ODE: A Fast And Robust Differential Evolution Based on Orthogonal Design

Wenyin Gong[1], Zhihua Cai[1,2] and Charles Ling[2]

[1] School of Computer Science
China University of Geosciences, Wuhan 430074, P. R. China
cug11100304@yahoo.com.cn
[2] Dept. of computer science
The University of Western Ontario London,Ontario,N6A 5B7,Canada

Abstract. In searching for optimal solutions, Differential Evolution (DE), a type of genetic algorithms, has been shown powerful. However, DE has been shown to have certain weaknesses, such as slow convergence. In this paper, we propose an improved differential evolution based on orthogonal design, and we call it ODE (Orthogonal Differential Evolution). ODE makes DE faster and more robust. It uses a novel and robust crossover based on orthogonal design and generates an optimal offspring by a statistical optimal method. A new selection strategy is applied to decrease the number of generations and make the algorithm converge faster. We evaluate ODE to solve twelve benchmark function optimization problems with a large number of local minima. Simulations indicate that ODE is able to find the near-optimal solutions in all cases. Compared to other state-of-the-art evolutionary algorithms, ODE performs significantly better in terms of the quality, speed, and stability of the final solutions.

1 Introduction

In the last two decades, evolutionary algorithms (including genetic algorithms, evolution strategies, evolutionary programming, and genetic programming) have received much attention regarding their potential as global optimization techniques [1]. Inspired from the mechanisms of natural evolution and survival of the fittest, evolutionary algorithms (EAs) utilize a collective learning process of a population of individuals. Descendants of individuals are generated using randomized operations such as mutation and recombination. Mutation corresponds to an erroneous self-replication of individuals, while recombination exchanges information between two or more existing individuals. According to a fitness measure, the selection process favors better individuals to reproduce more often than those that are relatively worse.

Differential evolution (DE) [2] is an improved version of Genetic Algorithm (GA) for faster optimization, which has won the third place at the 1st International Contest on Evolutionary Computation on a real-valued function test-suite. DE is the best genetic algorithm approach. Unlike simple GA that uses binary coding for representing problem parameters, DE is a simple yet powerful population based, direct search algorithm with the generation-and-test feature for globally optimizing functions using real valued parameters. The DE's advantages are its simple structure, ease of use, speed and

robustness. Price & Storn [2] gave the working principle of DE with single strategy. Later on, they suggested ten different strategies of DE [10]. A strategy that works out to be the best for a given problem may not work well when applied for a different problem. What's more, the strategy and key parameters to be adopted for a problem are to be determined by trial & error. The main difficulty with the DE technique, however, appears to lie in the slowing down of convergence as the region of global minimum is approaching.

Orthogonal design method [9] with both orthogonal array (OA) and factor analysis (such as the statistical optimal method) is developed to sample a small, but representative set of combinations for experimentation to obtain good combinations. OA is a fractional factorial array of numbers arranged in rows and columns, where each row represents the levels of factors in each combination, and each column represents a specific factor that can be changed from each combination. It can assure a balanced comparison of levels of any factor. The term "main effect" designates the effect on response variables that one can trace to a design parameter. The array is called orthogonal because all columns can be evaluated independently of one another, and the main effect of one factor does not bother the estimation of the main effect of another factor.

Recently, some researchers applied the orthogonal design method incorporated with EAs to solve optimization problems. Leung and Wang [5] incorporated orthogonal design in genetic algorithm for numerical optimization problems found such method was more robust and statistically sound. This method was also adopted by other researchers [6], [7] and [8] to solve optimization problems. Numerical results demonstrated that these techniques had a significantly better performance than the traditional EAs on the problems studied, and the resulting algorithm can be more robust and statistically sound.

In this paper, an improved version of the differential evolution based on the orthogonal design (ODE) is presented to make the DE faster and more robust. Orthogonal design method is nested in crossover operator with the statistical optimal method to select better genes as offspring, and consequently, enhances the performance of the CDE. Both orthogonal crossover and DE/rand/1/exp are used in the ODE. By combining with two crossover operators, faster convergence and better solutions are obtained. Moreover, the ODE can remedy the main defect of the DE technique mentioned above with the orthogonal crossover operator. The advantages of ODE are its simplicity, efficiency, and flexibility. It is shown empirically that ODE has high performance in solving benchmark functions comprising many parameters, as compared with some existing EAs.

