Real Parameter Optimization Using Levy Distributed Differential Evolution

Nanda Dulal Jana[1], Aditya Narayn Hati[1], Rajkumar Darbar[2], and Jaya Sil[3]

[1] Dept of Information Technology, NIT Durgapur, Durgapur-713209, India
[2] School of Information Technology, IIT Kharagpur, Kharagpur-721302, India
[3] Dept of Computer Science & Engineering, BESU, Shibpur, India
{nanda.jana,smartyadi88,rajdarbar.r}@gmail.com,
js@cs.becs.ac.in

Abstract. Differential Evolution (DE) algorithm is a real parameter encoded evolutionary algorithm for global optimization. In this paper, Levy distributed DE (LevyDE) has been proposed. The main objective of LevyDE algorithm is to introduce a parameter control mechanism in DE based on levy distribution, a heavy tail distribution, for both the mutation and crossover operations. The main emphasis of this paper is to analyze the behavior and dynamics of the LevyDE and make a comparison with other standard algorithms such as DE/best/1/bin **[1]**, DE/rand/1/bin **[1]** and ACDE **[8]** on basis of CEC'05 benchmark functions.

1 Introduction

The Global optimization problem **[2]** can be formalized as a pair $\langle s,f \rangle$, where $S \subseteq R^D$ and $f : S \to R$ is a D-dimensional real valued function. The problem is to find a point $x^* \in S$ such that $f(x^*) \leq f(x), \forall\, x \in S$. Here, f does not need to be continuous but to be bounded. There is no exact solution for this problem. Therefore, heuristic search algorithms are used to solve this problem efficiently. The Differential evolution (DE) outperforms the other existing algorithms in robust performance and faster execution to find global optimal solution. The DE **[1]** is a population based stochastic meta-heuristic algorithm for global optimization which is known for its simplicity, effectiveness and robustness. Sometime practical experience shows that it does not perform up to the expectations. The performance of the DE depends on the balance of exploration and exploitation strategies like other evolutionary algorithms. If the balance is hampered, the problem like stagnation of the population, premature convergence etc may appear. The situation when the algorithm does not show any improvement, though it accepts new individuals in the population is known as stagnation. Besides this, premature convergence arises when there is a loss of diversity in the population **[3]**. It generally arises when the objective function is non-separable multimodal having several local and global optimums. To keep the balance between exploration and exploitation strategies, two techniques such as parameter tuning and parameter controlling can be used according to Eiben et al **[4]**. Parameter tuning is the commonly practiced approach that amounts to finding good values for the parameters before the run of the algorithm and then running the algorithm using these values, which remain fixed during the run. This is a trial & error approach. Parameter control is an alternative which controls the parameter in every generation following some specified rules. It has three categories: (a) deterministic parameter control **[5,6]** (b) adaptive parameter control **[7, 8]** and (c) self adaptive parameter control **[9, 10, 11]**.

In this paper, a self adaptive control mechanism is introduced to control the scale parameter and crossover probability using Levy distribution with variable location parameter, because it produces significant amount of changes in the control parameters and reduces manual parameters setting. To validate the proposed strategy, experiments are performed on 10 benchmark test problems that were introduced by Suganthan et al. in CEC 2005 **[2]**. The obtained results are compared against the two conventional DE algorithms **[1]** and Adaptive Differential Evolution Algorithm (ACDE) proposed by Millie Pant et al **[8]**.

2 Related Works

There is a lot of research done on the parameter control mechanisms. The main objective of these researches is to reduce manual control and initialization of the scale factor and crossover probability and to reach the global optimum efficiently. Abbas et al. [14] introduced a self-adaptive approach to DE parameters using variable step length generated by a Gaussian distribution. These parameters are evolved during the optimization process. Liu and Lampinen introduced an adaptive parameter control mechanism using Fuzzy controller in [7]. Yang et al. [10] proposed a self adaptive differential evolution algorithm with neighborhood search (SaNSDE). SaNSDE proposes three self-adaptive strategies: self-adaptive choice of the mutation strategy between two alternatives, self-adaptation of the scale factor F, and self-adaptation of the crossover rate Cr. Qin and Suganthan [11] proposed a self-adaptive DE, called SaDE. In this proposed method, scale factor and mutation probability need not require any predefining. Teo [9] proposed a self-adaptive strategy where population size parameter is adapted during the optimization. A new Differential Evolution algorithm based on Adaptive Control parameters (ACDE) is introduced by Pant et al in [8].

