An Adaptive Unified Differential Evolution Algorithm for Global Optimization

Ji Qiang* and Chad Mitchell

Lawrence Berkeley National Laboratory

1 Cycltron Road, Berkeley, CA 94720, USA

LBNL Report Number: LBNL-6853E

Abstract

In this paper, we propose a new adaptive unified differential evolution algorithm for single-objective global optimization. Instead of the multiple mutation strategies proposed in conventional differential evolution algorithms, this algorithm employs a single equation unifying multiple strategies into one expression. It has the virtue of mathematical simplicity and also provides users the flexibility for broader exploration of the space of mutation operators. By making all control parameters in the proposed algorithm self-adaptively evolve during the process of optimization, it frees the application users from the burden of choosing appropriate control parameters and also improves the performance of the algorithm. In numerical tests using thirteen basic unimodal and multimodal functions, the proposed adaptive unified algorithm shows promising performance in comparison to several conventional differential evolution algorithms.

Keywords: differential evolution, evolutionary optimization

1. Introduction

Differential evolution is a simple yet efficient population-based, stochastic, evolutionary algorithm. It was first introduced by Storn and Price in 1995 as a

*Corresponding author
Email address: `jqiang@lbl.gov,chadmitchell@lbl.gov` (Ji Qiang* and Chad Mitchell)

global optimization algorithm to optimize real parameter, real-valued functions and has generated much interest since then [1, 2, 3, 4, 6, 7, 8]. In a number of studies, the differential evolution algorithm performed effectively in comparison to several stochastic optimization methods such as simulated annealing, controlled random search, evolutionary programming, the particle swarm method, and genetic algorithms [2, 9, 10, 11]. It has been successfully used in a variety of applications and demonstrated its effectiveness.

The differential evolution algorithm uses the scaled differences of parent solutions to form a mutation operator to generate next-generation candidates for global optimization. In the paper of Storn and Price, five different mutation strategies were proposed [12]. Several additional variants of these mutation strategies were later proposed to improve the properties of the mutation operation, e.g. to make it rotationally invariant [13]. The use of multiple mutation strategies makes the differential evolution algorithm complicated to implement and use appropriately. In this paper, we propose an adaptive unified differential evolution (AuDE) algorithm for global optimization. This algorithm uses only a single mutation expression, but encompasses almost all commonly-used mutation strategies as special cases. It is mathematically simpler than the conventional algorithm with its multiple mutation strategies, and also provides users the flexibility to explore new combinations of conventional mutation strategies during optimization. By making the control parameters in the mutation and crossover stages self-adaptive, it also sets the user free from choosing an appropriate set of control parameters for each optimization problem.

The rest of the paper is organized as follows: in Section 2, the standard differential evolution algorithm and its multiple mutation strategies are reviewed. In Section 3, the adaptive unified differential evolution algorithm is discussed. In Section 4, numerical benchmarks with conventional mutation strategies are presented. Conclusions are given in Section 5.

2. Standard differential evolution algorithm

The differential evolution algorithm starts with a population initialization. A group of NP solutions in the control parameter space is randomly generated to form the initial population. This initial population can be generated by sampling from a uniform distribution within the parameter space if no prior information about the optimal solution is available, or by sampling from a known distribution (e.g., Gaussian) if some prior information is available.

After initialization, the differential evolution algorithm updates the population from one generation to the next generation until reaching a convergence condition or until the maximum number of function evaluations is reached. At each generation, the update step consists of three operations: mutation, crossover, and selection. The mutation and the crossover operations produce new candidates for the next generation population and the selection operation is used to select from among these candidates the appropriate solutions to be included in the next generation.

2.1. Mutation Strategies

During the mutation operation stage, for each population member (target vector) $\vec{x}_i,\ i = 1, 2, 3, \cdots, NP$ at generation G, a new mutant vector $\vec{v}_i$ is generated by following a mutation strategy. Some commonly used conventional mutation strategies are [1, 2, 7]:

$$\begin{aligned}
\text{DE/rand/1}: \ & \vec{v}_i = \vec{x}_{r_1} + F_{xc}(\vec{x}_{r_2} - \vec{x}_{r_3}) && (1)\\
\text{DE/rand/2}: \ & \vec{v}_i = \vec{x}_{r_1} + F_{xc}(\vec{x}_{r_2} - \vec{x}_{r_3}) + F_{xc}(\vec{x}_{r_4} - \vec{x}_{r_5}) && (2)\\
\text{DE/best/1}: \ & \vec{v}_i = \vec{x}_b + F_{xc}(\vec{x}_{r_1} - \vec{x}_{r_2}) && (3)\\
\text{DE/best/2}: \ & \vec{v}_i = \vec{x}_b + F_{xc}(\vec{x}_{r_1} - \vec{x}_{r_2}) + F_{xc}(\vec{x}_{r_3} - \vec{x}_{r_4}) && (4)\\
\text{DE/current-to-best/1}: \ & \vec{v}_i = \vec{x}_i + F_{cr}(\vec{x}_b - \vec{x}_i) + F_{xc}(\vec{x}_{r_1} - \vec{x}_{r_2}) && (5)\\
\text{DE/current-to-best/2}: \ & \vec{v}_i = \vec{x}_i + F_{cr}(\vec{x}_b - \vec{x}_i) + F_{xc}(\vec{x}_{r_1} - \vec{x}_{r_2})\\
& \qquad + F_{xc}\ (\vec{x}_{r_3} - \vec{x}_{r_4}) && (6)\\
\text{DE/current-to-rand/1}: \ & \vec{v}_i = \vec{x}_i + F_{cr}(\vec{x}_{r_1} - \vec{x}_i) + F_{xc}\ (\vec{x}_{r_2} - \vec{x}_{r_3}) && (7)\\
\text{DE/current-to-rand/2}: \ & \vec{v}_i = \vec{x}_i + F_{cr}(\vec{x}_{r_1} - \vec{x}_i) + F_{xc}\ (\vec{x}_{r_2} - \vec{x}_{r_3})\\
& \qquad + F_{xc}\ (\vec{x}_{r_4} - \vec{x}_{r_5}) && (8)\\
\text{DE/rand-to-best/1}: \ & \vec{v}_i = \vec{x}_{r_1} + F_{cr}(\vec{x}_b - \vec{x}_i) + F_{xc}\ (\vec{x}_{r_2} - \vec{x}_{r_3}) && (9)\\
\text{DE/rand-to-best/2}: \ & \vec{v}_i = \vec{x}_{r_1} + F_{cr}(\vec{x}_b - \vec{x}_i) + F_{xc}(\vec{x}_{r_2} - \vec{x}_{r_3})\\
& \qquad + F_{xc}(\vec{x}_{r_4} - \vec{x}_{r_5}) && (10)
\end{aligned}$$

where the integers r_1, r_2, r_3, r_4 and r_5 are chosen randomly from the interval $[1, NP]$ and are different from the current index i, F_{xc} is a real scaling factor that controls the amplification of the differential variation, $\vec{x}_b$ is the best solution among the NP population members at the generation G, and F_{cr} is a weight for the combination between the original target vector and the best parent vector or the random parent vector. The strategy DE/rand/1 proposed in the original paper of Storn and Price is the most widely used mutation strategy. It has stronger exploration capability but may converge more slowly than the strategies that use the best solution from the parent generation. The strategy DE/rand/2 uses two difference vectors and may result in a better mutation solution than the strategies that use one difference vector [14]. The strategies DE/best/1 and DE/best/2 take advantage of the best solution found in the parent population and have a faster convergence towards the optimal solution [15]. However, they may become stuck at a local minimum point during multimodal function optimization. The DE/current-to-best/1 and DE/current-to-best/2 strategies provide a compromise between exploitation of the best solution and exploration of the parameter space. The DE/current-to-rand/1 and DE/current-to-rand/2 mutation strategies are rotation-invariant strategies [13]. The DE/rand-to-best/ strategies are similar to the DE/current-to-best/ strategies, but larger diversity of the mutant vector is attained by using a randomly selected parent vector instead of the current target parent vector.

