Innovative Systems Design and Engineering
ISSN 2222-1727 (Paper) ISSN 2222-2871 (Online)
Vol.6, No.2, 2015

A Cultivated Differential Evolution Algorithm using modified Mutation and Selection Strategy

Pooja[*1] Praveena Chaturvedi[1] Pravesh Kumar[2]
1. Department of Computer Science, Gurukula Kangri Vishwavidyalaya, Haridwar, India
2. Department of Mathematics, AMITY University, Gurgaon, India
* E-mail: mcapooja.singh2007@gmail.com

Abstract

In the present study a modified new variant of Differential Evolution (DE) is proposed, named Cultivated Differential Evolution (CuDE). This algorithm is different from basic DE in two ways. Firstly, the selection of the base vector for mutation operation is not totally randomized while in basic DE uniformly generated random numbers serve this task. Secondly, information preservation concept is used to generate population for the next generation. The performance of the proposed algorithm is validated on a bed of eight benchmark problems taken from literature and compared against the basic DE and some other variants of DE. The numerical results show that the proposed algorithm helps in formulating a better trade-off between convergence rate and efficiency.

Keywords: Differential Evolution, Mutation, Selection, Information Preservation, Population Segmentation

1. Introduction

Differential Evolution (DE) is a variant of Evolutionary Algorithm (EA) which was proposed by Storn and Price in 1997 (Storn & Price 1997). DE is used for solving global optimization problems over continuous spaces. It is a simple and efficient search engine which can handle nonlinear, non-differentiable and multimodal objective functions and a wide range of real life problems such as engineering design (Plagianakos *et al.* 2008), chemical engineering (Wang *et al.* 2007) and pattern recognition (Ilonen *et al.* 2003) and so on (Ali *et al.* 2013).

For enhancing the performance of DE, Several variants of the same are available in the literature. Some of which are: Modified DE (MDE) (Babu *et al*. 2006), DE with global and local neighborhood (DEGL) (Das *et al.* 2009), Cauchy mutation DE (CDE) (Ali & Pant 2010), Opposition based DE(ODE) (Rahnamayan *et al.* 2008), DE with Trigonometric Mutation (TDE) (Fan & Lampinen 2003), Self adaptive DE (SaDE) (Qin et al. 2009), Learning enhance DE (LeDE) (Cai *et al.* 2011), Fuzzy adaptive DE (FADE) (Liu & Lampinen 2005), DE with self-adaptive control parameter (JDE) (Brest *et al.* 2006), DE with random localization (DERL) (Kaelo & Ali 2006), Mixed mutation strategy based DE (Pant *et al.* 2009), DE with simplex crossover local search (DEahcSPX) (Noman & Iba 2008), adaptive DE with optional external archive (JADE) (Zhang & Sanderson 2009), and so on.

In the present study a different selection strategy named Reserve Selection is used for deciding the area from which the vectors for the mutation are to be chosen. This selection mechanism is based on the technique called 'population segmentation' a process of splitting into a Non-reserved Area and a Reserved area (Chen *et al.* 2007). The reserved area consists of best fit individuals (elite candidates) and non-reserved area maintains the rest of the population. The main operator of DE is mutation, which takes the solution vectors towards a global optimum. In the proposed algorithm, base vector is selected from the reserve area and the difference vectors are selected from the non-reserved area as to focus on the exploitation of the search space which then takes part in the mutation operation. Next the selection of the population for the next generation is formulated by using the information preservation concept taken from literature (Kumar *et al.* 2011).

The rest of the paper is organized as follows: Section 2 provides a compact overview of DE. Section 3 presents the proposed CuDE algorithm with flow chart. Benchmark problems and experimental settings are given in Section 4. Results and comparisons are reported in Section 5 and finally the conclusion derived from the present study is drawn in Section 6.