3 Differential Evolution Algorithm

Differential Evolution (DE) is a stochastic, population-based optimization algorithm which was proposed by Storn and Price in 1996 **[1]**. It was developed to optimize real parameter, real valued functions. Global optimization is necessary in fields such as engineering, statistics and finance. But many practical problems have objective functions that are non-differentiable, non-continuous, non-linear, noisy, flat, multi-dimensional or have many local minima, constraints or stochasticity. Such problems are difficult if not impossible to solve analytically. DE can be used to find approximate solutions to such problems. DE is an Evolutionary algorithm whose initial population is of size N_p and dimension is D. The population matrix is initialized as follows:

$$X_i = B_i^L + \text{rand}(0,1) * \left(B_i^U - B_i^L\right) \tag{1}$$

Each initialized vector is called 'target vector'. Here, B_i^U and B_i^L represent upper and lower bound of X_i respectively.

The classical DE has three operators: mutation, crossover and selection.

Mutation: Mutation is a kind of exploration technique that can explore the search space rapidly. It creates donor vectors. Mutation strategies are as follows:

1. DE/rand/1:

$$V_{ij} = X_{r_1 j} + F * \left(X_{r_2 j} - X_{r_3 j}\right);\ r_1 \neq r_2 \neq r_3 \tag{2}$$

2. DE/best/1:

$$V_{ij} = X_{best,j} + F * \left(X_{r_1 j} - X_{r_2 j}\right)\ ;\ r_1 \neq r_2 \tag{3}$$

Here F is a control parameter called scale factor. It controls the speed of convergence towards optimal solutions depending on its value. Range of f is given as [0, 2].

Crossover: In DE, Crossover is an exploration technique. It generates trial vector U_{ij}. There are two types of crossover strategies i.e. binomial crossover and exponential crossover. Generally, binomial crossover is used in DE and it is very similar to uniform crossover in evolutionary algorithms. It is described as follows:

$$U_{ij} = \begin{Bmatrix} V_{ij}\ ;\ \text{if rand}\ (0,1)\ \leq\ CR\ \vee \text{j=jrand} \\ X_{ij}\ ;\ \text{otherwise} \end{Bmatrix} \tag{4}$$

Here, jrand $\in$ [1, D]. The CR is crossover probability. It is defined as $CR \in [0,1]$. This crossover suggests that in trial vector, there must be at least one component from the donor vector.

Selection: Selection is the exploitation process. It is a greedy method. Selection is also termed as mother-child competition. As its name indicates, mother X_i and child U_i compete with each other for survival chance in the next generation. The mother-child competition is done as follows:

$$X_i = \begin{Bmatrix} U_i\ ;\ \text{if}\ f\left(U_i\right) \leq f\left(X_i\right) \\ X_i\ ;\text{otherwise} \end{Bmatrix} \tag{5}$$

4 Levy Distribution

Levy distribution **[12]** is a stable continuous probability distribution for non-negative random variable. It is stable because it has the property that a linear combination of independent copies of the variable has the same distribution, up to location and scale parameters. The Levy probability distribution function is

$$f(x;\alpha,\beta,c,\mu) = \sqrt{\frac{c}{2\pi}} \left(e^{-\frac{c}{2(x-\mu)}} \Big/ (x-\mu)^{\frac{3}{2}} \right) \tag{6}$$

Here, α is the characteristic exponent; β is the skewness parameter which represents the measure of asymmetry; c is the scale parameter which measures the width of the distribution; μ is the shift or location parameter. In Levy distribution, the value of the parameters are : α=0.5 ; β=1; c>0; $\mu \geq 0$; Like all stable distributions except the normal distribution, the wing of the probability density function exhibits heavy tail behavior falling off according to a power law :

$$\lim_{x\to\infty} f\left(x;0.5,1,c,\mu\right)=\sqrt{\frac{c}{2\pi}}\left(1\big/x^{\frac{3}{2}}\right) \tag{7}$$

5 LevyDE: Levy Distributed Differential Evolution

This paper introduces a new self-adaptive strategy for control parameters in the DE algorithm. Form the study of **[13]**, it has been observed that heavy tail distribution works better to probabilistically control the parameters. To create a considerable amount of diversity in the solution, a new heavy tail distribution is used in the scale factor and crossover rate. Therefore, these values will be altered according to the following Levy distribution described in equation **(6)**. The reason of choosing levy distribution is that the heavy tail will generate considerable changes more frequently and it has higher probability to do a long jump that may escape from local optima or move away from a plateau that might provide a better solution for multimodal optimization problem.