2.2. Crossover

A crossover operation between the new generated mutant vector $\vec{v}_i$ and the target vector $\vec{x}_i$ is used to further increase the diversity of the new candidate solution. This operation combines the two vectors into a new trial vector $\vec{U}_i, i = 1, 2, 3, \cdots, NP$, where the components of the trial vector are obtained from the components of $\vec{v}_i$ or $\vec{x}_i$ according to a crossover probability Cr. In the binomial crossover scheme, for a D dimensional control parameter space, the new trial vector $\vec{U}_i$, $i = 1, 2, \cdots, NP$ is generated using the following rule:

$$\vec{U}_i = (u_{i1}, u_{i2}, \cdots, u_{iD}) \tag{11}$$

$$u_{ij} = \begin{cases} v_{ij}, & \text{if } \ \text{rand}_j \leq Cr \ \text{ or } j = \text{mbr}_i \\ x_{ij}, & \text{otherwise} \end{cases} \tag{12}$$

where rand_j is a randomly chosen real number in the interval $[0, 1]$, and the index mbr_i is a randomly chosen integer in the range $[1, D]$. This ensures that the new trial vector contains at least one component from the new mutant vector.

2.3. Selection

The new generated trial solution $\vec{U}_i$ is checked against the boundary in the control parameter space. If the solution is out of the boundary, a new trial solution is generated from a random sampling within the boundary.

The selection operation in DE is based on a one-to-one comparison. The new trial solution $\vec{U}_i$ is checked against the original target parent solution $\vec{x}_i$. If the new trial solution produces a better objective function value, it will be put into the next generation $(G + 1)$ population. Otherwise, the original parent is kept in the next generation population.

The above procedure is repeated for all NP parents to generate the next generation population. Many generations are used to attain the final global optimal solution.

3. The adaptive unified differential evolution algorithm

Ten different mutation strategies have been proposed for the standard differential evolution algorithm (Eqs. 1-10). While DE/rand/1/bin has been widely used, DE/best/1/bin was shown to have better performance in a number of optimization test examples [15]. Meanwhile, DE/best/2/bin was suggested as a highly beneficial method in the ICEC'96 contest [16]. The presence of multiple mutation strategies can complicate the use of the differential evolution algorithm. A combination of different mutation strategies was also proposed in a number of studies [14, 17, 18, 19]. In this paper, we propose a single mutation expression that can unify most conventional mutation strategies used by the differential evolution algorithm. This single unified mutation expression can be written as:

$$\vec{v}_i = \vec{x}_i + F_1(\vec{x}_b - \vec{x}_i) + F_2(\vec{x}_{r_1} - \vec{x}_i) + F_3(\vec{x}_{r_2} - \vec{x}_{r_3}) + F_4(\vec{x}_{r_4} - \vec{x}_{r_5}) \quad (13)$$

Here, the second term on the right-hand side of equation (13) denotes the contribution from the best solution found in the current generation, the third term denotes the rotationally invariant contribution from the random solution [13], and the fourth and fifth terms are the same terms as those used in the original differential evolution algorithm to account for the contributions from the difference of parent solutions. Those last three terms divert the mutated solution away from the best solution and help to improve the algorithm's exploration of the decision parameter space. The four parameters F_1, F_2, F_3 and F_4 are the weights from each contribution. This unified mutation expression represents a combination of exploitation (from the best found solution) and exploration (from the random solutions) to generate a new mutant solution.

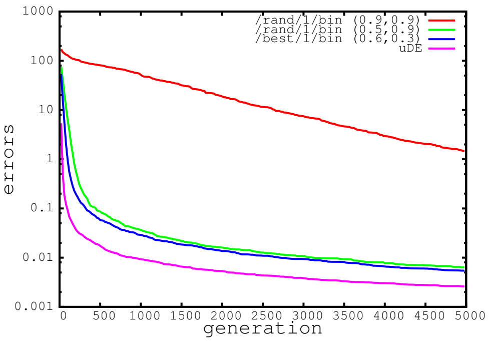

Figure 1: Evolution of the average error in the noisy quartic function with 50 dimensions.

From the above equation, one can see that for $F_1 = 0$, $F_2 = 1$, and $F_4 = 0$, this equation reduces to DE/rand/1; for $F_1 = 0$, $F_2 = 1$, and $F_3 = F_4$, it reduces to DE/rand/2; for $F_1 = 1$, $F_2 = 0$, and $F_4 = 0$, it reduces to DE/best/1; for $F_1 = 1$, $F_2 = 0$, and $F_3 = F_4$, it reduces to DE/best/2; for $F_2 = 0$ and $F_4 = 0$, it reduces to DE/current-to-best/1; for $F_2 = 0$ and $F_3 = F_4$, it reduces to DE/current-to-best/2; for $F_1 = 0$, and $F_4 = 0$, it reduces to DE/current-to-rand/1; for $F_1 = 0$, and $F_3 = F_4$, it reduces to DE/current-to-rand/2; for $F_2 = 1$, and $F_4 = 0$, it reduces to DE/rand-to-best/1; for $F_2 = 1$, and $F_3 = F_4$, it reduces to DE/rand-to-best/2. Using the single equation (13), the ten mutation strategies of the standard differential evolution algorithm can be written as a single mutation expression. This new expression provides an opportunity to explore more broadly the space of mutation operators. By using a different set of parameters F_1, F_2, F_3, F_4, a new mutation strategy can be achieved. For example, Fig. 1 shows a plot of errors in the objective function value from a numerical test by using a new mutation strategy with $F_1 = F_2 = F_4 = 0.2, F_3 = 0.5$ and $Cr = 0.8$ from the unified differential evolution algorithm (pink). Also shown is the standard differential evolution algorithm DE/rand/1 with $F = 0.9, Cr = 0.9$ (red), DE/rand/1 with $F = 0.5, Cr = 0.9$ (green), and DE/best/1 with $F = 0.6, Cr = 0.3$ (blue). Here the test function is a 50 dimensional quartic function with noise as described in the following section (Eq.3). It is seen that by expanding the space of mutation strategies (using the unified mutation strategy), it is possible to find a better optimized solution than the conventional standard differential evolution algorithm in some applications. Moreover, by adaptively adjusting these parameters during the optimization evolution, multiple mutation strategies and their combinations can be used during different stages of optimization. Thus, the unified mutation expression has the virtue of mathematical simplicity and also provides the user with flexibility for broader exploration of different mutation strategies.