The rest of this paper is organized as follows. Section 2 briefly describes function optimization problem and some properties of the orthogonal design method. Section 3 presents the proposed ODE. In Section 4, we test our algorithm through twelve benchmark functions to the test proposed algorithm. This is followed by discussions and analysis of the optimization experiments for the ODE in Section 5. The last section, Section 6, is devoted to conclusions and future studies.

2 Preliminary

2.1 Problem Definition

A global minimization problem can be formalized as a pair (S, f) , where $S \subseteq R^n$ is a bounded set on R^n and $f : S \rightarrow R$ is an n-dimensional real-valued function. The problem is to find a point $X_{min} \in S$ such that $f(X_{min})$ is a global minimum on S. More specifically, it is required to find an $X_{min} \in S$ such that

$$\forall X \in S : f(X_{min}) \leq f(X) \tag{1}$$

where f does not need to be continuous but it must be bounded

$$l_i \leq x_i \leq u_i, i = 1, 2, \cdots, n \tag{2}$$

2.2 Orthogonal Design

In a discrete single objective optimization problem, when there are N factors (variables) and each factor has Q levels, the search space consists of Q^N combinations of levels. When N and Q are large, it may not be possible to do all Q^N experiments to obtain optimal solutions. Therefore, it is desirable to sample a small, but representative set of combinations for experimentation, and based on the sample, the optimal may be estimated. The orthogonal design was developed for the purpose [9]. The selected combinations are scattered uniformly over the space of all possible combinations Q^N. And the orthogonal design is an important tool for robust design. Robust design is an engineering methodology for optimizing the product and process conditions which are minimally sensitive to the causes of variation, and which produce high-quality products with low development and manufacturing costs.

In general, the orthogonal array $L_M(Q^N)$ has the following properties.

1. For the factor in any column, every level occurs equal times.
2. For the two factors in any two columns, every combination of two levels occurs equal times to represent the experiments.
3. Any two experiments are not the same, so their results cannot be compared directly.
4. If any two columns of an orthogonal array are swapped, the resulting array is still an orthogonal array.
5. If some columns are taken away from an orthogonal array, the resulting array is still an orthogonal array with a smaller number of factors.

3 ODE: Orthogonal DE

In order to get rapid convergence as the region of global minimum, make it easy to manipulate and get better solutions, we modify the CDE. The enhancements of the ODE are as follows:

3.1 Orthogonal Crossover Operator

3.1.1 Design of the orthogonal array To design a minimal orthogonal array, in this research, we use the two level orthogonal array $L_{2^J}(2^{2^J-1})$, $R = 2^J$ denotes the number of the rows of orthogonal array and $C = 2^J - 1$ denotes the number of the columns. The orthogonal array needs to find a proper J to satisfy

$$\begin{array}{l} \text{Minimize: } R = 2^J \\ \text{s.t.: } C = 2^J - 1 \geq N \end{array} \tag{3}$$

where N is the number of the variables. In this study, we adopt the algorithm described in Ref. [5] to construct an orthogonal array. In particular, we use $L(R, C)$ to indicate the orthogonal array; and $a(i, j)$ denotes the level of the jth factor in the ith combination in $L(R, C)$.

3.1.2 Generation of the orthogonal sub-population After constructing a proper orthogonal array, we select two parents randomly, $X_1 = (x_{11}, x_{12}, \cdots, x_{1N})$ and $X_2 = (x_{21}, x_{22}, \cdots, x_{2N})$, to generate the orthogonal sub-population $O(R, N)$ for two level orthogonal crossover as follows

Algorithm 1: Generation of the orthogonal sub-population $O(R, N)$

for $i = 1$ to R
  for $j = 1$ to N
    $k = a(i, j)$
    if $k == 1$ then
      $O(i, j) = x_{1j}$
    End if
    if $k == 2$ then
      $O(i, j) = x_{2j}$
    End if
  End for
End for

Note: Because the number of the columns of the orthogonal array $C \geq N$, if $C > N$ we delete the last $C - N$ columns to get an orthogonal array with N factors in algorithm 1. The remainder of the orthogonal array is still an orthogonal array because of the fifth property of the orthogonal design.

