Improved Differential Evolution for Single Objective Optimization Problems

Ricardo de A. Araújo[1], **Germano C. Vasconcelos**[1], **Tiago A. E. Ferreira**[1]

[1]Center for Informatics – Federal University of Pernambuco
Av. Prof. Luiz Freire, s/n, CDU, 50732-970, Recife - PE - Brazil

{raa,gcv,taef}@cin.ufpe.br

Abstract. *This work proposes the Improved Differential Evolution (IDE) for single objective optimization problems with continuous variables. The proposed IDE uses improved Differential Evolution (DE) operators (mutation and crossover) in order to explore the state space of possible solutions with greater effectiveness, as well as to accelerate its convergence speed. Furthermore, an experimental analysis with the proposed IDE is presented using six well-known benchmark problems of single objective optimization. The results clearly show that the proposed IDE converges faster, as well as find better solutions than the Standard DE (SDE).*

1. Introduction

Evolutionary Algorithms (EAs) are a powerful class of stochastic optimization algorithms and have been widely used to solve complex problems which cannot be solved analytically. The most popular EA is the Genetic Algorithm (GA) [Holland 1975]. The GAs were developed by Holland [Holland 1975] motivated by Charles Darwin Evolution Theory [Darwin 1859]. The main goal of [Holland 1975] was to find ways in which the mechanisms of natural adaptation might be integrated into computer systems. The GAs are used successfully in several kind of real-world applications due to its high search power in state spaces, being widely applied to optimization and machine learning problems. The GAs allow the population evolution towards sub-optimal or optimal solution to a given problem by performing a search in the multiple trajectory simultaneously, being appropriate to parallel computing [Franklin 2001].

Recently, many works have been proposed to improve the GAs in order to accelerate its convergence speed and to prevent the algorithms from getting trapped into local minima of the objective function [Bersini and Renders 1994, Yao and Liu 1996, Gen and Cheng 1997, Chellapilla 1998, Angelov 2001, Leung and Wang 2001, Leung et al. 2003, Tsai et al. 2004]. In this way, these modified GAs can find optimal or close-to-optimal solutions of a given problem using an individuals' small number in the population and having high speed convergence due to its optimal genetic operators.

Despite the improvements achieved so far, GAs in optimizations problems are still considered time consuming and further efforts are made to overcome this limitation. An interesting evolutionary strategy has been proposed in order to overcome this GAs limitation [Storn and Price 1995, Storn 1999], referred to as Differential Evolution (DE), which is a simple and yet powerful EA, that has demonstrated good convergence properties and outperforms other well known EAs [Storn and Price 1995, Storn 1999, Brest et al. 2006]. In the DE [Storn and Price 1995, Storn 1999], all individuals have the same selection probability, which means that the selection scheme does not depend on fitness function.

Such as in the GAs case, interesting DE improvements were also proposed in the literature to improve state space exploration and acceleration of convergence [Ali and Torn 2004, Vesterstroem and Thomsen 2004, Sun et al. 2005, Liu and Lampinen 2005, Brest et al. 2006]. These works show that this kind of EA provides promising results, and can quickly find optimal or close-to-optimal solutions to optimizations problems.

In this paper, the Improved Differential Evolution (IDE) is presented to solve single objective optimization problems with continuous variables. It is based on the Differential Evolution (DE) proposed by [Storn and Price 1995,Storn 1999], but uses improved DE operators (mutation and crossover) in order to explore the state space more efficiently, as well as to quickly find optimal or close-to-optimal solutions, accelerating its convergence speed. Besides, six well-known benchmark problems of single objective optimization are used to assess performance of the proposed IDE. Since each benchmark problem has a very large number of local minima, finding satisfactory solutions becomes a great challenge. The results clearly show that the proposed IDE finds optimal or close-to-optimal solutions for all tested benchmark problems, and presents faster convergence speed when compared to the Standard DE (SDE).

