# Differential Evolution with Parent Centric Crossover

Millie Pant[*] millifpt@iitr.ernet.in
Musrrat Ali[*] ali82dpt@iitr.ernet.in
V.P. Singh[*] singhvp2@yahoo.co.in.
*Department of Paper Technology Indian Institute of Technology Roorkee-247667.

## *Abstract*

*Differential Evolution (DE) has emerged as a powerful tool for solving optimization problems in the last few years. However, the convergence rate of DE still does not meet all the requirements, and attempts to speed up differential evolution are considered necessary. In order to improve the performance of DE, we propose a modified DE algorithm called DEPCX which uses parent centric approach to manipulate the solution vectors. The performance of DEPCX is evaluated on a test bed of five functions. Numerical results are compared with original differential evolution (DE) and with TDE, another recently modified version of DE. Empirical results indicate that this modification enables the algorithm to get a better transaction between the convergence rate and robustness.*

Keywords: Stochastic optimization, differential evolution, mutation operation, crossover.

## 1. Introduction

Evolutionary Algorithms (EAs) are general-purpose stochastic search methods imitating the phenomena of biological evolution. These are also known as population based search techniques as they work with a population of solutions rather than a single solution, which makes them different from optimization methods, such as hill-climbing [1] and simulated annealing [2].

One of the reasons of the success of EAs is their population based strategy which prevents them from getting trapped in a local optimal solution and consequently increases their probability of finding a global optimal solution. Thus, EAs can be viewed as global optimization algorithms. Some frequently used EAs include Evolutionary Programming (EP) [3], Evolution Strategies (ES) [4] and Genetic Algorithms (GA) [5].

DE, proposed by Storn and Price [6], is comparatively a newer addition to the class of EAs. DE is similar to GAs in the sense that it uses same evolutionary operators like mutation and crossover for guiding the particles towards the optimum solution. Nevertheless, it's the application of these operators that makes DE different from GA. The main difference between GAs and DE is that; in GAs, mutation is the result of small perturbations to the genes of an individual while in DE mutation is the result of arithmetic combinations of individuals. Also in DE, mutation plays a prominent role whereas, in GA, crossover is the major operator. At the beginning of the evolution process, the mutation operator of DE favors exploration. As evolution progresses, the mutation operator favors exploitation. Hence, DE automatically adapts the mutation increments (i.e. search step) to the best value based on the stage of the evolutionary process. Mutation in DE is therefore not based on a predefined probability density function.

DE is easy to implement, requires little parameter tuning and exhibits fast convergence. It has been successfully applied to solve a wide range of optimization problems such as clustering [7], unsupervised image classification [8], digital filter design [6], optimization of non-linear functions [9], global optimization of non-linear chemical engineering processes [10] and multi-objective optimization [11] etc.

Although, DE has given good results in most of the real life and test problems, researchers have observed that its convergence rate do not meet the desired expectations in some of the cases. In order to overcome this drawback of DE, we propose a modified version which uses parent centric approach to generate new solution vectors. This modification provides a measure to tune the balance between the convergence rate and the robustness of the algorithm. It accelerates the convergence velocity of DE algorithm locally without compromising the quality of solution.

Remaining of the paper is organized as follows; in Section 2, we give a brief description of DE. Section

3, describes the PCX operator. In section 4, we explain the proposed DEPCX algorithm. Benchmark problems are given in section 5. Section 6 deals with experimental settings and parameter selection. Section 7 makes comparison of algorithms. The paper finally concludes with section 8.

## 2. Differential Evolution (DE):

In this section, we briefly describe the classical DE algorithm as given by Storn and Price. DE attempts to replace each point in population set S with a new better point. Therefore, in each generation, NP (population size) competitions are held to determine the members of S for the next generation. The *i*th (i = 1, 2, . . . , NP) competition is held to replace $X_{i,G}$ in S. Considering $X_{i,G}$ as the target point, a trial point $U_{i,G+1}$ is found from two points (parents), the point $X_{i,G}$, i.e., the target point and the mutated point $V_{i,G+1}$ are determined by the mutation operation. In its mutation phase, DE randomly selects three distinct points from the population. The *i*th perturbed individual, $V_{i,G+1}$, is therefore generated based on the three chosen individuals as follows:

$V_{i,G+1} = X_{r3,G} + F * (X_{r1,G} - X_{r2,G})$ (1)