The unified mutation strategy provides a method to combine and use different mutation strategies. However, choosing appropriate control parameters F_1, F_2, F_3, F_4 can be challenging and time consuming for application users. Also, using a set of fixed control parameters does not necessarily lead to the best performance of the algorithm since different mutation strategies might have superior performance at different points during the process of evolutionary optimization. A self-adaptive method to select these control parameters will free the user from such a burden and also improve the performance of the algorithm. In previous studies, a number of parameter control methods were used for the conventional differential evolution algorithm [9, 10, 11, 14, 20, 21]. In this study, we follow a self-adaptive method used in the reference [10] to evolve the five control parameters (F_1, F_2, F_3, F_4 and Cr) dynamically in the proposed unified differential evolution algorithm. This self-adaptive scheme is simple to implement and achieved a good performance in a number of benchmark tests.

During the mutation stage, the self-adaptive method used in this study assumes that at generation G, each individual solution $\vec{x}_i^G,\ i = 1, 2, 3, \cdots, NP$ has a set of control parameters $F_{1,i}^G, F_{2,i}^G, F_{3,i}^G, F_{4,i}^G$ and Cr_i^G associated with it. Before generating a new mutant solution using the unified differential evolution expression (13), a new set of control parameters $F_{1,i}^{G+1}, F_{2,i}^{G+1}, F_{3,i}^{G+1}, F_{4,i}^{G+1}$ and Cr_i^{G+1} are calculated as:

$$F_{j,i}^{G+1} = \begin{cases} F_{jmin} + r_{j1}(F_{jmax} - F_{jmin}), & \text{if} \quad r_{j2} < \tau_j \\ F_{1,i}^G, & \text{otherwise} \end{cases} \tag{14}$$

$$Cr_i^{G+1} = \begin{cases} Cr_{min} + r_3(Cr_{max} - Cr_{min}), & \text{if } \; r_4 < \tau_5 \\ Cr_i^G, & \text{otherwise} \end{cases} \qquad (15)$$

where $r_{j1}, r_{j2}, j = 1, 2, 3, 4,\ r_3,\ r_4$ are uniform random values in the interval $[0, 1]$, F_{jmin} and F_{jmax} for $j = 1, 2, 3, 4$ are the minimum and the maximum allowed values of those control parameters, Cr_{min} and Cr_{max} are the minimum and the maximum cross-over probability, and $\tau_j, j = 1, 2, 3, 4, 5$ represents the probability to use a new value or to keep the old value for the j^{th} control parameter. The values of τ_j are normally kept small so that better control parameters associated with surviving solutions will be reused to generate the new trial solution. In this paper, we set $\tau_j = 0.1$ following the reference [10]. We also did numerical tests with $\tau_j = 0.05, 0.15$, and 0.2 using the benchmark functions in the following section and did not see significant difference in most functions. The values of F_{jmin} and F_{jmax} are set to 0 and 1 respectively in this paper. We also set $Cr_{min} = 0$ and $Cr_{max} = 1$. The selection of these values is based on the consideration that the various conventional differential evolution mutation strategies of Eq. (1-10) can be covered by the settings of those parameters, and in the literature, F_3 and F_4 are rarely greater than one. The new set of control parameters (Eqs. 14-15) are used to generate the mutant solution in Eq. 13. The initial values of these control parameters are uniform random values between the minimum and the maximum values. A pseudo-code of the adaptive unified differential evolution algorithm is given as follows:

4. numerical benchmark

4.1. Test functions

Thirteen well-known test functions that have been widely used in studies of global optimization in evolutionary computation are used in this paper for numerical benchmarking [10, 11, 15, 22]. These functions include unimodal and multimodal functions, separable and non-separable functions, and smooth and non-smooth functions. They can also be scaled for arbitrary dimensions so

Pseudo-code for the AuDE Algorithm

```
**Step 1**: Generate a set of initial control parameters $(F_1, F_2, F_3, F_4, Cr)$
  from random uniform sampling in the interval [0,1]. Generate a set
  of initial population members by randomly sampling $NP$ points
  within the feasible control parameter space $\vec{x}$ and evaluate their objective
  function values $f(\vec{x})$. Set the generation number $G = 0$.
**Step 2**: While the stopping criterion is not satisfied, Do:
  For i = 1 to $NP$ (for each target parent solution $\vec{x}_i$):
    *Step 2.1*: Mutation
      Find a set of control parameters (for j=1,2,3,4):
      $F_{j,i}^{G+1} = \begin{cases} F_{jmin} + r_{j1}(F_{jmax} - F_{jmin}), & \text{if} \quad r_{j2} < \tau_j \\ F_{1,i}^{G}, & \text{otherwise} \end{cases}$
      $Cr_i^{G+1} = \begin{cases} Cr_{min} + r_3(Cr_{max} - Cr_{min}), & \text{if} \quad r_4 < \tau_5 \\ Cr_i^{G}, & \text{otherwise} \end{cases}$
      Find a mutant solution vector using the uDE mutation strategy:
      $\vec{v}_i = \vec{x}_i + F_1^{G+1}(\vec{x}_b - \vec{x}_i) + F_2^{G+1}(\vec{x}_{r_1} - \vec{x}_i) + F_3^{G+1}(\vec{x}_{r_2} - \vec{x}_{r_3}) + F_4^{G+1}(\vec{x}_{r_4} - \vec{x}_{r_5})$
    *Step 2.2*: Crossover
      Generate a new trial solution $\vec{U}_i(u_{i1}, u_{i2}, \cdots, u_{iD})$ through
      a binominal crossover scheme:
      $u_{ij} = v_{ij}$ if $\text{rand}_{ij}[0,1] \leq Cr_i^{G+1}$ *or* $j = j_{\text{rand}}$,
      otherwise $u_{ij} = x_{ij}$.
    *Step 2.3*: Selection
      Evaluate the objective function of the trial solution $f(\vec{U}_i)$.
      If $f(\vec{U}_i) \leq f(\vec{x}_i)$, then $\vec{x}_{i,G+1} = \vec{U}_i$,
      else $\vec{x}_{i,G+1} = \vec{x}_{i,G}$.
  End For
  $G = G + 1$
End While
```

that the scaling of the proposed algorithm can be tested by choosing different dimension numbers. In this study, we have used 10, 30 and 50 dimensions in the numerical benchmark tests. These functions are given below:

(1) Sphere function

$$F_{\mathrm{sph}}(\vec{x}) = \sum_{i=1}^{N} x_i^2; \quad -100 \le x_i \le 100;$$

(2) Schwefel's problem 1.2

$$F_{\mathrm{sch2}}(\vec{x}) = \sum_{j=1}^{N} \left(\sum_{i=1}^{j} x_i \right)^2; \quad -100 \le x_i \le 100;$$

(3) Quartic function with noise

$$F_{\mathrm{qrt}}(\vec{x}) = \sum_{i=1}^{N} i x_i^4 + \mathrm{rand}[0,1); \quad -1.28 \le x_i \le 1.28;$$