2. Fundamentals

2.1. Problem Description

Let $f(\underline{x})$ be an objective function, where $\underline{x} = (x_1, x_2, \ldots, x_N) \in \mathbb{R}^N$, and let $\underline{l} = (l_1, l_2, \ldots, l_N)$ and $\underline{r} = (r_1, r_2, \ldots, r_N)$ be the feasible space solution of $f(\underline{x})$ in $\mathbb{R}^N$. This defines a single objective optimization problem as the minimization of an objective function $f(\underline{x})$ subject to the domain $\underline{l} \leq \underline{x} \leq \underline{r}$ (the domain of each x_i is denoted by $l_i \leq x_i \leq r_i$). Therefore, the main goal is to apply some technique in order to find the optimal or close-to-optimal $\underline{x}$ vector values to minimize $f(\underline{x})$.

2.2. Differential Evolution

In this section, it is presented a brief description of Differential Evolution (DE) procedure, which is illustrated in Figure 1. More details will be supplied as follows. For further details see [Storn and Price 1995, Storn 1999].

```
DifferentialEvolutionProcedure() {
  $G = 0$;
  initialize $\underline{x}_G$;        // $\underline{x}_G$: population at generation G
  evaluate $f(\underline{x}_G)$;    // $f(\cdot)$: evaluate function or fitness
  while ( not termination condition ) {
    for each individual ($\underline{x}_{k,G}$) from population $\underline{x}_G$ {
      perform the mutation operator (generating $\underline{v}_{i,G+1}$ by Equation (1));
      perform the crossover operator (generating $\underline{u}_{i,G+1}$ by Equation (2));
      perform the selection operator (generating $\underline{x}_{k,G+1}$ by Equation (4));
    }
    $G = G + 1$;
  }
}
```

Figure 1. Differential evolution procedure.

The DE was created around the nineties, being developed by Storn [Storn and Price 1995,Storn 1999] in the attempt to solve the Chebychev Polynomial problem. There are several variants of the DE [Storn and Price 1995, Storn 1999]. In this work, we use the DE scheme which can be classified according to [Storn and Price 1995, Storn 1999] as the *DE/rand/1/bin* strategy. This strategy is widely used in practice [Storn and Price 1995, Liu and Lampinen 2005, Sun et al. 2005, Brest et al. 2006] and will be described as follows.

The DE is a novel parallel direct search method [Storn and Price 1995]. It uses a population of NP parameter vectors $\underline{x}_{k,G}, k = 1, 2, \ldots, NP$ (i.e. possible solutions or individuals), where NP denotes the number of individuals in the DE population and G denotes the generation of the population (we have one population for each generation). Each individual is represented by a N-dimensional parameter vector (N is the number of optimization parameters) and its initial parameters are chosen randomly with uniform distribution in the attempt to generate solutions throughout all the search domain.

According to Storn and Price [Storn and Price 1995] there are three DE operators: mutation, crossover and selection. These operators are based on natural evolution principle in order to keep the population diversity, as well as to avoid premature convergence. The crucial idea behind the DE is a scheme for generating trial parameter vectors [Brest et al. 2006]. In this way, mutation and crossover operators are used to generate new vectors (trial vectors). Then, the selection operator determines which of these vectors will survive in the next generation. This procedure is repeated until a stop condition is reached.

2.2.1. Mutation Operator

Let be the vectors $\underline{x}_{r1,G}$, $\underline{x}_{r2,G}$ and $\underline{x}_{r3,G}$ with randomly chosen indexes $r1$, $r2$ and $r3$ $\in [1, NP]$. For each target vector $\underline{x}_{k,G}$, a mutant vector ($\underline{v}_{i,G+1}$) is generated according to [Brest et al. 2006]:

$$\underline{v}_{i,G+1} = \underline{x}_{r1,G} + F(\underline{x}_{r2,G} - \underline{x}_{r3,G}) \text{ with } i \neq r1 \neq r2 \neq r3, \tag{1}$$

where the term $F \in [0, 2]$ is a real-valued number which controls the amplification of the difference vector $(\underline{x}_{r2,G} - \underline{x}_{r3,G})$. Note that indexes have to be different from each other and from the running index so that there must be at least four. It is important to mention that due to this operator, the target vector values may exceed their valid boundary values. Whenever that happens, the corresponding values are truncated (projected) to their upper/lower bounds.