In LevyDE, scale factor, $F \in [0.5\ ,\ 0.9]$ and crossover rate, $CR \in [0.5,1]$. This F and CR are updated according to the Levy distribution. Here, two new parameters are introduced delF and delCR. These parameters are used to control the updation of the shift parameters (i.e. scaleF and scaleCR) of the Levy distribution to get updated control parameters. The parameters scaleF, scaleCR $\in$ [0,1], are initialized randomly. The LevyDE algorithm is as follows:

Procedure LevyDE ()

```
1.  Initialize the population matrix X[N, D] randomly in the search
    space by equation 1 and other parameters.
2.  fitness := calculateFitness(X)
3.  while termination condition not satisfied
    3.1 if delF < U(0, 1)
            3.1.1  scaleF := scaleF + ($scaleF^u$ - $scaleF^l$)*U(0, 1)
            3.1.2  F := F + ($F_{max}$ - $F_{min}$) *
                   Levy(iteration,0.5,1,1,scaleF)
    3.2 else
            3.2.1  F := F + ($F_{max}$ - $F_{min}$) * Levy
                   (iteration,0.5,1,1,scaleF)
    3.3 if F > $F_{max}$
            3.3.1  F := $F_{min}$ + 0.4*U(0, 1)
    3.4 if delCR < U(0, 1)
            3.4.1  scaleCR := scaleCR + ($scaleCR^u$ - $scaleCR^l$)*U(0, 1)
            3.4.2  CR := CR + $CR_{max}$* Levy (iteration,0.5,1,1,scaleCR)
    3.5 else
            3.5.1  CR := CR + ($CR_{max}$ - $CR_{min}$) *
                   Levy(iteration,0.5,1,1,scaleCR)
```

```
3.6 if CR > $CR_{max}$
        3.6.1  CR := $CR_{max}$*U(0, 1)
3.7 U := mutation(X, type)
3.8 V := crossover(X, U, type)
3.9 fitness_trial := calculateFitness(V)
3.10    for i := 1 to N in the step of 1
        3.10.1 if $fitness\_trial_i \le fitness_i$
                3.10.1.1       $X_i := V_i$
                3.10.1.2  $fitness_i := fitness\_trial_i$
```

6 Experiments and Analysis of the Results

To test the algorithm, we have selected CEC 2005 benchmark problems **[2]** and considered 10 different functions, described in Table 1. The function 1- 4 is unimodal and rest is multimodal functions. The F10 is an expanded multimodal benchmark function.

Table 1. Benchmark functions selected from CEC 2005 function sets

Function Definition	Dimension	Range	Optimum
$f_1(X)=\sum_{i=1}^{D} X_i^2+f_{bias}$; $X=X-o$;	10	[-100,100]	- 450
$f_2(X)=\sum_{i=1}^{D}\left(\sum_{j=1}^{i} X_j\right)^2+f_{bias}$; $X=X-o$	10	[-100,100]	- 450
$f_3(X)=(A_iX-B_i)+f_{bias}$; $\det(A)\neq 0, a_{ij}\in[-500,500], B_i=A_i*o, o_i\in[-100,100]$	10	[-100,100]	- 310
$f_4(X)=\sum_{i=1}^{D}\left(100\left(X_i^2-X_{i+1}\right)^2+(X_i-1)^2\right)+f_{bias}$; $X=(X-o+1)$	10	[-100,100]	390
$f_5(X)=-20e^{\left(0.2\sqrt{\left(\sum_{i=1}^{D} X_i^2/D\right)}\right)}-e^{\left(\sum_{i=1}^{D}\cos(2\pi X_i)/D\right)}+20+e+f_{bias}$; $X=(X-o)*M$	10	[-32,32]	-140
$f_6(X)=\sum_{i=1}^{D}\left(X_i^2-10\cos(2\pi X_i)+10\right)+f_{bias}$; $X=(X-o)$	10	[-5,5]	-330
$f_7(X)=\sum_{i=1}^{D}\left(X_i^2-10\cos(2\pi X_i)+10\right)+f_{bias}$; $X=(X-o)*M$	10	[-5,5]	-330
$f_8(X)=\sum_{i=1}^{D}\left(\sum_{k=o}^{k_{max}}\left[a_k\cos(2\pi b_k(X_i+0.5))\right]\right)-D\sum\left[a_k\cos(2\pi b_k\times 0.5)\right]+f_{bias}$; $a=0.5, b=3, k_{max}=20$; $X=(X-o)*M$	10	[-0.5,0.5]	90
$f_9(X)=\sum_{i=1}^{D-1}F_G(F_R(X_i,X_{i+1}))+F_G(F_R(X_D,X_1))+f_{bias}$; $F_G=\sum_{i=1}^{D}X_i^2/4000-\prod_{i=1}^{D-1}\cos\left(X_i/\sqrt{i}\right)+1$, $F_G=\sum_{i=1}^{D}\left(100\left(X_i^2-X_{i+1}\right)^2+(X_i-1)^2\right)$;$X=X-o+1$	10	[-π, π]	-460
$f_{10}(X)=\sum_{i=1}^{D-1}G(X_i,X_{i+1})+G(X_D,X_1)+f_{bias}$;$G(X,Y)=0.5+\left(\sin^2\sqrt{(X^2+Y^2)}-0.5\right)/\left(1+0.001(X^2+Y^2)\right)^2$;$X=(X-o)*M$	10	[-100,100]	-300