2. Basic Differential Evolution Algorithm

Differential Evolution (DE) is proposed by Storn and Price (Storn & Price 1997) is simple, fast and robust evolutionary algorithm. A brief introduction of the basic DE is given as follows:

DE starts with a population of NP solutions: $X_{i,G}$, $i = 1,..., NP$, where the index i denotes the i^{th} candidate solution of the population and G denotes the generation to which the population belongs. The three main operators of DE are mutation, crossover and selection.

- *Mutation*: The mutation operation of DE applies the vector difference between the existing population members in determining both the degree and direction of perturbation applied to the individual subject of the mutation operation. The mutation process at each generation begins by randomly selecting three solutions $\{X_{r1,} X_{r2,} X_{r3}\}$ in the population set of (say) NP elements. The i^{th} perturbed individual, $V_{i,G}$, is generated based on the three chosen solutions, as follows: i^{th}

$$DE/rand/1/bin: \; V_{i,G} = X_{r1,G} + F*(X_{r2,G} - X_{r3,G}) \tag{1}$$

Where, $i = 1,...,NP$, $r1, r2, r3 \in \{1,...,NP\}$ are randomly selected such that $r1 \neq r2 \neq r3 \neq i$, and F is the control parameter such that $F \in [0,1]$

- *Crossover:* Once the perturbed individual $V_{i,G} = (v_{1,i,G}, v_{2,i,G}, \ldots, v_{n,i,G})$ is generated, it is subjected to a crossover operation with target individual $X_{i,G} = (x_{1,i,G}, x_{2,i,G}, \ldots, x_{n,i,G})$, that finally generates the trial solution, $U_{i,G} = (u_{1,i,G}, u_{2,i,G}, \ldots, u_{n,i,G})$, as follows:

$$u_{j,i,G} = \begin{cases} v_{j,i,G} & if \;\; rand_j \leq C_r \vee j = jj \\ x_{j,i,G} & otherwise \end{cases} \tag{2}$$

- Where, $j = 1,...,n$, $jj \in \{1,...,n\}$ is a random parameter's index, chosen once for each i. The crossover rate $C_r \in [0,1]$ is set by the user.
- *Selection:* The selection scheme of DE also differs from that of other EAs. The population for the next generation is selected from the solution in current population and its corresponding trial solution according to the following rule:

$$X_{i,G+1} = \begin{cases} U_{i,G} & if \;\; f(U_{i,G}) \leq f(X_{i,G}) \\ X_{i,G} & otherwise \end{cases} \tag{3}$$

Thus, each solution of the temporary (trial) population is compared with its counterpart in the current population. The one with the lower objective function value will survive from the tournament selection to the population of the next generation. As a result, all the solutions for the next generation are as good as or better than their counterparts in the current generation. In DE, trial solution is not compared against all the solutions in the current generation, but only against one solution, its counterpart, in the current generation.

3. Proposed Algorithm

In this section, the modified operators of the basic algorithm are described. The proposed algorithm uses different mutation and selection operation from that of basic DE as for maximum exploration of the search space.

3.1 Mutation

Three different vectors are chosen from the whole population for performing the basic mutation operation, one of which is to be perturbed is called base vector (donor vector) and the rest two are known as difference vectors. The convergence speed of basic DE highly depends on the selection of the base vector (Kaelo & Ali 2006) and if the base vector is fitter than the difference vectors, the convergence speed of the DE will be better. The nature of the base vector has a direct impact on the newly generated mutant vector. The Proposed Strategy uses the same concept.

A new mutation scheme is proposed in the present study, in which sorting of the population is performed by taking the fitness of the individuals into consideration. Then the population is divided into two areas: one where the elite individuals are kept and the other one where the rest of the population resides. The proposed CuDE algorithm uses a Reserve Area (RA) maintaining the elite individuals which serves the base vector having size $NP*m\%$, where m is an user defined integer and the remaining population Non-Reserved Area (NRA) having size $NP-NP*m\%$, serves other two difference vectors which are uniformly randomly generated vectors from that population.