2.2.2. Crossover Operator

The crossover operator is introduced to increase the mutation individuals diversity. Thus, the target vectors are mixed with mutated vectors. This scheme generates the following vector [Brest et al. 2006]

$$\underline{u}_{i,G+1} = (u_{1i,G+1}, u_{2i,G+1}, \ldots, u_{Di,G+1}), \tag{2}$$

with

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1} & \text{if } r(j) \le CR \text{ or } j = rn(i) \\ x_{ji,G} & \text{if } r(j) < CR \text{ and } j \ne rn(i) \end{cases}, \tag{3}$$

where terms $j = 1, 2, \ldots, D$, $r(j) \in [0, 1]$ denotes the j-th evaluation of a uniform random generator number, $CR \in [0, 1]$ is the crossover probability and $rn(i) \in (1, 2, \ldots, D)$ is a randomly chosen index which ensures that $\underline{u}_{i,G+1}$ gets at least one element from $\underline{v}_{i,G+1}$. Otherwise, no new parent vector would be produced and the population would not be altered [Brest et al. 2006].

2.2.3. Selection Operator

The selection operator is responsible for the generation of the best sons (i.e. the vectors with the best evaluations). Thus, a greedy selection scheme is used, and is defined as [Brest et al. 2006]:

$$\underline{x}_{k,G+1} = \begin{cases} \underline{u}_{i,G+1} & \text{if } f(\underline{u}_{i,G+1}) < f(\underline{x}_{k,G}) \\ \underline{x}_{k,G} & \text{otherwise} \end{cases}. \tag{4}$$

If, and only if, the trial vector $\underline{u}_{i,G+1}$ yields a better cost function value than $\underline{x}_{k,G}$, then $\underline{x}_{k,G+1}$ is set to $\underline{u}_{i,G+1}$; otherwise, the old value $\underline{x}_{k,G}$ is retained [Brest et al. 2006].

3. The Proposed Improved Differential Evolution

In this section, we introduce the Improved Differential Evolution (IDE). It uses improved DE operators (mutation and crossover) in order to explore the state space more efficiently and to enhance convergence speed. The IDE procedure is shown in Figure 2 and its details will be presented in the next sections.

3.1. Initial Population Generation

The initial IDE population at generation G, defined by $\underline{x}_G$, is composed of randomly generated individuals, which are possible solutions to the problem. It is given by,

$$\underline{x}_G = (\underline{x}_{1,G}, \underline{x}_{2,G}, \ldots, \underline{x}_{S,G}), \tag{5}$$