(4) Rosenbrock's function

$$F_{\mathrm{ros}}(\vec{x}) = \sum_{i=1}^{N-1} \left(100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \right);$$

$$-100 \le x_i \le 100;$$

(5) Ackley's function

$$F_{\mathrm{ack}}(\vec{x}) = 20 + \exp(1) - 20 \exp\left(-0.2 \sqrt{\frac{1}{N} \sum_{i=1}^{N} x_i^2} \right)$$

$$- \exp\left(\frac{1}{N} \sum_{i=1}^{N} \cos(2\pi x_i) \right); \quad -32 \le x_i \le 32;$$

(6) Griewank's function

$$F_{\mathrm{grw}}(\vec{x}) = \sum_{i=1}^{N} \frac{x_i^2}{4000} - \prod_{i=1}^{N} \cos \frac{x_i}{\sqrt{i}} + 1;$$

$$-600 \le x_i \le 600;$$

(7) Rastrigin's function

$$F_{\mathrm{ras}}(\vec{x}) = 10N + \sum_{i=1}^{N} \left(x_i^2 - 10\cos(2\pi x_i) \right);$$

$$-5 \le x_i \le 5;$$

(8) Schwefel's function

$$F_{\mathrm{sch}}(\vec{x}) = 418.9829N - \sum_{i=1}^{N} (x_i \sin(\sqrt{|x_i|}));$$

$$-500 \le x_i \le 500;$$

(9) Salomon's function

$$F_{\mathrm{sal}}(\vec{x}) = -\cos\left(2\pi\sqrt{\sum_{i=1}^{N} x_i^2}\right)$$

$$+0.1\sqrt{\sum_{i=1}^{N} x_i^2} + 1; \quad -100 \le x_i \le 100;$$

(10) Whitely's function

$$F_{\mathrm{wht}}(\vec{x}) = \sum_{j=1}^{N}\sum_{i=1}^{N}\left(\frac{y_{i,j}^2}{4000} - \cos(y_{i,j}) + 1\right);$$

where $y_{i,j} = 100(x_j - x_i^2)^2 + (1 - x_i)^2$;

$$-100 \le x_i \le 100;$$

(11) Weierstrass's function

$$F_{\mathrm{wst}}(\vec{x}) = \sum_{i=1}^{N} w(x_i, 0.5, 3, 20) - Nw(0, 0.5, 3, 20);$$

where

$$w(x_i, a, b, m) = \sum_{k=0}^{m} a^k \cos(2\pi b^k (x_i + 0.5));$$

$$-0.5 \le x_i \le 0.5;$$

(12) Generalized penalized function I

$$F_{\mathrm{pn1}}(\vec{x}) = \frac{\pi}{N}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{N-1}(y_i-1)^2[1\right.$$

$$\left.+10\sin^2(\pi y_{i+1})] + (y_N-1)^2\right\}$$

$$+\sum_{i=1}^{N} u(x_i, 10, 100, 4); \qquad -50 \le x_i \le 50;$$

(13) Generalized penalized function II

$$F_{\mathrm{pn2}}(\vec{x}) = \frac{\pi}{N}\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{N-1}(x_i-1)^2[1\right.$$

$$\left.+\sin^2(3\pi x_{i+1})] + (x_N-1)^2[1+\sin^2(2\pi x_N)]\right\}$$

$$+\sum_{i=1}^{N} u(x_i, 5, 100, 4); \qquad -50 \le x_i \le 50;$$

where in (12) and (13), $\quad y_i = 1 + \frac{1}{4}(x_i+1) \quad$ and

$$u(x_i,a,k,m)$$

$$= \begin{cases} k(x_i-a)^m, & x_i > a_i \\ 0, & -a \le x_i \le a, \\ k(-x_i-a)^m, & x_i < -a. \end{cases}$$

The sphere function is a continuous, unimodal and separable function. The Schewefel's problem 1.2 is a non-separable unimodal function. The noisy quartic function is a unimodal function with random noise in the objective value. The Rosenbrock's function with dimension greater than three is a multimodal and non-separable problem. The global minimum lies inside a parabolic shaped flat valley. The Ackely's function is also a multimodal non-separable problem and has many local minima and a narrow global minimum. The Rastrigin's function is a complex multimodal separable problem with many local minima. The Griewank's function is a multimodal non-separable function. The Salomon's function and the Whitely's function are non-separable and highly multimodal with many local minima. The Weierstrass function is a multimodal, nonseparable continuous function that is differentiable only on a set of points. The generalized penalized functions are multimodal, nonseparable, irregular, and discontinuous functions. The number of local minima from test functions (5) to (13) increases quickly with the problem dimension. The exact global minimum for all of these problems is achieved for an objective function value of zero.

4.2. Benchmark results

To test the proposed AuDE algorithm, we carried out numerical optimization using the 13 benchmark objective functions listed in the above subsection with dimensions $N = 10$, 30, and 50 respectively. We compare the proposed AuDE algorithm with two widely used conventional DE algorithms, DE/rand/1/bin ($F = 0.9$, $Cr = 0.9$) [2, 23], DE/rand/1/bin ($F = 0.5$, $Cr = 0.9$) [10, 11, 15], and DE/best/1/bin ($F = 0.6$, $Cr = 0.3$) [15], and an adaptive conventional differential evolution algorithm (jDE) [10]. The maximum number of function evaluations is set as $10,000N$. The population size (NP) for the 10, 30, and 50 dimensional problems is set as 50, 60, and 100, respectively. Each optimization is performed for 25 different random seeds. The average objective function value and its standard deviation at the end of the maximum number of function evaluations is reported in Table I for each of the 10 dimensional objective functions, in Table II for the 30 dimensional functions, and in Table III for the 50 dimensional functions. The minimum average objective value for each problem is shown in bold font. It is seen that among the conventional differential evolution algorithms, the algorithms DE/best/1/bin and DE/rand/1/bin $F = 0.5, Cr = 0.9$ have similar performance. The performance of the conventional differential evolution algorithm DE/rand/1/bin is quite sensitive to the choice of the scaling parameter F. The use of $F = 0.5$ for DE/rand/1/bin shows better performance than the case with $F = 0.9$. The adaptive unified differential evolution algorithm (AuDE) proposed in this paper shows quite good performance in these tests. Its performance is comparable or superior to the other algorithms shown in seven out of thirteen test examples in 10, 30 and 50 dimensions. The performance is better than the conventional differential evolution algorithms DE/best/1 and DE/rand/1. It is also slightly better than the conventional adaptive differential evolution algorithm (jDE), which achieves six of the best solutions out of thirteen test examples in the 50 dimensional test.

Table 1: Performance comparison of different DE strategies for $N = 10$.

