, 2009.

[26] F. Herrera, M. Lozano and D. Molina, 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, 2010.

[27] L. J. Eshelman, "The CHC adaptive search algorithm: how to have safe search when engaging in nontraditional genetic recombination," *Foundations of Genetic Algorithms*, vol. 1, pp. 265–283, 1991.

[28] L. J. Eshelman and J. D. Schaffer, "Real-coded genetic algorithm and interval schemata," *Foundation of Genetic Algorithms*, vol. 2, pp. 187–202, 1993.

[29] A. Auger and N. Hansen, "A restart CMA evolution strategy with increasing population size," *Proceedings of IEEE Congress on Evolutionary Computation*, 2005, pp 1769–1776.

[30] S. García, A. Fernández, J. Luengo and F. Herrera, "Advanced non-parametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Information Sciences*, vol. 180, pp. 2044–2064, 2010.