with

$$\underline{x}_{S,G} = (x_{S1,G}, x_{S2,G}, \ldots, x_{SN,G}), \tag{6}$$

and

$$\underline{pmin} \le \underline{x}_{S,G} \le \underline{pmax}, \tag{7}$$

where $\underline{x}_G$ denotes the population at generation G; $\underline{x}_{S,G}$ denotes S-th individual of the population $\underline{x}_G$; $x_{SN,G}$ denotes the N-th parameter of the S-th individual of the population; S denotes the population size (number of individuals); N denotes the number of individual parameters (variables of the problem). The terms $\underline{pmin} = (pmin_1, pmin_2, \ldots, pmin_N)$ and $\underline{pmax} = (pmax_1, pmax_2, \ldots, pmax_N)$ denote, respectively, the minimum and maximum values (domain) of the parameters of any individual.

```
ImprovedDifferentialEvolutionProcedure() {
  $G = 0$;
  initialize $\underline{x}_G$;        // $\underline{x}_G$: population at generation G
  evaluate $f(\underline{x}_G)$;     // $f(\cdot)$: evaluate function or fitness
  while ( not termination condition ) {
    $\underline{x}_{G+1}$ = $\underline{x}_G$;
    select a random individual ($\underline{x}_{k,G}$) from population $\underline{x}_G$;
    perform the mutation operator {
      generate the mutated individuals $\underline{m}_1$, $\underline{m}_2$, $\underline{m}_3$, $\underline{m}_4$ and $\underline{m}_5$
        by Equations (9)-(13);
      the son with the best fitness function is denoted $\underline{v}_{i,G+1}$;
    }
    perform the crossover operator {
      generate the crossover individuals $\underline{c}_1$, $\underline{c}_2$, $\underline{c}_3$ and $\underline{c}_4$
        by Equations (14)-(17);
      the son with the best fitness function is denoted $\underline{u}_{i,G+1}$;
    }
    perform the selection operator (generating $\underline{x}_{k,G+1}$ by Equation (4));
    $G = G + 1$;
  }
}
```

Figure 2. Improved differential evolution procedure.

3.2. Evaluation Process

Each IDE individual should be evaluated by a defined fitness function (or cost function). In this way, better chromosomes in IDE population will have high fitness function values. Hence, a possible fitness function definition is given by

$$fitness\ function = \frac{1}{1 + |min - f(\underline{x}_{S,G})|}, \tag{8}$$

where $f(\cdot)$ is an heuristic function and min denotes the minimum function value. It is worth to mention that the fitness function is dependent of the application objective [Leung et al. 2003].

3.3. Proposed Mutation Operator

The proposed mutation generates five new individuals ($\underline{m}_j, j = 1, \ldots, 5$), which are defined by the following equations:

$$\underline{m}_1 = \underline{x}_{r1,G} + F(\underline{x}_{r2,G} - \underline{x}_{r3,G}), \tag{9}$$

$$\underline{m}_2 = \underline{x}_{best,G} + F(\underline{x}_{r1,G} - \underline{x}_{r2,G}), \tag{10}$$

$$\underline{m}_3 = \underline{x}_{r1,G} + F[(\underline{x}_{r2,G} - \underline{x}_{r3,G}) + (\underline{x}_{r4,G} - \underline{x}_{r5,G})], \tag{11}$$

$$\underline{m}_4 = \underline{x}_{best,G} + F[(\underline{x}_{r1,G} - \underline{x}_{r2,G}) + (\underline{x}_{r3,G} - \underline{x}_{r4,G})], \tag{12}$$

$$\underline{m}_5 = \underline{x}_{k,G} + \lambda(\underline{x}_{best,G} - \underline{x}_{k,G}) + F(\underline{x}_{r1,G} - \underline{x}_{r2,G}), \tag{13}$$

where $r1 \neq r2 \neq r3 \neq r4 \neq r5 \neq k$ are randomly chosen individuals indexes and $best$ denotes the best individual index of the current population. Term $F \in [0, 2]$ and $\lambda \in [0, 2]$ are a real-valued number which control the amplification of the difference vectors. These mutated individuals (Equations (9)-(13)) are used in order to explore the state space of possible solutions with greater effectiveness, thus preventing the algorithm going to the local minima of the fitness function surface.

After the generation of the sons through mutation (Equations (9)-(13)), the son with the best evaluation (greater fitness value) will be chosen as the son generated by the mutation process and will be denoted by $\underline{v}_{i,G+1}$.