3.1.3 Statistical optimal method From the third property of the orthogonal design we know that any two experimental results cannot be compared directly. We adopt the statistical optimal method [9] in order to generate a better offspring, which is similar to the method adopted in Ref. [8], from the orthogonal sub-population $O(R, N)$. Calculation of the value of the kth level of the jth factor $E(j, k)$ is

$$E(j, k) = \sum_{a(i,j)=k} X_i.f \tag{4}$$

where $X_i.f$ is the fitness of the individual X_i. For each factor, we select the level with the minimal $E(j, k)$ as the component of the offspring X'.

Algorithm 2: Generation of the offspring with statistical optimal method

```
for j = 1 to N
    for i = 1 to R
        k = a(i, j)
        if k == 1 then
            E(j, 1)+ = X_i.f
        End if
        if k == 2 then
            E(j, 2)+ = X_i.f
        End if
    End for
End for
for j = 1 to N
    if E(j, 1) < E(j, 2) then
        x'_j = x_1j
    End if
    if E(j, 1) ≥ E(j, 2) then
        x'_j = x_2j
    End if
End for
```

3.2 Decision variable fraction strategy

If the dimension of the variable is higher, it needs to design a larger orthogonal array to satisfy $C \geq N$. For example, if $N = 100$, the smallest J is 7, and $R = 128$. When we use the two level orthogonal crossover, it needs to evaluate the fitness function 128 times to generate an offspring. Therefore, each pair of parents should produce too many potential offspring. In order to avoid a large number of function evaluations during selection, we use the decision variable fraction strategy to divide the variables into groups, which is a small design parameter, and each group with the same number of components is treated as one factor. Consequently, the corresponding orthogonal array has a small number of combinations, and hence a small number of potential offspring are generated.

3.3 Simplifying the Scaling Factor

The scaling factor F is generated uniform randomly from $F \in (0, 1]$ in the proposed ODE to make it simple and easy to use.

3.4 Handling the Constraint of the Variables

When we adopt the DE/rand/1/exp strategy to generate a point X, if some dimension values of the point beyond the constraint of the variables, $x_i \notin [l_i, u_i]$, we use the following rules to adjust it:

$$x_i = \begin{cases} l_i + U_i(0,1) \times (u_i - l_i) & \text{if } x_i < l_i \\ u_i - U_i(0,1) \times (u_i - l_i) & \text{if } x_i > u_i \end{cases} \tag{5}$$

where $U_i(0, 1)$ is the uniform random variable from [0,1] in each dimension i.

3.5 Orthogonal DE

The evolutionary process of the ODE is similar to GAs. It is described as follows.

Algorithm 3: Algorithm of the Orthogonal DE

- Generate the initial population randomly and calculate the fitness of each individual;
- Find the best and the worst individuals in the current population;
- Construct a proper orthogonal array;
- While the halting criterion is not satisfied Do
  - For each individual in the population, execute DE/rand/1/exp strategy to generate a new population;
  - Find the best and the worst individuals in the new population;
  - Select two different individuals, X_1 and X_2, randomly from the new population;
  - Execute algorithm 1 and algorithm 2 to generate a child X';
  - if $f(X') < f(X_{worst})$ then
    $X_{worst} = X'$;
  - End if
  - Report the results;
- Do

Note: In algorithm 3, the halting criterion is that the maximal number of fitness function evaluations is reached or $|f(X_{best}) - f(X_{worst})| < \varepsilon$.

4 Simulations

4.1 Experimental Setup

For all experiments in ODE, we used the following parameters:

Population size: *M* = 100;

Number of fitness function evaluations: *NFFE* = 100,000;

Probability of crossover: *CR* = 0.9;

Number of decision variable fractions: if $N > 15$, then $F = 2$, else $F = N$

Halting precision: $\varepsilon = 1 \times 10^{-30}$. This halting criterion used is to make the algorithm stop earlier when the results satisfy the precision of the problems.

To assess the efficiency of the proposed ODE approach, we also test the functions in CDE, which does not use the orthogonal design, with DE/rand/1/exp strategy. And for all experiments in CDE, we used the parameters mentioned above only expect for the number of fitness function evaluations, which is different for different problems.