3.2 Selection

The CuDE algorithm uses information preservation concept of IPDE (Kumar *et al.* 2011) in which population NP of target vectors and population NP of trial vectors are combined. Now the size of the whole population becomes $2*NP$. Then sort the whole population and take the best fit NP individuals for the next generation. This strategy escapes us from loss of potential information.

3.3 Flowchart of the proposed CuDE algorithm

The flow chart of the proposed CuDE algorithm, explaining its working, is shown in Figure 1.

4. Benchmark Problems and Experimental Set-up

We have validated the proposed algorithm on 8 traditional benchmark problems (Zhang & Sanderson 2009). These test problems are given below in Table.1 with the parameters, population NP, Dimension D, F, C_r, m, Value to reach VTR, maximum Number of Function Evaluation NFE.

The algorithm is compiled in Dev C++ and is executed on Intel Core i3 PC with 4 GB RAM, 50 times for each test problem. The inbuilt *Rand () function* of C++ is used to generate the uniformly distributed random numbers. In every case, a run is terminated when NFE reaches the threshold value of maximum NFE.

Performance Criteria: To evaluate the performance of the algorithms, the performance criteria are selected from the literature (Rahnamayan *et al.* 2008, Tang *et al*. 2008). These criteria are:

- *NFEs:* The NFE is recorded when the VTR is reached before to reach maximum NFE i.e. we set the termination criteria as $|f_{opt} - f_{global}| \leq VTR$ and record average NFE of a successful run over 50 runs.
- *Error:* The average error $|f_{opt} - f_{global}|$ is recorded by using predefined maximum NFEs, in each run. Also the average and standard deviation of the fitness values are calculated.
- *Convergence Graph:* The convergence graphs show the mean fitness performance of the total runs, in the respective experiments.
- *Acceleration rate (AR) in %:* The acceleration rate is used to compare the convergence speeds between CuDE and other algorithms (Rahnamayan *et al.* 2008, Kumar *et al.* 2011). It is defined as follows:

$$AR = \frac{\sum NFE_I - \sum NFE_{II}}{\sum NFE_I}\ \%$$

Where I and II are two different algorithms.

5. Numerical Results and Comparisons

In Table 2, Comparison of CuDE is driven against basic DE, IPDE and DERL algorithms. As discussed above, the prime requirement for the proposed CuDE algorithm is the selection of RA, i.e. the value of m should be chosen neither very small nor very large. Now for testing purpose we have taken two cases for which the value of m is 20% and 30% respectively of the whole population. For both the cases CuDE is giving better results than basic DE, IPDE and DERL. But it is clear from the Table 2 that for m=20%, it is producing much optimized results.

The comparison is shown in terms of average NFEs, acceleration rate, mean fitness and standard deviation in Table 2, 3 and 4 respectively.

We can see from Table 2, with CuDE (m=20) each benchmark problem takes lesser NFEs to reach the VTR than basic DE, IPDE and DERL and total NFEs for each algorithm is shown in Table 2 with which the acceleration rate of CuDE (m=20) and CuDE (m=30) with respect to basic DE, IPDE and DERL are calculated and shown in Table 3.

In Table 4, Comparison in terms of average error and standard deviation of 50 runs is derived and from the table it is quite clear that CuDE (m=20) produces much better results than the other algorithms taken for comparison.

In Figure 2, convergence graphs between average error and NFE are shown for function F_1, F_2, F_5, F_6, F_7 respectively.

6. Conclusion

The objective of the present study is to use a modified selection strategy for choosing the individuals for mutation operation and to use the information preservation concept from IPDE for evaluating the new population for the next generation. In this study, the search space is divided into two areas (maximum exploration of the search space): first having the best fit individuals from which the donar vector is selected, second having low fit individuals from which the difference vectors are selected. The proposed CuDE algorithm is validated on a set of 8 standard benchmark problems and for m=20, CuDE produces best results. Numerical results derive that the proposed algorithm performs better than basic DE and some other variants of it.

References

Storn, R. & Price, K. (1997), "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces", *Journal of Global Optimization* **11**, Springer, 341–359.