Function	rand/1/bin (0.9,0.9)		rand/1/bin (0.5,0.9)		best/1/bin (0.6,0.3)		jDE		AuDE	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
$F_{\rm sph}$	2.61E-13	2.31E-13	2.88E-83	5.42E-83	**0.00**	0.00	2.46E-83	5.17E-83	3.15E-76	8.22E-76
$F_{\rm sch2}$	7.60E-06	5.15E-06	**1.15E-53**	1.42E-53	2.06E-20	5.05E-20	1.43E-20	2.02E-20	7.98E-24	1.63E-23
$F_{\rm qrt}$	8.15E-03	2.79E-03	8.15E-04	3.64E-04	**6.56E-04**	1.74E-04	1.10E-03	3.70E-04	7.88E-04	3.17E-04
$F_{\rm ros}$	1.08E-03	8.00E-04	3.08E+00	1.24E+00	2.11E+00	1.29E+00	2.00E-09	9.60E-09	**1.40E-14**	5.98E-14
$F_{\rm ack}$	2.88E-07	1.34E-07	3.00E-15	6.96E-16	3.11E-15	0.00	2.54E-15	1.30E-15	**2.11E-15**	1.60E-15
$F_{\rm grw}$	2.10E-01	1.46E-01	1.56E-02	1.25E-02	3.08E-2	1.86E-02	**0.00**	0.00	1.96E-02	1.71E-02
$F_{\rm ras}$	8.88E+00	6.11E+00	7.64E-01	8.57E-01	1.11E+00	8.59E-01	**2.07E-53**	5.63E-53	5.62E-24	1.94E-23
$F_{\rm sch}$	2.58E+01	9.18E+01	**1.27E-04**	3.15E-12	5.21E+01	8.26E+01	**1.27E-04**	3.15E-12	**1.27E-04**	3.15E-12
$F_{\rm sal}$	1.07E-01	1.05E-02	**9.99E-02**	1.32E-09	**9.99E-02**	2.28E-09	**9.99E-02**	0.00E+00	**9.99E-02**	0.00E+00
$F_{\rm wht}$	3.88E+01	1.70E+01	1.26E+01	1.36E+01	4.16E+00	4.06E+00	**1.57E+00**	1.33E+00	6.07E+00	2.39E+00
$F_{\rm wst}$	2.52E-03	7.86E-04	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
$F_{\rm pn1}$	2.06E-13	1.90E-13	**4.71E-32**	0.00E+00	**4.71E-32**	0.00	**4.71E-32**	0.00	**4.71E-32**	0.00
$F_{\rm pn2}$	1.33E-13	9.85E-14	**1.35E-32**	2.60E-40	**1.35E-32**	2.60E-40	**1.35E-32**	2.60E-40	**1.35E-32**	2.60E-40

Table 2: Performance comparison of different DE strategies for $N = 30$.

Function	rand/1/bin (0.9,0.9)		rand/1/bin (0.5,0.9)		best/1/bin (0.6,0.3)		jDE		AuDE	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
$F_{\rm sph}$	1.19E+00	8.96E-01	1.35E-71	1.84E-71	**0.00**	0.00	3.07E-74	7.57E-74	3.51E-84	5.92E-84
$F_{\rm sch2}$	4.10E+03	1.93E+03	**4.95E-12**	6.41E-12	4.94E+01	2.76E+01	9.39E-04	9.74E-04	1.14E-09	1.10E-09
$F_{\rm qrt}$	6.73E-02	1.82E-02	2.25E-03	4.84E-04	2.12E-03	6.42E-04	2.98E-03	8.14E-04	**1.43E-03**	4.49E-04
$F_{\rm ros}$	2.44E+03	1.92E+03	1.28E+01	9.16E+00	2.06E+01	3.65E+00	1.49E+00	3.27E+00	**9.57E-01**	1.70E+00
$F_{\rm ack}$	6.32E-01	3.79E-01	**3.11E-15**	0.00	7.51E-15	2.08E-15	**3.11E-15**	0.00E+00	**3.11E-15**	0.00E+00
$F_{\rm grw}$	8.31E-01	1.38E-01	5.92E-04	2.01E-03	2.17E-03	3.96E-03	**0.00E+00**	0.00E+00	8.87E-04	3.03E-03
$F_{\rm ras}$	1.27E+02	3.72E+01	1.22E+01	4.11E+00	9.39	3.23	**3.98E-02**	1.95E-01	2.20E+01	2.08E+00
$F_{\rm sch}$	5.79E+03	1.40E+03	7.16E+02	7.96E+02	1.99E+02	1.20E+02	**3.82E-04**	7.28E-12	1.10E+02	1.35E+02
$F_{\rm sal}$	1.57E+00	2.41E-01	1.82E-01	3.66E-02	1.96E-01	1.96E-02	1.96E-01	1.96E-02	**1.80E-01**	3.99E-02
$F_{\rm wht}$	7.04E+06	3.17E+07	2.71E+02	1.42E+02	**2.14E+02**	1.27E+02	2.17E+02	8.19E+01	2.74E+02	5.54E+01
$F_{\rm wst}$	2.12E+00	7.44E-01	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
$F_{\rm pn1}$	2.29E-01	2.89E-01	4.15E-03	2.03E-02	**1.57E-32**	4.86E-40	**1.57E-32**	4.86E-40	**1.57E-32**	4.86E-40
$F_{\rm pn2}$	1.35E+00	1.42E+00	4.40E-04	2.15E-03	**1.35E-32**	2.60E-40	**1.35E-32**	2.60E-40	**1.35E-3**	2.60E-40

In Figs. 1 and 2, we show the evolution of the objective function value of the 13 test functions for the algorithms shown in Table 3 with dimension $N = 50$. At each generation, the objective function value has been averaged over 25 random seeds. It is seen that the adaptive unified differential algorithm performs quite well in these test examples.