4.2 Benchmark Functions

In order to test the performance of our method twelve benchmark functions ($f_{01} - f_{12}$) were used. All of the functions are **minimization** problems. Where functions $f_{01} - f_{07}$ are from $f_{01} - f_{07}$ in Ref. [5]; functions $f_{08} - f_{10}$ are from f_{09}, f_{12}, f_{13} in Ref. [5]; and functions f_{11}, f_{12} are from f_{22}, f_{23} in Ref. [4]. For functions $f_{01} - f_{06}$ and f_{09}, f_{10}, the

dimension of variables is 30; for functions f_{07}, f_{08}, the dimension of variables is 100; and the dimension of variables for functions f_{11}, f_{12} is 4. Functions $f_{01} - f_{08}$ are high-dimensional and multimodal problems with many local minima. Functions f_{09}, f_{10} are high-dimensional and unimodal problems. Also function f_{09} is a noisy quartic function, where *random* [0,1) is a uniformly distributed random variable in [0,1). Functions f_{11}, f_{12} are low-dimensional and multimodal problems with few local minima; and they are difficult to find the global minima on each run for many algorithms [4]. The test functions are described as follows.

$$f_{01} = \sum_{i=1}^{N} (-x_i \sin(\sqrt{|x_i|}))$$

$$f_{02} = \sum_{i=1}^{N} (x_i^2 - 10\cos(2\pi x_i) + 10)$$

$$f_{03} = -20\exp(-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}) - \exp(\frac{1}{N}\sum_{i=1}^{N}\cos(2\pi x_i)) + 20 + \exp(1)$$

$$f_{04} = \frac{1}{4000}\sum_{i=1}^{N} x_i^2 - \prod_{i=1}^{N}\cos(\frac{x_i}{\sqrt{i}}) + 1$$

$$f_{05} = \frac{\pi}{N}\{10\sin^2(\pi y_i) + \sum_{i=1}^{N-1}(y_i - 1)^2 \cdot [1 + 10\sin^2(\pi y_{i+1})] + (y_N - 1)^2\}$$
$$+ \sum_{i=1}^{N} u(x_i, 10, 100, 4)$$

where $y_i = 1 + \frac{1}{4}(x_i + 1)$
and

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

$$f_{06} = \frac{1}{10}\{\sin^2(3\pi x_1) + \sum_{i=1}^{N-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_N - 1)^2[1 + \sin^2(2\pi x_N)]\}$$
$$+ \sum_{i=1}^{N} u(x_i, 5, 100, 4)$$

$$f_{07} = -\sum_{i=1}^{N}\sin(x_i)\sin^{20}(\frac{i \times x_i^2}{\pi})$$

$$f_{08} = \frac{1}{N}\sum_{i=1}^{N}(x_i^4 - 16x_t^2 + 5x_i)$$

$$f_{09} = \sum_{i=1}^{N} x_i^4 + random[0, 1)$$

$$f_{10} = \sum_{i=1}^{N}|x_i| + \prod_{i=1}^{N}|x_i|$$

$$f_{11,12} = -\sum_{i=1}^{m} [(x - a_i)(x - a_i)^T + c_i]^{-1}$$

where for function f_{11}, $m = 7$; for function f_{12}, $m = 10$. The coefficients are not described here.

4.3 Experimental Results

Table 1 shows the results compared with OGA/Q [5] and CDE. Table 2 shows the results compared with FEP [4] and CDE. Where all results have been averaged over 50 independent trails on each test function in standard C++ and recorded: 1) the mean number of fitness function evaluations (MFFE), 2) the mean function value (Mean Best), and 3) the standard deviation of the functions (Std. Dev.). Some of the convergence results of the test functions compared with ODE and CDE are shown in Fig. 1.

Table 1. Comparison with ODE, CDE and OGA/Q on functions $f_{01} - f_{10}$ over 50 independent runs. A result in **Boldface** indicates that a better result or the global optimum (or best known solution) was reached. Where "Optimal" in column 11 indicates the global optimum (or best known solution), similarly hereinafter.