3.4. Proposed Crossover Operator

The proposed crossover process to generate the vectors $\underline{c}_1$, $\underline{c}_2$, $\underline{c}_3$ and $\underline{c}_4$ is done with the use of four crossover operators, which are defined by the following equations (based on [Leung et al. 2003]):

$$\underline{c}_1 = \frac{\underline{v}_{i,G+1} + \underline{x}_{k,G}}{2}, \tag{14}$$

$$\underline{c}_2 = \underline{pmax}(1 - w) + max(\underline{v}_{i,G+1}, \underline{x}_{k,G})w, \tag{15}$$

$$\underline{c}_3 = \underline{pmin}(1 - w) + min(\underline{v}_{i,G+1}, \underline{x}_{k,G})w, \tag{16}$$

$$\underline{c}_4 = \frac{(\underline{pmax} + \underline{pmin})(1 - w) + (\underline{v}_{i,G+1} + \underline{x}_{k,G})w}{2}, \tag{17}$$

where $w \in [0, 1]$ denotes the crossover weight (the closer w is to 1, the greater is the direct contribution from parents), $max(\underline{v}_{i,G+1}, \underline{x}_{k,G})$ and $min(\underline{v}_{i,G+1}, \underline{x}_{k,G})$ denote the vector whose elements are the maximum and the minimum, respectively, between the gene values of $\underline{v}_{i,G+1}$ and $\underline{x}_{k,G}$. The terms $\underline{pmax}$ and $\underline{pmin}$ denote the maximum and minimum possible gene values, respectively. In this way, the use of these crossover operators, produces better offspring over the domain, since $\underline{c}_1$ and $\underline{c}_4$ results in searching around the center region of the domain, and $\underline{c}_2$ and $\underline{c}_3$ move the potential offspring to be near the maximum and minimum, respectively, domain boundary [Leung et al. 2003].

After the generation of the sons through crossover (Equations (14)-(17)), the son with the best evaluation (greater fitness value) will be chosen as the son generated by the crossover process and will be denoted by $\underline{u}_{i,G+1}$.

3.5. Selection Operator

The selection operator is the same of the Differential Evolution (Section 2.2.3). However, it is worth to mention that only one individual ($\underline{x}_{k,G+1}$) is modified (this factor depends on the evaluation of the individual $\underline{u}_{i,G+1}$), while in the standard DE all the individuals are modified if its fitness is worse than the individuals generated by the crossover operator.

4. Simulations and Experimental Results

A set of six well-known benchmark problems of single objective optimization are used to assess performance of the proposed algorithm. The SDE parameters used are: a maximum number of generations corresponding to 10^3, population size equals to 100, $F = 0.5$ and $CR = 0.9$. These values are based on values proposed in the literature [Storn and Price 1995, Storn 1999, Ali and Torn 2004, Vesterstroem and Thomsen 2004, Liu and Lampinen 2005, Brest et al. 2006]. The proposed IDE parameters used are: a maximum number of generations corresponding to 10^3, population size equals to 10, $F = 0.5$, $\lambda = 0.95$ and $w = 0.9$. Also, these values are based on values proposed by [Storn and Price 1995,Storn 1999,Leung et al. 2003]. The feasible solution space depends on the benchmark function and will be presented later for each used function.

For each benchmark function, a hundred experiments were done, where it was used, for comparison effect, the mean of convergence results in order to prove graphically the fastest convergence of the proposed IDE when compared to SDE. The obtained results suggest that the proposed IDE performance is better (in terms of the fitness function and the convergence speed) than that of the SDE.

4.1. Benchmark Functions

$$f_1(\underline{x}) = \sum_{i=1}^{N} x_i^2, \ -100 \leq x_i \leq 100, \tag{18}$$

where $\underline{x} = (x_1, x_2, \ldots, x_N)$ and N represents the dimension of vector $\underline{x}$.