Where, $i$ = 1. . . NP, r1, $r_2$, $r_3 \in \{1, . . . ,NP\}$ are randomly selected such that $r_1 \neq r_2 \neq r_3 \neq i$. Scaling factor F ($F \in [0, 1.2]$ ) is a control parameter of the DE algorithm. The perturbed individual, $V_{i,G+1} = (v_{1,i,G+1}, . . . , v_{D,i,G+1})$, and the current population member, $X_{i,G} = (x_{1,i,G}, . . . , x_{D,i,G})$, are then subject to the crossover operation, that finally generates the population of candidates, or “trial” vectors,$U_{i,G+1} = (u_{1,i,G+1}, . . . , u_{D,i,G+1})$, as follows:

$$u_{j,i.G+1} = \begin{cases} v_{j,i.G+1} \; if \; rand_j \leq C_r \vee j = k \\ x_{j,i.G} \quad otherwise \end{cases} \tag{2}$$

Where j, k ∈ {1,…, D} k is a random parameter index, chosen once for each *i*, Cr is crossover factor ∈ [0, 1].

The population for the next generation is selected from the individuals in current population and its corresponding trial vector according to the following rule:

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

Thus, each individual of the temporary (trial) population is compared with its counterpart in the current population. The one with the lower objective function value survives the tournament selection and go to the next generation. As a result, all the individuals of the next generation are as good or better than their counterparts in the current generation. A notable point in DE’s selection scheme is that a trial vector is not compared against all the individuals in the current generation, but only against one individual, its counterpart, in the current generation.

## 3. Parent Centric Crossover

Parent centric crossover (PCX) was first given by Deb [12, 13]. It is a multiparent operator consisting of more than one parent. Its working may be described as follows: in the beginning $\mu$ (say three) parents are selected for which the mean vector $\vec{g}$ is computed. From the three chosen parents, one parent $\vec{x}_{p,G}$ is selected at random to generate offspring and direction vector $\overrightarrow{d_p} = \overrightarrow{x_{p,G}} - \vec{g}$ is calculated. From the remaining ($\mu$-1) parents perpendicular distances $D_i$ to the line $\overrightarrow{d_p}$ are computed and their average $\overline{D}$ is found. The offspring is created as follows:

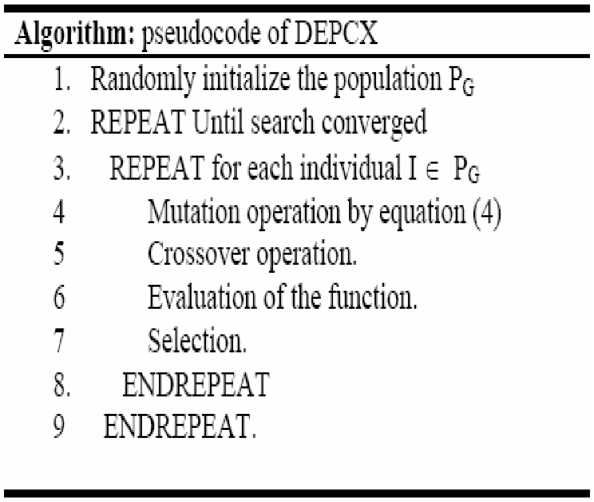
$$V_{i,G+1} = \overrightarrow{x_{p,G}} + w_\varsigma d_p + \sum_{i=1, i \neq p}^{\mu} w_\eta \overline{D} \overrightarrow{e_i} \tag{4}$$

Where $\overrightarrow{e_i}$ are the ($\mu$-1) orthonormal bases that span the sub space perpendicular to $\overrightarrow{d_p}$. The parameters $w_\varsigma$ and $w_\eta$ are zero mean normally distributed variables with variances $\sigma_\varsigma^2$ and $\sigma_\eta^2$ respectively.

## 4 Proposed DEPCX

The proposed DEPCX is a simple and modified version of the original DE algorithm. The PCX operator described in section 3 creates new candidate solutions (children) near by the parent vectors. In [12,13], PCX is a normal crossover operator, whereas in DEPCX it is treated as a mutant vector and is applied to each candidate solution vector. Thus DEPCX differs from original DE only in mutation phase. Crossover and selection scheme for DEPCX is same as in original DE. The presence of PCX operator in DE helps in enhancing its performance by giving it a chance of exploring its neighborhood more efficiently. The pseudo code of DEPCX algorithm is given in table 1.

**Table 1: Pseudo code of DEPCX.**



```
Algorithm: pseudocode of DEPCX
1. Randomly initialize the population $P_G$
2. REPEAT Until search converged
3.     REPEAT for each individual $I \in P_G$
4          Mutation operation by equation (4)
5          Crossover operation.
6          Evaluation of the function.
7          Selection.
8.     ENDREPEAT
9    ENDREPEAT.
```

## 5 Benchmark problems

The performance of proposed DEPCX is evaluated on a test bed of five standard, unconstrained benchmark problems taken from the literature [14]. All the test problems are scalable and are tested for 10, 50 and 100 number of variables. Except for the first function f1, all the other test functions are highly multimodal in nature. The degree of multimodality for functions f2 – f5, keeps on increasing with the increase in the number of variables thereby, increasing the complexity of the problem. Function f4 is a noisy function due to the presence of a uniformly distributed random noise.