The algorithms are tested on Intel® Pentium(R) CPU B960 @ 2.20GHz × 2 processor with 4 GB DDR3 RAM. The operating system platform is Ubuntu 12.04, 32 bit. The programming language is MATLAB R2012a.

For basic DE, F=0.5 and CR=0.50. For LevyDE, F ∊ [0.5, 0.9], CR ∊ [0.5, 1.0], delF= 0.50 and delCR=0.70. The maximum iteration is fixed at D*1e+04 where D is the dimension of the problem. Here D is 10 and population size is 20. In our experiment, four algorithms i.e. LevyDE, DE/best/1/bin, DE/rand/1/bin and Adaptive Differential Evolution algorithm (ACDE) **[8]** are run for 25 times for each of the functions. The results are described in Table 2. The Best Result, Mean Error, Standard Deviation and Success Rate are considered for comparison of different algorithms. We have considered error value 1e-06 for function 1-3, 1e-02 for others.

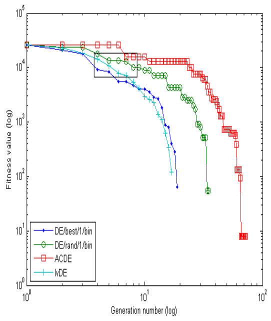

Fig. 1. Convergence plot of 'F1'

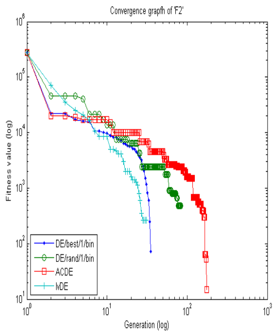

Fig. 2. Convergence plot of 'F2'

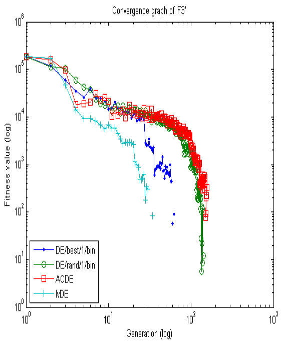

Fig. 3. Convergence plot of 'F3'

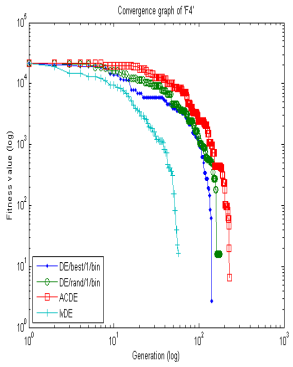

Fig. 4. Convergence plot of 'F4'

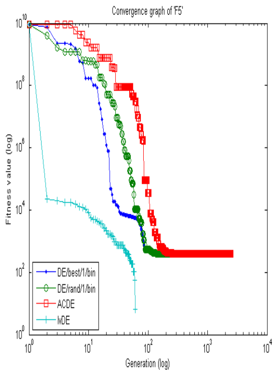

Fig. 5. Convergence plot of 'F5'

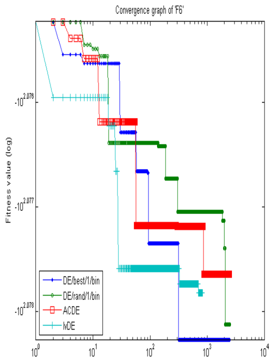

Fig. 6. Convergence plot of 'F6'