F	MFFE			Mean Best			Std. Dev.			Optimal
	ODE	CDE	OGA/Q	ODE	CDE	OGA/Q	ODE	CDE	OGA/Q	
f_{01}	**47,980**	72,100	302,166	**-12569.5**	**-12569.5**	-12569.4537	**0**	**0**	$6.4*10^{-4}$	-12569.5
f_{02}	**36,430**	159,600	224,710	**0**	**0**	**0**	**0**	**0**	**0**	0
f_{03}	**57,850**	206,300	112,421	$5.9*10^{-15}$	$4.9*10^{-15}$	**4.4**$*10^{-16}$	**0**	$1.4*10^{-15}$	$4.0*10^{-17}$	0
f_{04}	**51,970**	119,600	134,000	**0**	**0**	**0**	**0**	**0**	**0**	0
f_{05}	**100,000**	**100,000**	134,556	**1.4**$*10^{-16}$	$5.2*10^{-15}$	$6.0*10^{-6}$	**7.3**$*10^{-17}$	$1.6*10^{-15}$	$1.2*10^{-6}$	0
f_{06}	**99,850**	100,000	134,143	**4.1**$*10^{-19}$	$6.6*10^{-13}$	$1.9*10^{-4}$	**2.8**$*10^{-19}$	$3.5*10^{-13}$	$2.6*10^{-5}$	0
f_{07}	**98,570**	100,000	302,773	**-97.211**	-59.30467	-92.83	0.3	0.830304	**2.6**$*10^{-2}$	-99.2784
f_{08}	**98,570**	100,000	245,930	**-78.33233**	-78.3254	-78.30	**3.0**$*10^{-6}$	0.001458	$6.3*10^{-3}$	-78.3324
f_{09}	**100,000**	**100,000**	112,652	**6.6**$*10^{-4}$	0.01647	$6.3*10^{-3}$	**3.8**$*10^{-4}$	0.004207	$4.1*10^{-4}$	0
f_{10}	**100,000**	**100,000**	112,612	**0**	$1.7*10^{-8}$	**0**	**0**	$3.1*10^{-9}$	**0**	0

Table 2. Comparison with ODE, CDE and FEP on functions f_{11}, f_{12} over 50 independent runs. A result in **Boldface** indicates that a better result or the global optimum (or best known solution) was reached.

F	MFFE			Mean Best			Std. Dev.			Optimal
	ODE	CDE	FEP	ODE	CDE	FEP	ODE	CDE	FEP	
f_{11}	**6,640**	7,100	10,000	**-10.4029**	-10.40285	-5.52	**0**	$1.246*10^{-7}$	2.12	-10.4029
f_{12}	**7,030**	7,900	10,000	**-10.5364**	-10.53638	-6.57	**0**	$6.576*10^{-9}$	3.14	-10.5364

5 Experimental Discussions and Analysis

From table 1, table 2 and Fig. 1, we can get following conclusions.

– The proposed ODE can find optimal or close-to-optimal solutions and gave the smallest MFFE for all functions compared with the OGA/Q, CDE and FEP.

- For functions $f_{01} - f_{04}$ and $f_{10} - f_{12}$, the proposed ODE can find the global optimal solutions on each run. Hence, the "Std. Dev." of these functions are zero.
- For six functions (f_{01} and $f_{05} - f_{09}$), the ODE can obtain a better "Mean Best" solutions than the OGA/Q.
- Both the ODE and OGA/Q can give the same optimal or close-to-optimal solutions in three functions (f_{02}, f_{04} and f_{10}).
- Only except for the function f_{07}, the ODE gave a worse "Std. Dev." than the OGA/Q. However, it can give smaller standard deviations of function values in six functions ($f_{01}, f_{03}, f_{05}, f_{06}, f_{08}$ and f_{09}) than the OGA/Q and in the remaining three functions (f_{02}, f_{04} and f_{10}), both of them are 0. Hence, the proposed ODE has a more stable solution quality.
- The ODE found similar standard deviations of function values as the CDE for three functions (f_{01}, f_{02} and f_{04}), both of them are 0. And for the remaining nine functions, ODE was able to obtain smaller standard deviations of function values than the CDE. Therefore, ODE has a more stable solution quality than the CDE.
- The ODE found a better or a similar "Mean Best" solution in eleven functions compared with the CDE, only expect for the function f_{03}, the CDE is slightly better than ODE.
- The proposed ODE requires fewer mean numbers of function evaluations than the OGA/Q, CDE and FEP and, hence, the proposed ODE has a lower computational time requirement.
- From Fig. 1, we can see that the ODE can get faster convergence than that of the CDE. Especially, for some functions (such as f_{05}, f_{09} and f_{10}), the proposed ODE can find the close-to-optimal solutions at the beginning of the iteration. The reason is that the two level orthogonal crossover operator with the statistical optimal method can exploit the optimum offspring.
- Compared with the FEP for functions f_{11} and f_{12}, the performance of the ODE is significantly better than the FEP in terms of the MFFE, Mean Best and Std. Dev.