**Table 2: benchmark problems**

| Function | Dim | Ranges | Min value |
|---|---|---|---|
| $f_1(X) = \sum_{i=1}^{n} x_i^2$ | 10,50,100 | [-5.12,5.12] | 0 |
| $f_2(X) = -20 * \exp\left(-.2\sqrt{1/n\sum_{i=1}^{n} x_i^2}\right) - \exp\left(1/n\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$ | 10,50,100 | [-32, 32] | 0 |
| $f_3(X) = \sum_{i=1}^{n}\left(-x_i \sin\left(\sqrt{\lvert x_i\rvert}\right)\right)$ | 10,50,100 | [-500, 500] | - 418.9829*n |
| $f_4(X) = \left(\sum_{i=1}^{n} i \times x_i^4\right) + rand[0,1[$ | 10,50,100 | [-1.28, 1.28] | 0 |
| $f_5(X) = \sum_{i=1}^{n}\left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$ | 10,50,100 | [-5.12, 5.12] | 0 |

**Table 3: Mean of fitness and diversity for all algorithms on benchmark problems of dimensionality 10, 50,100(mean of 30 runs) for each problem.**

| Fun | Dim. | DE | | DEPCX | | TDE | |
|---|---|---|---|---|---|---|---|
| | | Fitness | Diversity | Fitness | Diversity | Fitness | Diversity |
| $f_1$ | 10 | 2.43176e-006 | 0.00398081 | **7.15463e-008** | 0.0034912 | 3.00576e-006 | 0.0084662 |
| | 50 | 7.80355e-006 | 0.00448866 | **3.23314e-006** | 0.00419823 | 5.55495e-006 | 0.00390207 |
| | 100 | **3.0074e-005** | 0.00696293 | 4.60153e-005 | 0.00855983 | 3.42414e-005 | 0.00815677 |
| $f2$ | 10 | 8.91482e-006 | 3.50226e-005 | **3.32626e-006** | 6.51266e-005 | 1.17702e-005 | 6.57102e-05 |
| | 50 | 3.29725e-005 | 0.000102938 | **1.99416e-005** | 6.0943e-005 | 2.6708e-005 | 8.49865e-05 |
| | 100 | 0.000124886 | 0.000417206 | 8.60006e-005 | 0.000286457 | **7.28019e-005** | 0.000251843 |
| $f3$ | 10 | -4189.83 | 0.00936212 | -4189.83 | 0.0117356 | -4189.83 | 0.0148009 |
| | 50 | -20949.1 | 0.0170014 | -20949.1 | 0.00334897 | -20949.1 | 0.0110136 |
| | 100 | -41898.3 | 0.0158414 | -41898.3 | 0.0261631 | -41898.3 | 0.0114593 |
| $f4$ | 10 | **0.00065363** | 0.209331 | 0.00155938 | 0.152251 | 0.000671257 | 0.211284 |
| | 50 | 0.022227 | 0.50223 | **0.0136434** | 0.399991 | 0.0161357 | 0.482207 |
| | 100 | **0.0525307** | 0.682851 | 0.0538623 | 0.623006 | 0.0582792 | 0.640255 |
| $f5$ | 10 | 1.17102e-006 | 0.000291831 | **5.42596e-008** | 0.000259009 | 6.52855e-007 | 0.000189773 |
| | 50 | 6.72631e-006 | 0.000190036 | **3.06816e-006** | 0.000193599 | 5.91064e-006 | 0.000380697 |
| | 100 | 3.46447e-005 | 0.000554309 | 3.90783e-005 | 0.0005885 | **1.05083e-005** | 0.000329286 |

**Table 4: mean of generation for all algorithms on benchmark problems of dimensionality 10, 50,100(mean of 30 runs) for each problem.**

| | Mean Generation | | | |
|---|---|---|---|---|
| Functions | Dimension | DE | DEPCX | TDE |
| $f_1$ | (10,50,100) | (173,925,1175) | (125,750,1550) | (161,879,1550) |
| $f_2$ | (10,50,100) | (339,1575,2800) | (200,1300,2600) | (300,1466,2550) |
| $f_3$ | (10,50,100) | (275,1350,2750) | (175,1475,2525) | (265,1300,2550) |
| $f_4$ | (10,50,100) | (8000,8000,8000) | (8000,8000,8000) | (8000,8000,8000) |
| $f5$ | (10,50,100) | (275,1600,3250) | (250,1389,5625) | (281,1425,3050) |