Plagianakos, V., Tasoulis, D. & Vrahatis, M. (2008), "A Review of Major Application Areas of Differential Evolution", *In: Advances in Differential Evolution, Berlin* **143**, Springer, 19- 238.

Wang, Y., Zhang, J. & Zhang, G. (2007), "A Dynamic Clustering Based Differential Evolution Algorithm for Global Optimization", *European Journal of Operational Research* **183(1)**, Elsevier, 56–73.

Ilonen, J., Kamarainen, J.K. & Lampinen, J. (2003), Differential Evolution Training algorithm for Feed-Forward Neural Networks, *Neural Process Letters* **17(1)**, Springer, 93–105.

Ali, M., Pant, M. & Abraham, A. (2013), "Unconventional initialization methods for differential evolution", *Applied Mathematics and Computation* **219**, Elsevier, 4474-4494.

Babu, B.V. & Angira, R. (2006), "Modified differential evolution (MDE) for optimization of non-linear chemical processes", *Computer and Chemical Engineering* **30**, Elsevier, 989-1002.

Das, S., Abraham, A., Chakraborty, U. & Konar, A. (2009), "Differential evolution using a neighborhood based mutation operator", *IEEE Transaction of Evolutionary Computing* **13(3)**, 526–553.

Ali, M. & Pant, M. (2010), "Improving the performance of differential evolution algorithm using cauchy mutation", *Soft Computing*, Springer, doi:10.1007/s00500-010-0655-2.

Rahnamayan, S., Tizhoosh, H. & Salama, M. (2008), "Opposition based differential evolution", *IEEE Transaction of Evolutionary Computing* **12(1),** 64–79.

Fan, H. & Lampinen, J. (2003), "A trigonometric mutation operation to differential evolution", *Journal of Global Optimization* **27**, Springer, 105-129.

Qin, A.K., Huang, V.L. & Suganthan, P. N. (2009), "Differential evolution algorithm with strategy adaptation for global numerical optimization", *IEEE Transaction of Evolutionary Computing* **13(2)**, 398–417.

Cai, Y., Wang, J. & Yin, J. (2011), "Learning enhanced differential evolution for numerical optimization", *Soft Computing*, Springer, doi: 10.1007/s00500-011-0744-x.

Liu, J. & Lampinen, J. (2005), "A fuzzy adaptive differential evolution algorithm", *Soft Computing Fusion Found Method Appl.* **9(6)**, 448–462.

Brest, J., Greiner, S., Boskovic, B., Mernik, M. & Zumer, V. (2006), "Self adapting control parameters in differential evolution: a comparative study on numerical benchmark problems", *IEEE Transaction of Evolutionary Computing* **10(6)**, 646–657.

Kaelo, P. & Ali, M. M. (2006), "A numerical study of some modified differential evolution algorithms", *European Journal of Operational Research* **169**, Elsevier, 1176-1184.

Pant, M., Ali, M. & Abraham, A. (2009), "Mixed mutation strategy embedded differential evolution", *IEEE Congress on Evolutionary Computation*, 1240-1246.

Noman, N. & Iba, H. (2008), "Accelerating differential evolution using an adaptive local search", *IEEE Transaction of Evolutionary Computing* **12(1)**, 107–125.

Zhang, J. & Sanderson, A. (2009), "JADE: Adaptive differential evolution with optional external archive", *IEEE Transaction of Evolutionary Computing* **13(5)**, 2009, 945–958.

Chen, Y., Hu, J., Hirasawa, K. & Yu, S. (2007), "GARS: An Improved Genetic Algorithm with Reserve Selection for Global Optimization", *Proceedings of the 9th annual conference on Genetic and evolutionary computation (GECCO '07)*, 1173-1178.

Kumar, P., Pant, M. & Singh, V.P. (2011), "Information Preserving Selection Strategy for Differential Evolution Algorithm", *Proc IEEE World Congress on Information and Communication Technology (WICT 2011), Mumbai, India*, 466-470.