$$\begin{aligned} f_2(\underline{x}) = -20exp&\left(-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N}x_i^2}\right) \\ &- exp\left[\frac{1}{N}\sum_{i=1}^{N}cos(2\pi x_i)\right] + 20 + exp(1), \ -32 \leq x_i \leq 32; \end{aligned} \tag{19}$$

$$f_3(\underline{x}) = \sum_{i=1}^{N}[x_i^2 - 10cos(2\pi x_i) + 10], \ -5.12 \leq x_i \leq 5.12; \tag{20}$$

$$f_4(\underline{x}) = \sum_{i=1}^{N}(\lfloor x_i + 0.5 \rfloor)^2, \ -100 \leq x_i \leq 100; \tag{21}$$

$$f_5(\underline{x}) = \frac{1}{4000}\sum_{i=1}^{N}x_i^2 - \prod_{i=1}^{N}cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, \ -600 \leq x_i \leq 600; \tag{22}$$

$$\begin{aligned} f_6(\underline{x}) = \frac{\pi}{N}&\left\{10sin^2(\pi y_i) + \sum_{i=1}^{N-1}(y_i - 1)^2[1 + 10sin^2(\pi y_{i+1})] + (y_N - 1)^2\right\} \\ &+ \sum_{i=1}^{N}u(x_i, 10, 100, 4), \\ &y_i = 1 + \frac{1}{4}(x_i + 1), \\ u(x_i, a, k, m) &= \left\{ \begin{array}{ll} k(x_i - a)^m & x_i > a \\ 0 & -a \leq x_i \leq a \\ k(-x_i - a)^m & x_i < a \end{array} \right. , \ -50 \leq x_i \leq 50. \end{aligned} \tag{23}$$

Table 1. Results for all benchmark functions.

	Best Fitness		Averaged Fitness (Standard Deviation)		Expected Fitness
	IDE	SDE	IDE	SDE	
$f_1(\underline{x})$	5.20e-9	2.32e-8	9.11e-9 (1.17e-9)	1.61e-7 (2.08e-7)	0
$f_2(\underline{x})$	6.54e-9	1.06e-4	1.16e-8 (3.52e-9)	3.86e-4 (2.56e-4)	0
$f_3(\underline{x})$	0	59.44	0 (0)	80.84 (9.12)	0
$f_4(\underline{x})$	0	0	0 (0)	0.22 (0.73)	0
$f_5(\underline{x})$	0	2.32e-8	0 (0)	1.54e-7 (5.88e-3)	0
$f_6(\underline{x})$	9.08e-12	2.08e-7	3.32e-6 (9.81e-6)	0.06 (0.22)	0

All benchmark functions (f_1, f_2, f_3, f_4, f_5 and f_6) [Brest et al. 2006] use $N = 30$. Table 1 shows the obtained results for all benchmark functions for the IDE and SDE. Figure 3 shows the averaged convergence results for all the benchmark functions.

According to these accomplished computational experiments, it was observed that the proposed IDE can find the optimal or close-to-optimal solutions having small standard deviations (it means that the proposed IDE have a stable solution quality). Note that the proposed IDE found the optimal solution in three out of the six tested benchmark functions, while the SDE found the optimal solution just in one of the benchmark functions. Furthermore, it was observed that the proposed IDE quickly converges, in all the tested benchmark functions, for the global minimum. Also, it is worth to mention that, according to the presented convergence results in Figure 3, it can be seen that the proposed IDE had fastest convergence speed than the SDE.

Through the comparisons between the IDE and SDE algorithms (in terms of fitness function and convergence speed), it was observed that the proposed mutation and crossover operators can improve the SDE performance, obtaining more robust solutions and better convergence stability. Also, it can be seen in the convergence figures for most of the tested functions that, the number of generations necessary for the IDE convergence was no more than two hundred epochs. Additionally, it is observed that with the use of only ten individuals in the population, the IDE algorithm performed much better (fitness) and converged much faster (number of generations) than the SDE (using a hundred individuals in its population), clearly showing the superior performance of the proposed IDE.