## 6 Experimental settings and Parameter Selection

Population size for all the algorithms is taken as NP (=3*D), where D is the dimension of the problem. Crossover rate (Cr) and scaling factor (F) are fixed at 0.01 and 0.5 respectively; for TDE, the trigonometric mutation probability Pt is 0.01; Maximum generations for all the algorithms are fixed at 8000; PCX parameters $\sigma_\varsigma$ and $\sigma_\eta$ are taken as 0.1 each. All the algorithms were run 30 times for each of the test problems to determine the mean fitness and diversity. In every case, a run was terminated when the function values of all points in population S were identical to an accuracy of four decimal places, i.e., $|f_{\max} - f_{\min}| \le \varepsilon = 10^{-4}$ or when the maximum generation was reached.

Diversity measure for all the algorithms is calculated as

$$\text{Diversity} = \frac{1}{NP}\sum_{i=1}^{NP}\sqrt{\sum_{j=1}^{D}\left(x_{ij} - \overline{x_j}\right)^2}$$

Where $\overline{x_j}$ is the average of the $j^{th}$ dimension over all individuals, i.e.

$$\overline{x_j} = \frac{1}{NP}\sum_{i=1}^{NP} x_{ij}$$

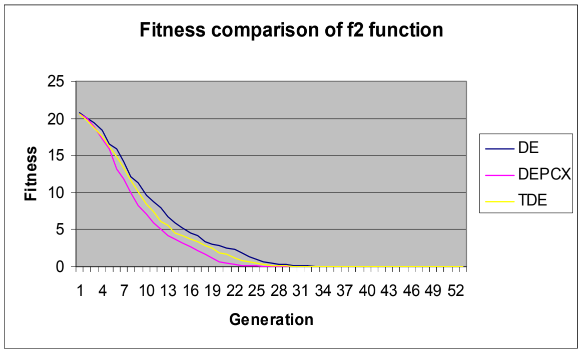
We do not claim these values to be the optimal for any given problem, in general, but empirically they are good values to choose. All the algorithms are executed on a Pentium IV PC using Dev C++

## 7 Comparisons of algorithms

The performance is DEPCX is compared with the classical DE of Storn and Price and with Trigonometric Differential Evolution TDE [15]. TDE is one of the latest versions of DE and has reportedly given better performance than the original DE.

Results of our experiments are reported in table 3 and 4. For each function the results are arranged as mean fitness and diversity of thirty runs and dimension ten, fifty and hundred. In figure 1, the graphs represent the mean of the best evaluation in thirty runs for the functions of fifty dimensions. And in figure 2, the graphs represent the diversity of functions for fifty dimensions. Because of limited space, only some representative graphs for different functions are presented.

From the numerical results given in Table 2, it can be observed that out of the 15 test cases considered, DEPCX gave superior solution in 7 cases (success rate = 46.2%) in comparison to TDE and DE. TDE and DE gave good performances in 2 and 3 cases respectively. TDE gave better performance than DEPCX and DE for 100 variables problems. Surprisingly, for f4 which is a noisy function, DE gave best results followed closely by DEPCX and TDE. Also we would like to add that in terms of fitness function values, the performance of all the algorithms vary from each other only marginally. However, from Table 3, the performance comparison of the algorithms becomes more visible when the convergence rate is considered. From Table 3, it can be seen that for all the test problems, except for f4, DEPCX gave a visibly better convergence rate.



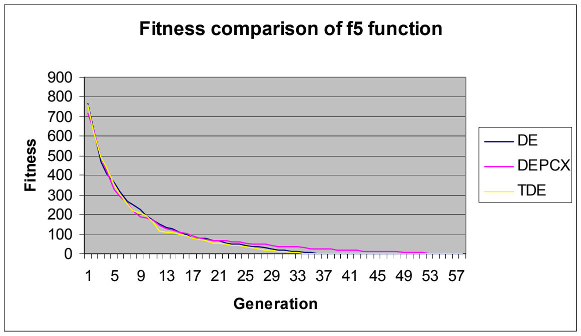
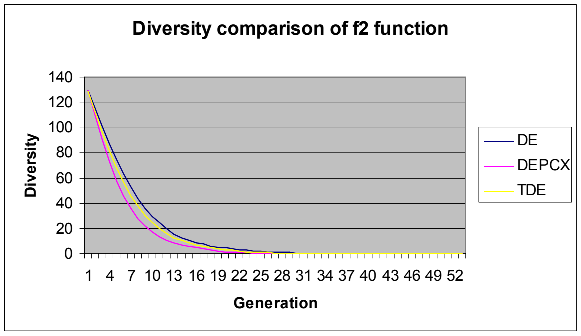