In the proposed adaptive uDE, a set of control parameters is associated with

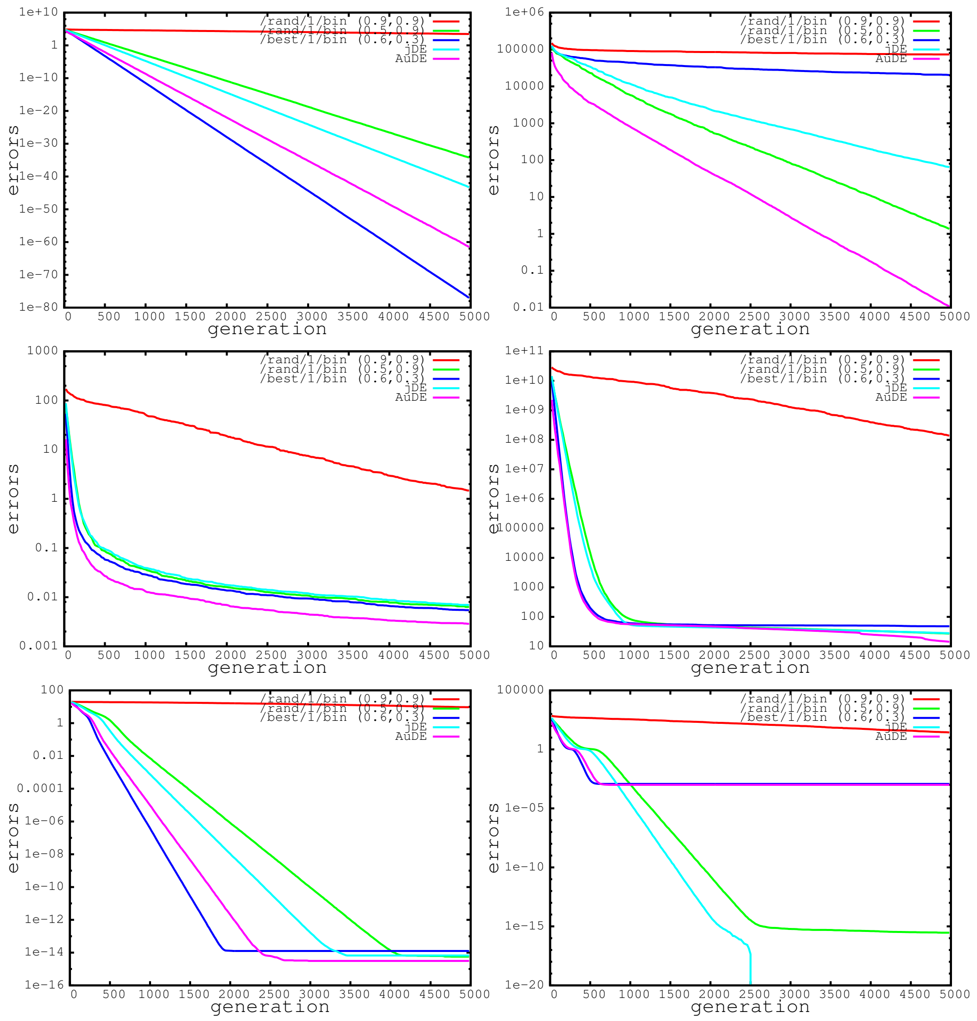

Figure 2: Evolution of the average error in the first six test functions: top left is the sphere function, top right is the Schwefel's problem 1.2, middle left is the noisy quartic function, middle right is the Rosenbrock's function, bottom left is the Ackley's function, and bottom right is the Griewank's function.

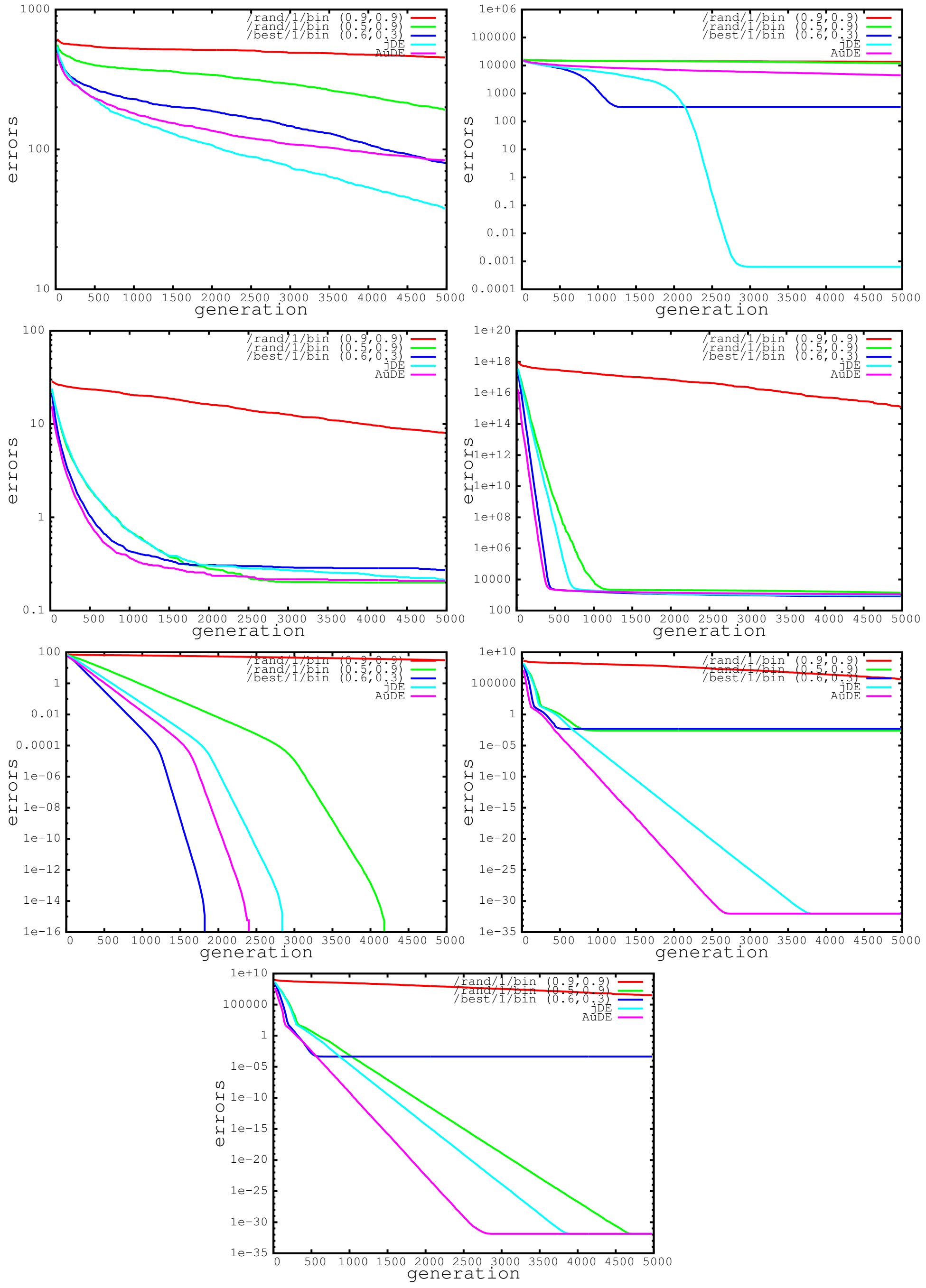

Figure 3: Evolution of the average error in the second seven test functions: top left is the Rastrigin's function, top right is the Schwefel's function, middle left is the Salomon's function, middle right is the Whitely's function, bottom left is the Weierstrass's function, bottom right is the generalized penalized function I, and last plot is the generalized penalized function II.

Table 3: Performance comparison of different DE strategies for $N = 50$.

Function	rand/1/bin (0.9,0.9)		rand/1/bin (0.5,0.9)		best/1/bin (0.6,0.3)		jDE		AuDE	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
F_{sph}	2.85E+03	9.21E+02	4.84E-35	8.77E-35	**4.87E-78**	5.76E-78	4.01E-44	3.22E-44	1.92E-62	4.42E-62
F_{sch2}	7.35E+04	5.83E+03	1.28E+00	4.69E-01	2.03E+04	4.17E+03	6.33E+01	2.08E+01	**1.03E-02**	6.24E-03
F_{qrt}	1.43E+00	8.18E-01	6.34E-03	1.18E-03	5.38E-03	1.02E-03	6.80E-03	1.29E-03	**2.86E-03**	7.20E-04
F_{ros}	1.36E+08	6.28E+07	2.69E+01	1.59E+01	4.79E+01	1.78E+01	2.78E+01	1.21E+01	**1.41E+01**	1.66E+01
F_{ack}	9.29E+00	8.04E-01	5.67E-15	1.60E-15	1.26E-14	3.29E-15	6.66E-15	0.00E+00	**3.11E-15**	0.00E+00
F_{grw}	2.66E+01	8.29E+00	2.93E-16	6.18E-17	1.18E-03	3.42E-03	**0.00E+00**	0.00E+00	9.86E-04	2.70E-03
F_{ras}	4.53E+02	4.00E+01	1.91E+02	5.33E+01	7.90E+01	5.55E+01	**3.74E+01**	4.03E+00	8.33E+01	6.73E+00
F_{sch}	1.37E+04	5.61E+02	1.22E+04	1.63E+03	3.27E+02	2.54E+02	**6.36E-04**	4.49E-12	4.49E+03	3.70E+02
F_{sal}	8.01E+00	7.13E-01	**2.00E-01**	1.97E-08	2.72E-01	4.44E-02	2.08E-01	2.71E-02	2.08E-01	2.71E-02
F_{wht}	1.23E+15	1.38E+15	1.39E+03	3.36E+02	**8.23E+02**	2.67E+02	9.71E+02	3.12E+01	1.11E+03	5.38E+01
F_{wst}	3.13E+01	3.09E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
F_{pn1}	4.63E+05	5.99E+05	2.49E-03	1.22E-02	4.98E-03	1.69E-02	**9.42E-33**	0.00E+00	**9.42E-33**	0.00E+00
F_{pn2}	3.06E+06	3.67E+06	1.39E-32	1.34E-33	4.39E-04	2.15E-03	**1.35E-32**	2.60E-40	**1.35E-32**	2.60E-40