5. Conclusion

In this paper, the Improved Differential Evolution (IDE) has been presented to solve single objective optimization problems with continuous variables. The proposed IDE uses improved DE operators (mutation and crossover) in order to explore with larger effectiveness the search space, as well as to quickly find optimal or close-to-optimal solutions, accelerating its convergence speed and allowing to the IDE to escape with larger effectiveness from local minima in state space, improving the DE drawbacks.

Furthermore, six well-known benchmark problems of single objective optimization were used to assess performance of the proposed algorithm. All benchmark problems had thirty dimensions. Since these benchmark functions has a very large number of local minima, finding satisfactory solutions becomes a great challenge. Therefore, according to the presented results, the proposed IDE converges quickly for the global minimum (had fastest convergence speed than SDE, as illustrated in the convergence figures), as well as

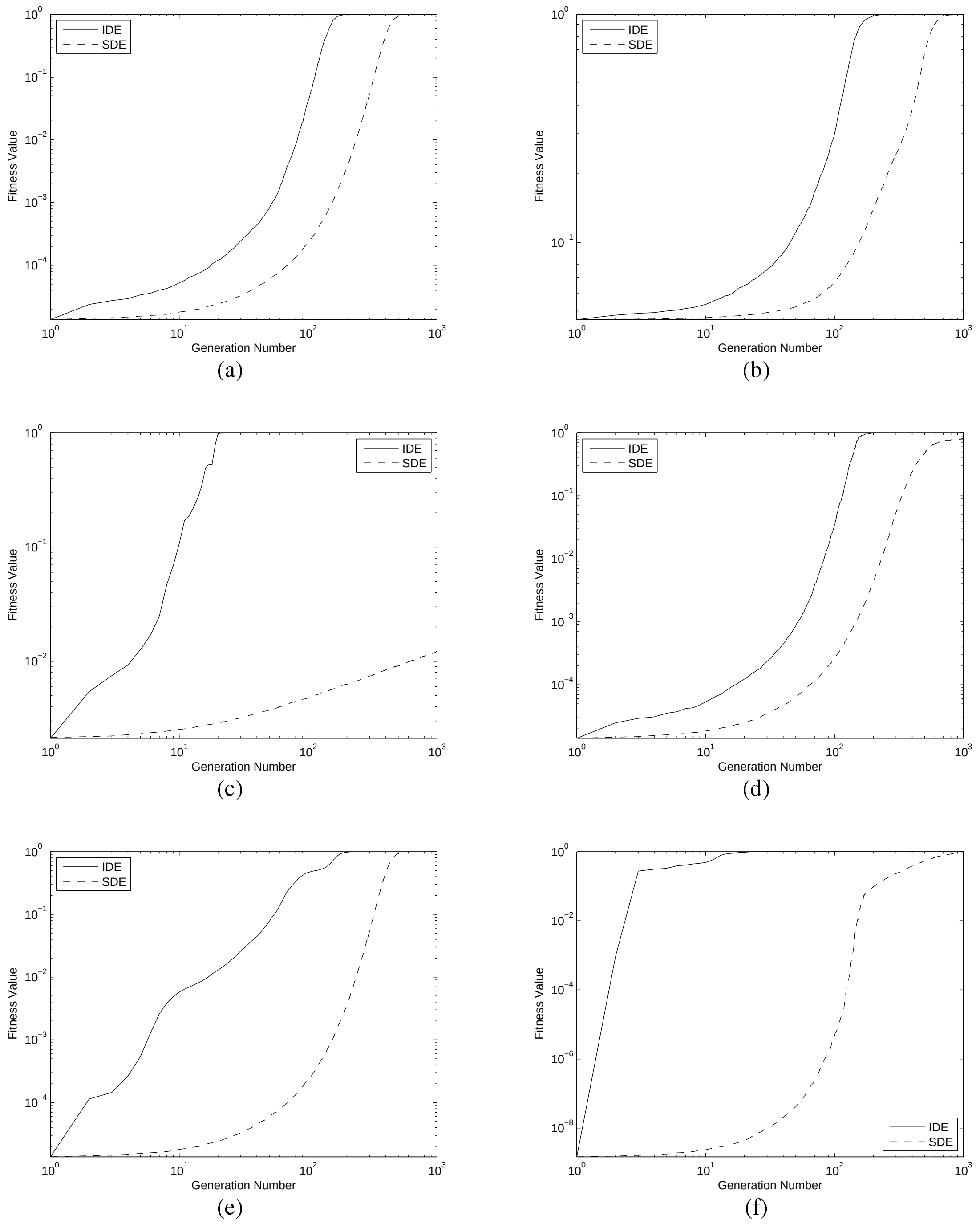

Figure 3. Convergence results of all benchmark functions: (a) $f_1(\underline{x})$**, (b)** $f_2(\underline{x})$**, (c)** $f_3(\underline{x})$**, (d)** $f_4(\underline{x})$**, (e)** $f_5(\underline{x})$ **and (f)** $f_6(\underline{x})$**.**

obtained better solutions than the SDE.

The accomplished computational experiments show that the proposed IDE can find optimal or close-to-optimal solutions, and it is more more efficient than the SDE on the tested benchmark functions. Future works will consider the development of the multi-objective IDE.

References

Ali, M. M. and Torn, A. (2004). Population set-based global optimization algorithms: some modifications and numerical studies. *Comput. Oper. Res.*, 31(10):1703–1725.

Angelov, P. (2001). Supplementary crossover operator for genetic algorithms based on the centre-of-gravity paradigm. *Control and Cybernetics*, 30(2):159–176.