These results in table 1, table 2 and Fig. 1 indicate that the proposed ODE can, in general, give better mean solution quality, faster convergence speed and more stable solution quality than the OGA/Q, CDE and FEP. Thereore, ODE can be more robust and statistically sounder.

6 Conclusion and Future Work

A fast and robust DE (ODE) based on the orthogonal design is proposed in this paper; and then it is used to solve the global numerical optimization problems with continuous variables. The ODE combines the conventional DE (CDE), which is simple and efficient, with the orthogonal design, which can exploit the optimum offspring. Orthogonal design method is nested in crossover operator to select better genes as offspring, and consequently, enhances the performance of the CDE. Both orthogonal crossover and DE/rand/1/exp are used in the ODE. By combining two crossover operators, faster convergence and better solutions are obtained. The statistical optimal method is incorporated in the two level orthogonal crossover operation to select the better genes to achieve crossover, and consequently enhance the genetic algorithm. Moreover, decision variable

fraction strategy is applied to decrease the number of the orthogonal design combinations and make the algorithm converge faster. Therefore, the proposed ODE possesses the merits of global exploration, fast convergence, robustness, and statistical soundness. We executed the proposed algorithm to solve twelve benchmark problems with high or low dimensions, where some of these problems have numerous local minima. The computational experiments show that the proposed ODE can find optimal or close-to-optimal solutions, and it is more robust than the compared techniques (OGA/Q, CDE and FEP). It is, therefore, concluded that the proposed ODE is a promising technique in performing global optimization for practical applications. Furthermore, the ODE can have faster convergence speed than the CDE. Our future work consists on adding to a diversity mechanism to handle the constraints to solve the global constrained optimization problems.

Acknowledgment

The authors would like to thank Professor Y. W. Leung for his constructive advices. This work is supported by the Humanities Base Project of Hubei Province under Grant No. 2004B0011 and the Natural Science Foundation of Hubei Province under Grant No. 2003ABA043.

References

1. T. Bäck and H.-P. Schwefel: An overview of evolutionary algorithms for parameter optimization. Evolutionary Computation. **1** (1993) 1–23
2. R. Storn and K. Price: Differential evolution–A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization. **11** (1997) 341–359
3. Guo Tao and Kang Li-shan: A new Evolutionary Algorithm for Function Optimization. Wuhan University Journal of Nature Sciences. **4** (1999) 409–414
4. Xin Yao, Yong Liu, and Guangming Lin: Evolutionary Programming Made Faster. IEEE Transactions on Evolutionary Computation. **3** (1999) 82–102
5. Y. W. Leung and Y. Wang: An Orthogonal Genetic Algorithm with Quantization for Global Numerical Optimization. IEEE Transactions on Evolutionary Computation, **5** (2001) 41–53
6. Ding C. M, Zhang C. S. and Liu G. Z: Orthogonal Experimental Genetic Algorithms and Its Application in Function Optimization (in Chinese). Systems Engineering and Electronics, **10** (1997) 57–60
7. SHI Kui-fan, DONG Ji-wen, LI Jin-ping, et al: Orthogonal Genetic Algorithm (in Chinese). ACTA ELECTRONICA SINICA, **10** (2002) 1501–1504
8. ZENG San-You, WEI Wei, KANG Li-Shan, et al: A Multi-Objective Evolutionary Algorithm Based on Orthogonal Design (in Chinese). Chinese Journal of Computers, **28** (2005) 1153–1162
9. K. T. Fang and C. X. Ma: Orthogonal and Uniform Design (in Chinese). Science Press, (2001)
10. K. Price and R. Storn: Home Page of Differential Evolution. Available: http://www.ICSI.Berkeley.edu/~storn/code.html. (2003)