Tang, K., Yao, X., Suganthan, P.N., MacNish, C., Chen, Y.P., Chen, C.M. & Yang, Z. (2008), "Benchmark Functions for the CEC'2008 Special Session and Competition on Large Scale Global Optimization", *Technical Report CEC-08*, 1-18.

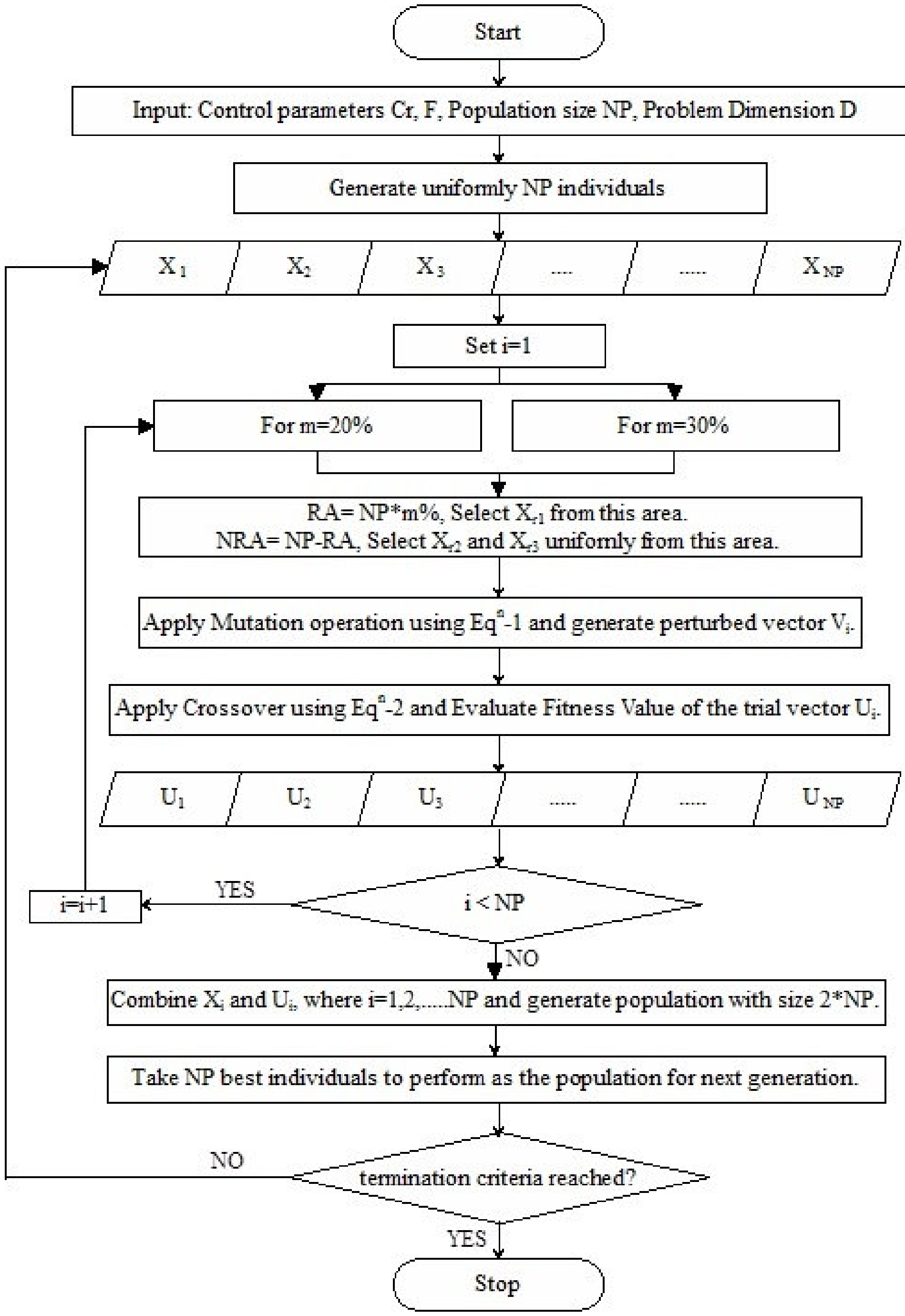

Figure 1. Flowchart of CuDE algorithm

Table 1. Test problems with their Parameter Settings

Traditional Benchmark function		**Parameterization**					
		NP	**D**	**F, C_r**	**m**	**VTR**	**Max. NFE**
F_1	Sphere Function	100	30	0.5, 0.9	20, 30	10^{-8}	150000
F_2	Ackley Function	100	30	0.5, 0.9	20, 30	10^{-8}	150000
F_3	Rastrigin Function	100	30	0.5, 0.9	20, 30	10^{-8}	300000
F_4	Rosenbrock Function	100	30	0.5, 0.9	20, 30	10^{-8}	500000
F_5	Noise Function	100	30	0.5, 0.9	20, 30	10^{-2}	300000
F_6	Schwefel 1.2 Function	100	30	0.5, 0.9	20, 30	10^{-8}	500000
F_7	Schwefel 2.22 Function	100	30	0.5, 0.9	20, 30	10^{-8}	200000
F_8	Griewank Function	100	30	0.5, 0.9	20, 30	10^{-8}	200000

Table 2. Experimental Results and Comparison of CuDE (m=20, 30) in term of Average NFE of 50 runs for Standard Test Problems

Fun.	DE	IPDE	DERL	m=20	m=30
F_1	116000	80030	67400	32930	42700
F_2	162410	123510	86410	51460	67825