each individual solution. It is interesting to see how the parameters associated with the best solution in each generation evolve through the process of optimization. Figure 4 shows the evolution of the five control parameters for the 50-dimensional sphere function test and using 25 random seeds. Figure 5 shows the evolution of the five control parameters for the 50-dimensional generalized penalized function II test and 25 random seeds. It is seen that the control parameter F_1 scatters randomly between 0 and 1 through the optimization and among the 25 runs. The parameter F_2 stays below 0.5, F_3 and F_4 stay below 0.3 for a large fraction of runs in the sphere function test and within the first 3000 generations of the penalized function II test. The parameter Cr stays mostly beyond 0.2 in the sphere function test and scatters randomly between 0 and 1 in the penalized function II test. These plots suggest that smaller values of F_2, F_3, F_4 might be helpful to improve the best solution in each generation during the process of optimization.

5. Conclusion

In this paper, we proposed a single unified mutation expression for the differential evolution algorithm. In comparison to the standard differential evolution algorithm that normally contains multiple mutation strategies, this method has the advantages of both mathematical simplicity and flexibility for exploring the

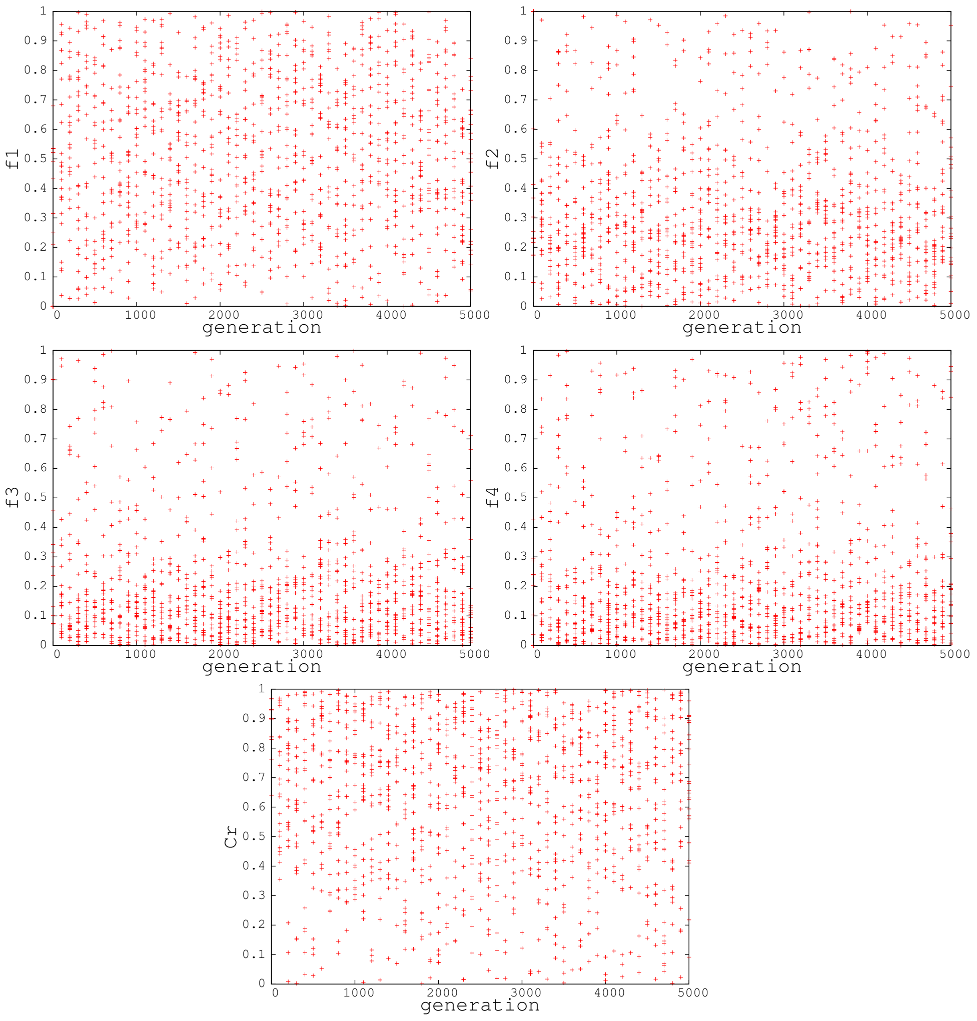

Figure 4: Evolution of the control parameters associated with the best solution in each generation in the 50-dimensional sphere function test and using 25 random seeds.