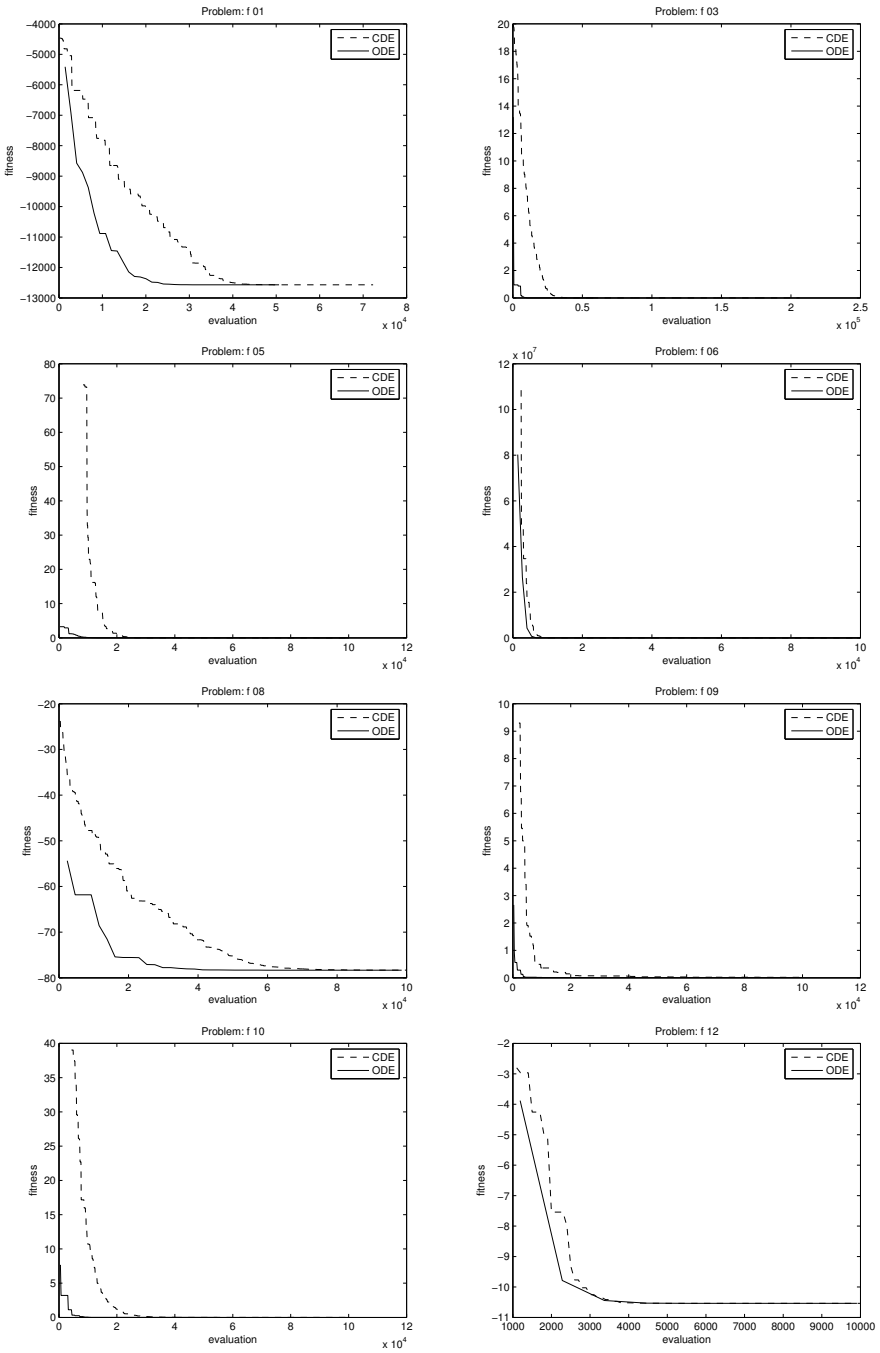

Fig. 1. Comparison with ODE and CDE on some convergence results of the test functions.