Bersini, H. and Renders, J.-M. (1994). Hybridizing genetic algorithms with hill-climbing methods for global optimization: Two possible ways. In *International Conference on Evolutionary Computation*, pages 312–317.

Brest, J., Greiner, S., Bošković, B., Mernik, M., and Žumer, V. (2006). Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. *IEEE Transactions on Evolutionary Computation*, 10(6):646–657. DOI: 10.1109/TEVC.2006.872133.

Chellapilla, K. (1998). Combining mutation operators in evolutionary programming. *IEEE Trans. on Evolutionary Computation*, 2(3):91–96.

Darwin, C. (1859). *The origin of species*. United King.

Franklin, S. (2001). *Articicial Minds*. MIT Press.

Gen, M. and Cheng, R. (1997). *Genetic Algorithms and Engineering Design*. John Wiley and sons, New York.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Michigan.

Leung, F. H. F., Lam, H. K., Ling, S. H., and Tam, P. K. S. (2003). Tuning of the structure and parameters of the neural network using an improved genetic algorithm. *IEEE Transactions on Neural Networks*, 14(1):79–88.

Leung, Y.-W. and Wang, Y. (2001). An orthogonal genetic algorithm with quantization for global numerical optimization. *IEEE Trans. Evolutionary Computation*, 5(1):41–53.

Liu, J. and Lampinen, J. (2005). A fuzzy adaptive differential evolution algorithm. *Soft Comput.*, 9(6):448–462.

Storn, R. (1999). System design by constraint adaptation and differential evolution. *IEEE Trans. on Evolutionary Computation*, 3(1):22–34.

Storn, R. and Price, K. (1995). Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, Berkeley, CA.

Sun, J., Zhang, Q., and Tsang, E. P. K. (2005). De/eda: a new evolutionary algorithm for global optimization. *Inf. Sci. Inf. Comput. Sci.*, 169(3-4):249–262.

Tsai, J.-T., Liu, T.-K., and Chou, J.-H. (2004). Hybrid taguchi-genetic algorithm for global numerical optimization. *IEEE Trans. Evolutionary Computation*, 8(4):365–377.

Vesterstroem, J. and Thomsen, R. (2004). A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 2, pages 1980–1987.

Yao, X. and Liu, Y. (1996). Fast evolutionary programming. In *Evolutionary Programming*, pages 451–460.