F_3	NA	NA	NA	NA	NA
F_4	437970	204000	262000	126400	153470
F_5	117030	117170	76480	42700	54175
F_6	412040	287180	211870	131390	156250
F_7	174030	131860	96300	53670	72550
F_8	106090	82970	59800	35320	44050
$\sum$	1525570	1026720	860260	473870	591020

Table 3. Acceleration Rate of CuDE with respect to DE, IPDE and DERL

w.r.t.	DE	IPDE	DERL
m=20	68.94%	53.85%	44.92%
m=30	61.26%	42.44%	31.30%

Table 4. Average error and Standard Deviation* for 50 runs when NFE is fixed

Fun.	DE	IPDE	DERL	m=20	m=30
F_1	3.56e-14 1.83e-14*	7.09e-20 4.80e-20*	5.04e-30 7.75e-31*	7.26e-54 2.71e-54*	2.87e-40 1.82e-40*
F_2	6.96e-08 3.03e-08*	5.64e-11 1.66e-11*	7.55e-15 0.0e+00*	3.99e-15 0.0e+00*	4.89e-15 1.54e-15*
F_3	NA	NA	NA	NA	NA
F_4	2.04e-10 3.39e-10*	2.13e-19 2.99e-19*	1.73e-29 3.00e-29*	5.44e-45 1.91e-45*	3.99e-30 1.42e-30*
F_5	4.77e-03 7.78e-04*	3.81e-03 2.70e-04*	1.88e-03 3.39e-04*	1.62e-03 4.36e-04*	1.57e-03 3.76e-04*
F_6	2.37e-11 2.04e-11*	1.76e-17 1.74e-17*	1.44e-24 1.91e-24*	1.56e-38 1.56e-38*	5.99e-35 4.29e-35*
F_7	3.65e-10 1.27e-10*	6.57e-14 4.88e-14*	2.50e-20 1.07e-20*	1.25e-36 5.88e-37*	2.59e-27 1.18e-27*
F_8	0.0e+00 0.0e+00*	0.0e+00 0.0e+00*	0.0e+00 0.0e+00*	0.0e+00 0.0e+00*	0.0e+00 0.0e+00*