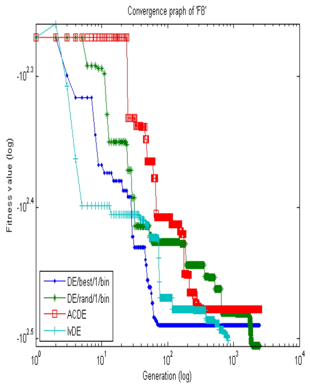

Fig. 7. Convergence plot of 'F8'

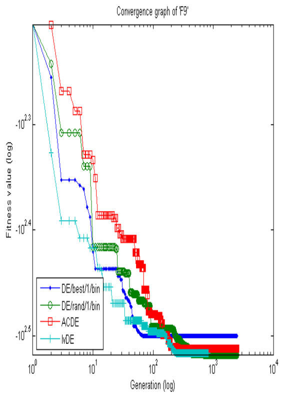

Fig. 8. Convergence plot of 'F9'

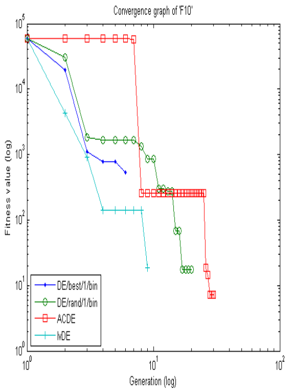

Fig. 1. Convergence plot of 'F10'

Table 2. Comparison analysis of the best result, mean error, standard deviation and success rate of DE/best/1/bin, DE/rand/1/bin, ACDE, and LevyDE

Functions	Procedures	Best result	Mean Error	Std Deviation	Success rate (%)
F1	DE/best	-4.500000e+02	2.042965e-07	1.670335e-07	100
	DE/rand	-4.500000e+02	1.561729e-07	1.151371e-07	100
	ACDE	-4.500000e+02	4.815826e-08	1.358376e-07	100
	LevyDE	**-4.500000e+02**	**2.004074e-09**	**1.065551e-07**	**100**
F2	DE/best	-4.500000e+02	2.695379e-07	2.090216e-07	100
	DE/rand	-4.500000e+02	1.717866e-02	8.583652e-02	084
	ACDE	-4.499767e+02	2.325408e-02	1.867916e-07	000
	LevyDE	**-4.500000e+02**	**1.732750e-07**	**1.260445e-07**	**100**
F3	DE/best	-4.500000e+02	2.611088e-07	1.922718e-07	100
	DE/rand	-4.500000e+02	1.104072e-07	1.096909e-07	100
	ACDE	-4.499985e+02	1.338269e+00	2.899092e+00	000
	LevyDE	**-4.500000e+02**	**1.002874e-07**	**1.809248e-07**	**100**
F4	DE/best	-3.100000e+02	**8.984726e-08**	**5.933756e-08**	100
	DE/rand	-3.100000e+02	9.914377e-02	4.593829e-01	076
	ACDE	-3.061221e+02	3.877860e+00	1.668932e+00	000
	LevyDE	**-3.100000e+02**	7.7675e-07	1.944300e-07	**100**
F5	DE/best	3.900060e+02	4.583631e+00	1.356022e+01	024
	DE/rand	3.900059e+02	4.339623e+00	3.108901e-01	012
	ACDE	3.966768e+02	6.666787e+00	5.933756e+00	000
	LevyDE	**3.900011e+02**	**1.255594e+00**	**2.189511e+00**	**072**
F6	DE/best	-1.197357e+02	2.036411e+01	**6.547529e-02**	000
	DE/rand	-1.198571e+02	2.038093e+01	8.970581e-02	000
	ACDE	-1.198655e+02	2.038609e+01	9.796716e-02	000
	LevyDE	**-1.208680e+02**	**2.036857e+01**	9.175338e-02	000
F7	DE/best	-3.270151e+02	8.745213e+00	5.989216e+00	000
	DE/rand	**-3.300943e+02**	**2.584488e+00**	**3.332907e+00**	**036**
	ACDE	-3.290050e+02	8.996444e+00	5.631635e+00	000
	LevyDE	-3.290050e+02	1.643564e+01	3.854307e+00	000
F8	DE/best	-3.270151e+02	**1.460600e+01**	7.608035e+00	000
	DE/rand	-3.260202e+02	1.701437e+01	**7.496491e+00**	000
	ACDE	-3.190555e+02	2.452018e+01	1.306585e+01	000