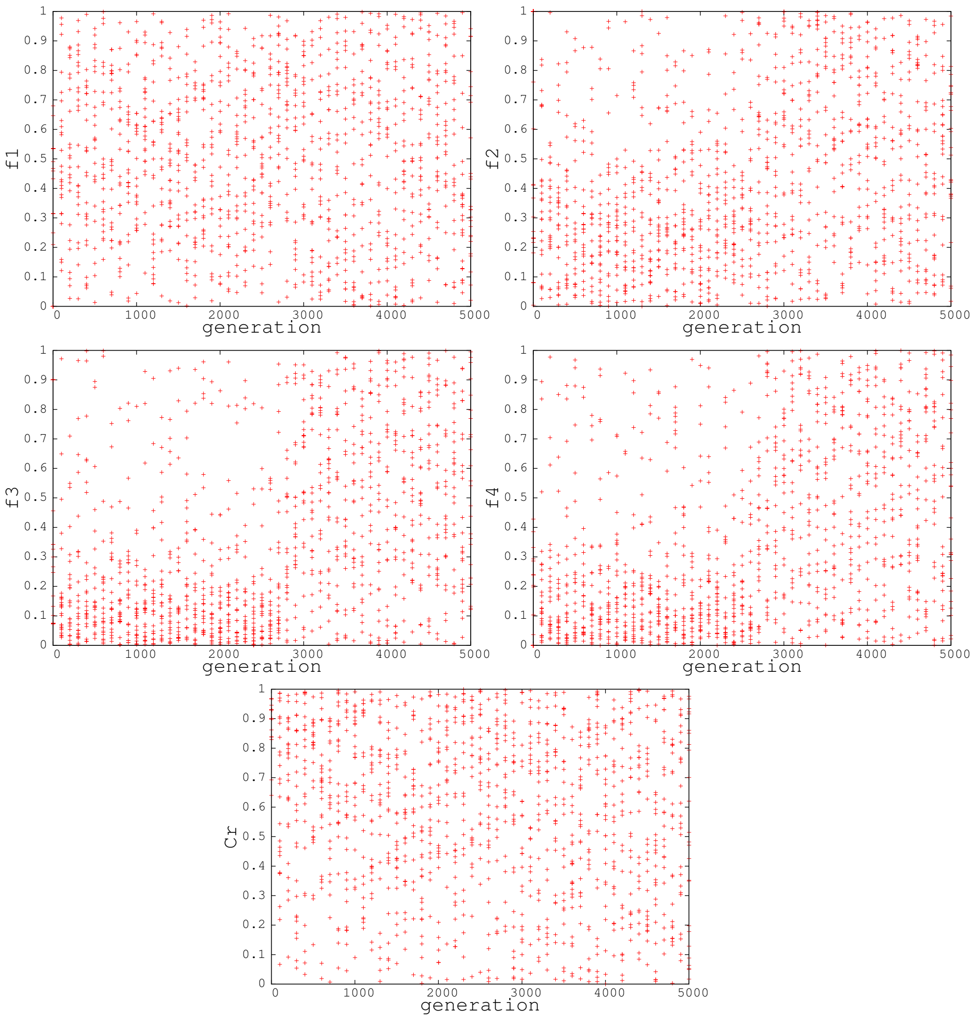

Figure 5: Evolution of the control parameters associated with the best solution in each generation in the 50-dimensional generalized penalized function II test and 25 random seeds.

space of mutation operators. The disadvantage of this algorithm is that it involves more control parameters. Instead of three control parameters, F_{cr}, F_{xc} and Cr, as in the conventional DE, the unified DE has five control parameters, F_1, F_2, F_3, F_4 and Cr. The performance of the algorithm will depend on the choice of these parameters. As a result, we proposed an adaptive scheme to select a set of control parameters for each individual solution of the population and dynamically evolve those control parameters from one generation to the next generation. This self-adaptive unified differential evolution scheme explores a broader mutation strategy space than the conventional differential evolution scheme during the process of optimization. It not only frees the application user from the burden of choosing appropriate control parameters but also improves the performance of the algorithm. In numerical benchmark tests using thirteen unimodal and multimodal functions, the adaptive unified differential evolution algorithm proposed here shows a good convergence rate in comparison with the standard differential evolution algorithms and a self adaptive standard differential evolution algorithm, and scales well with 10, 30, and 50 dimensional test problems. Parameter control in evolutionary algorithms is still an active research area [24, 25]. In future work, we will continue to explore different adaptive methods to further improve the performance of the proposed unified differential evolution algorithm.

Acknowledgment

The work was supported by the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. This research used computer resources at the National Energy Research Scientific Computing Center and at the National Center for Computational Sciences.

References

[1] R. Storn and K. Price, "Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces," ICSI, USA,

Tech. Rep. TR-95-012, 1995.

[2] R. Storn and K. Price, "Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces," Journal of Global Optimization 1a1:341-359, (1997).

[3] K. Price, R. Storn, and J. Lampinen, *Differential Evolution - A Practical Approach to Global Optimization*, Berlin, Germany: Springer, 2005.

[4] U. K. Chakraborty (ed.), *Advances in Differential Evolution*, Springer, Berlin, 2008.

[5] A. Qing, *Differential Evolution: Fundamentals and Applications in Electrical Engineering*, John Wiley, New York, 2009.

[6] J. Zhang and A. C. Sanderson, *Adaptive Differential Evolution: A Robust Approach to Multimodal Problem Optimization*, Springer, Berlin, Germany, 2009.

[7] F. Neri and V. Tirronen, "Recent advances in differential evolution: a survey and experimental analysis," Artif. Intell. Rev. 33, p. 61, 2010.

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

[9] M. M. Ali and A. Torn, "Population set based global optimization algorithms: Some modifications and numerical studies," Comput. Oper. Res., vol. 31, no. 10, pp. 1703-1725, 2004.

[10] J. Brest, S. Greiner, B. Bovskovic, M. Mernik, and V. Zumer, "Self adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," IEEE Transactions on Evolutionary Computation, vol. 10, no. 6, pp. 646-657, 2006.

[11] J. Zhang and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," IEEE Transactions on Evolutionary Computation, vol. 13, no. 5, pp. 945-958, 2009.

[12] Storn R (1996b) On the usage of differential evolution for function optimization. In: Proceedings of the IEEE biennial conference of the North American fuzzy information processing society, pp 519-523.

[13] K. V. Price, "An introduction to differential evolution," in New Ideas in Optimization, D. Corne, M. Dorigo, and V. Glover, Eds. London, U.K.: McGraw-Hill, 1999, pp. 79-108.

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

[15] E. Mezura-Montes, J. Velazquez-Reyes, and C. A. Coello Coello, "A comparative study of differential evolution variants for global optimization," in Proc. Genet. Evol. Comput. Conf., 2006, pp. 485- 492.

[16] K. Price and R. Storn, "Minimizeing the real functions of the ICEC'96 contest by differential evolution," IEEE International Conference on Evolutionary Computation, 1996, pp.842-844.

[17] Q. K. Pan, P. N. Suganthan, L. Wang, L. Gao, and R. Mallipeddi, "A differential evolution algorithm with self-adapting strategy and control parameters," Computers & Operations Research, vol. 38, no. 1, pp. 394-408, 2011.

[18] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," Applied Soft Computing 11, p. 1679, 2011.

[19] Y. Wang, Z. Cai, Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," IEEE transcations on Evolutionary Computation, vol. 15, no.1, p. 55, 2011.

[20] J. Liu and J. Lampinen, “A fuzzy adaptive differential evolution algorithm,” Soft Computing, A Fusion of Foundations, Methodologies and Applications, vol. 9, no. 6, pp. 448-642, 2005.

[21] W. Gong, Z. Cai, C. X. Ling, and H. Li, “Enhanced differential evolution with adaptive strategies for numerical optimization,” IEEE Transactions on Systems, Man, and Cybernetics - Part B, vol. 41, no. 2, pp. 397-413, 2011.

[22] N. Noman and H. Iba, “Accelerating differential evolution using an adaptive local search,” IEEE Transactions on Evolutionary Computation, vol. 12, no. 1, pp. 107-125, 2008.

[23] J. Ronkkonen, S. Kukkonen, and K. Price, in Proceedings IEEE Congress on Evolutionary Computation, CEC 2005, vol. 1, p. 506-513, Edinburgh, UK (2005).

[24] A. E. Eiben, R. Hinterding, and Z. Michalewicz, “Parameter control in evolutionary algorithms,” IEEE Trans. Evol. Comput., vol. 3, no. 2, pp. 124-141, 1999.

[25] T. Chiang, C. Chen, and Y. Lin, “Parameter Control Mechanisms in Differential Evolution: A Tutorial Review and Taxonomy,” in IEEE Symposium Series on Computational Intelligence, Singapore, April 2013.