**Figure 1: figures show fitness Vs generation for function f2 and f5 of fifty dimension.**



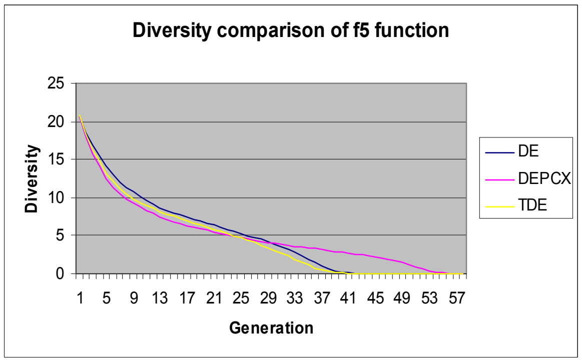


**Figure 2: figure show diversity Vs generation for the function f2 and f5 of fifty dimension.**

## 8. Conclusion

A parent centric approach to DE is proposed in this paper in order to improve the convergence rate without compromising with the quality of the solution. The proposed DEPCX algorithm helps in attaining the better convergence rate in comparison to DE and TDE with a good solution quality. The parent centric approach i.e. getting a solution nearby the parent helps in exploration of the neighborhood in order to get a better candidate for the next generation. We would like to maintain that although in terms of solution quality, no algorithm can be claimed as a clear winner but in terms of convergence rate DEPCX definitely gave a better performance. In future we shall apply DEPCX for solving more complex problems and also we will extend it for solving constrained problems.

## References

[1] Z. Michalewicz, D. Fogel, "How to Solve It: Modern Heuristics". Springer, Berlin. 2000.

[2] Van Laarhoven, P. Aarts,"Simulated Annealing: Theory and Applications." Kluwer Academic Publishers 1987.

[3] L.Fogel," Evolutionary programming in perspective: The top-down view. In: Zurada, J.M., Marks, R. Jr., Robinson, C. (Eds.), Computational Intelligence: Imitating Life. IEEE Press, Piscataway, NJ, USA. 1994.

[4] T.Back, F. Hoffmeister, H.Schwefel, "A survey of evolution strategies." In: Proceedings of the Fourth International Conference on Genetic Algorithms and their Applications, 1991.pp. 2–9.

[5] D. Goldberg, "Genetic Algorithms in Search Optimization and Machine Learning." Addison-Wesley. 1989.

[6] R.Storn,"Differential evolution design for an IIR-filter with requirements for magnitude and group delay". Technical Report TR-95-026, International Computer Science Institute, Berkeley, CA 1995..

[7] S.Paterlini, T.Krink, "High performance clustering with differential evolution." In: Proceedings of the IEEE Congress on Evolutionary Computation, vol. 2, 2004, pp. 2004–2011.

[8] M.Omran, A.Engelbrecht, A.Salman, Differential evolution methods for unsupervised image classification." In: Proceedings of the IEEE Congress on Evolutionary Computation, vol. 2, 2005a, pp. 966–973.

[9] B. Babu, R.Angira, "Optimization of non-linear functions using evolutionary computation". In: Proceedings of the 12th ISME International Conference on Mechanical Engineering, India, 2001, pp. 153–157.

[10] R.Angira, B.Babu, "Evolutionary computation for global optimization of non-linear chemical engineering processes." In: Proceedings of International Symposium on Process Systems Engineering and Control, Mumbai, 2003,pp. 87–91.

[11] H. Abbass, "A memetic pareto evolutionary approach to artificial neural networks" Lecture Notes in Artificial Intelligence, vol. 2256. Springer, 2002a, pp. 1–12.

[12] K Deb, A.Anand D Joshi "A computationally efficient evolutionary algorithm for real-parameter optimization." Evol Comput J 2002, 10(4): 371–395

[13] K Deb, "A population-based algorithm-generator for real-parameter optimization" .soft Comput J 2005, 9: 236–253.

[14] X.Yao, Y.Liu, "Fast evolutionary programming" in L.J.Fogel, P.J.Angeline and T.Back, editors, Proceeding of the 5th Annual Conference on Evolutionary Programming MIT Press, 1996, pp 451-460.

[15] Hui-Yuan Fan, Jouni Lampinen, "A Trigonometric Mutation Operation to Differential Evolution," Journal of Global Optimization 2003, 27:105-129.