Table 2.(*Continued*)

	LevyDE	**-3.290050e+02**	1.643564e+01	1.078447e+01	000
F9	DE/best	1.120927e+02	2.208274e+01	1.450389e-14	000
	DE/rand	1.120927e+02	2.208274e+01	1.450389e-14	000
	ACDE	1.120927e+02	2.208274e+01	1.450389e-14	000
	LevyDE	**1.012927e+02**	**2.208274e+01**	**1.450389e-14**	000
F10	DE/best	-1.299605e+02	8.421141e-01	9.943467e-01	000
	DE/rand	-1.295507e+02	8.334355e-01	**1.997779e-01**	000
	ACDE	-1.299913e+02	**5.945683e-01**	1.792737e+00	000
	LevyDE	**-1.299208e+02**	8.119437e-01	3.954782e-01	000

From the results of Table 2, it is prominent that the LevyDE outperforms the rest 3 algorithms for most of these benchmark functions in all four criteria. The following Figure 1 – 9 shows the convergence rate plot of the four algorithms for the above mentioned function values with respect to generations in the log scale. These figures also show that the convergence rate of LevyDE is better than the rest three algorithms.

7 Conclusion

In this paper, we proposed a modified version of basic Differential Evolution (DE) using Levy distribution for parameter adaption. Here, we have applied 'DE/best/1' mutation strategy and binary crossover strategy with parameter adaption by Levy distribution to improve the exploration and exploitation mechanism of mutation and selection procedure. This mutation strategy improves the problem of premature convergence and trapping in local optimal. Experimental results on standard benchmark functions prove the better efficiency of LevyDE algorithm. Future work is to evaluate the performance of LevyDE using more test functions and make it more robust by upgrading and balancing the exploration and exploitation strategies.

References

[1] Price, K., Storn, R., Lampinen, J.: Differential Evolution - A Practical Approach to Global optimization. Springer (2005)

[2] Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.-P., Auger, A., Tiwari, S.: Problem definitions and evaluation criteria for the CEC05 special session on real-parameter optimization. Technical Report, Nanyang Teechnological University, Singapore (May 2005)

[3] Epitropakis, M.G., Plagianakos, V.P., Vrahatis, M.N.: Balancing the exploration and exploitation capabilities of the Differential Evolution algorithm. In: Proceedings of 2008 IEEE Congress on Evolutionary Computation (CEC 2008), pp. 2686–2693 (2008)

[4] Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter Control in Evolutionary Algorithms. IEEE Transaction on Evolutionary Computation 3(2), 124–141 (1999)
[5] Krink, T., Filipič, B., Fogel, G.B.: Noisy optimization problems: A particular challenge for differential evolution. In: Proc. IEEE Congr. Evol. Comput., pp. 332–339 (2004)
[6] Liu, B., Zhang, X., Ma, H.: Hybrid differential evolution for noisy optimization. In: Proc. IEEE Congr. Evol. Comput., pp. 587–592 (June 2008)
[7] Liu, J., Lampinen, J.: A fuzzy adaptive differential evolution algorithm. Soft Comput. 2005 9(6), 448–462 (2005)
[8] Thangaraj, R., Pant, M., Abraham, A.: A Simple Adaptive Differential Evolution Algorithm. In: Proc. NaBIC 2009, pp. 457–462 (2009)
[9] Teo, J.: Exploring Dynamic Self-adaptive Populations in Differential Evolution. Soft Computing - A Fusion of Foundations' Methodologies and Applications 10(8), 673–686 (2006)
[10] Yang, Z., Yao, K.T.X.: Self-adaptive Differential Evolution with Neighborhood Search. In: IEEE World Congress on Computational Intelligence, pp. 1110–1116 (2008)
[11] Qin, A.K., Suganthan, P.N.: Self-adaptive Differential Evolution Algorithm for Numerical Optimization. In: Proc. IEEE Congress on Evolutionary Computation (September 2005)
[12] Applebaum, D.: Lectures on Lévy processes and Stochastic calculus, Braunschweig; Lecture 2: Lévy processes, pp. 37–53. University of Sheffield
[13] Pant, M., Thangaraj, R., Abraham, A., Grosan, C.: Differential Evolution with Laplace Mutation Operator. In: CEC 2009, pp. 2841–2849 (2009)
[14] Abbass, H.A.: The self-adaptive pareto differential evolution algorithm. In: Proc. of 2002 Congress on Evolutionary Computation, pp. 831–836 (2002)