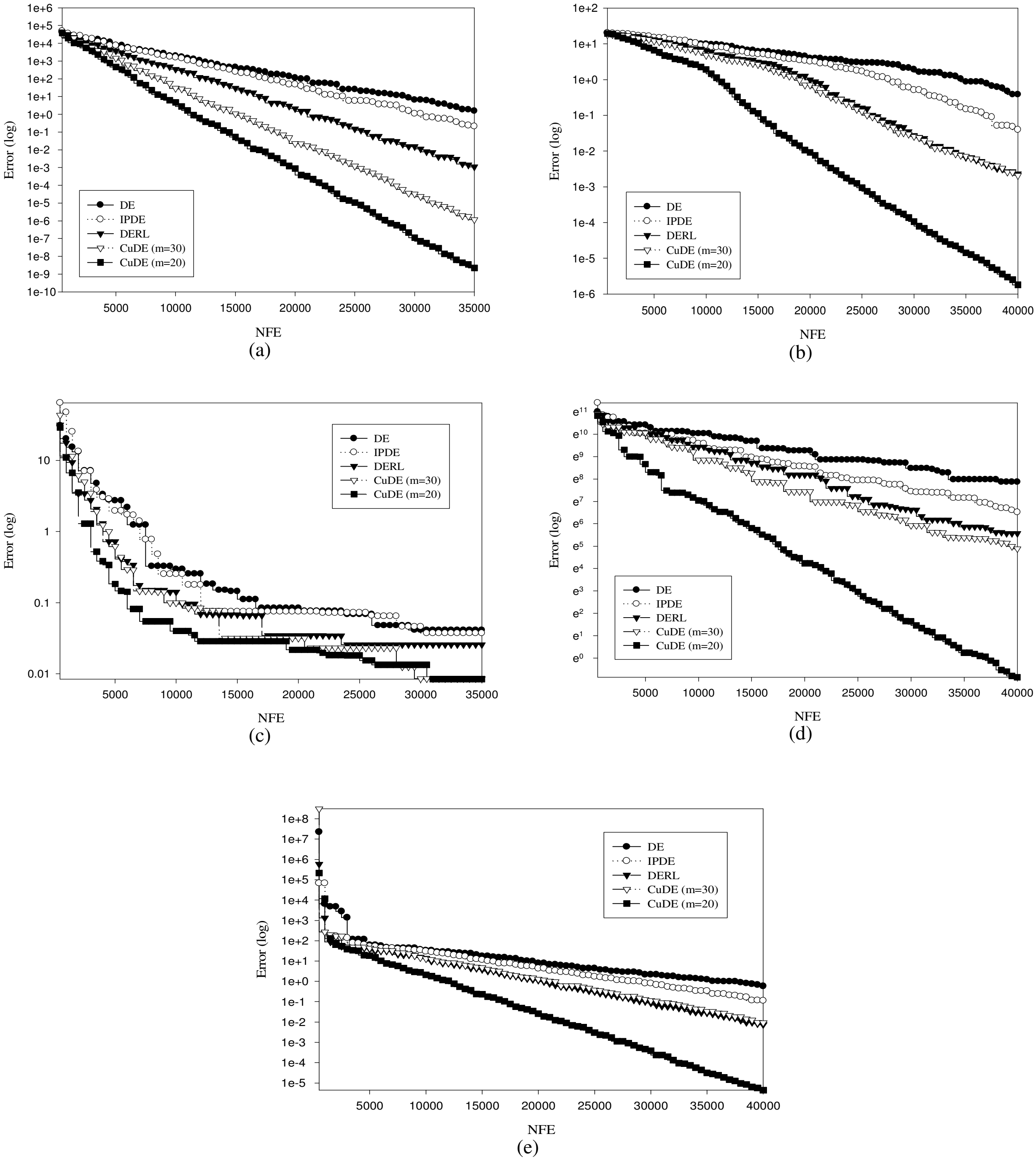

Figure 2. Convergence graph between Average Error and NFE for (a) Sphere, (b) Ackley, (c) Noise, (d) Schwefel 1.2, (e) Schwefel 2.22 respectively