Journal of Nature Inspired Computing (JNIC)
Vol. 1, No. 1, 2012

www.worldsciencepublisher.org

Modified Mutation in Differential Evolution Algorithm to Optimize Supply Chain System

[1]Tarun Kumar Sharma, [2]Millie Pant, [3]V.P.Singh

[1, 2] Indian Institute of Technology Roorkee, Roorkee, India
[3] Director General, SCET, Saharanpur, India

Email:{taruniitr1; millidma; singhvp3}@gmail.com

Abstract – Differential Evolution (DE) is a simple, efficient algorithm which has reportedly outperformed many other optimization algorithms in terms of convergence speed and robustness over common benchmark problems and real world applications. However, one is required to set the values of the control parameters of DE for each problem. Such parameter tuning is a time consuming task. The proposed scheme dynamically adapts the mutation step size for better exploration and exploitation of the search space. In this paper we propose SaMSDE algorithm that incorporates exponential distributions to produce mutation steps with varying lengths and suitably adjusts the current step length. To show the performance of our proposed SaMSDE, experiments are carried out on a set of seven well-known benchmark problems. Further the proposed SaMSDE is applied to solve supply chain system. Simulation results show that the proposed algorithm can effectively enhance the searching efficiency and greatly improve the searching quality.

Keywords – Self-Adaptation; Mutation; Differential Evolution; Global Optimization, Supply Chain System

1. Introduction

Global optimization problem is not easy to solve and standard deterministic algorithms tend to stop the search in local minimum nearest to the input starting point. Therefore, heuristic search techniques are widely used by the optimization community when obtaining the global optimum is not only desirable but is also necessary. Such heuristics are often population based search techniques, inspired by some natural process like theory of evolution (Genetic Algorithms, Differential Evolution etc.) or behavior of species (Particle Swarm Optimization, Ant Colony Optimization etc).

Evolutionary algorithms such as Genetic Algorithms (GA) [1] and Differential Evolution (DE) [2][3] are able to find the acceptable solution within a reasonable time limit, but the success of these algorithms depends, to a large extent on the careful tuning of control parameters. In case of DE which is the focus of the present work, the two important parameters are Scaling Factor (F) and Crossover Rate (CR).

Mutation operation plays the most significant role in the performance of a DE algorithm.

Despite having several attractive features, it has been observed that DE sometimes does not perform as per the expectations. Empirical analysis of DE has shown that it may stop proceeding towards a global optimum even though the population has not converged even to a local optimum [5]. The situation when the algorithm does not show any improvement though it accepts new individuals in the population is known as stagnation. Besides this, DE also suffers from the problem of premature convergence. This situation arises when there is a loss of diversity in the population. It generally arises when the objective function is multi objective having several local and global optimums. Like other EA, the performance of DE deteriorates with the increase in dimensionality of the objective function. Several modifications have been made in the structure of DE to improve its performance. One class of modification deals with the development of adaptive control parameters [9] – [15]. Use of self adaptive parameters saves the user from the trouble of fine tuning of parameters,

Many of the developments in DE algorithm design and applications can be found in [15].

In this paper we introduce a new Differential Evolution algorithm SaMSDE, which incorporates exponential distribution to produce mutation steps with varying lengths and suitably adjusts the current step length.

Performance of SaMSDE is analyzed on a set of seven standard unconstrained benchmark functions and a supply chain model.

Optimization of a supply chain model is an integer programming problem or a constrained integer-mixed problem [12]. Several methodologies for optimizing a supply chain have been proposed in the literature so far. Regarding evolutionary algorithms, genetic algorithms are the most popular for supply chain optimization problems [13-18].

The remaining paper is organized as follows: Section II gives a very brief survey of adaptive control parameter

algorithms in DE, an overview of DE is given in the Section III, Motivation for the given work and proposed algorithms are given in sections IV and V respectively. Section VI describes supply chain system in brief. Section VII describes parameter settings, performance metrices and simulation results. Finally the paper concludes with section VIII.

2. Related Work

There are quite different conclusions about the rules for choosing the control parameters of DE. Price and Storn [1] stated that the control parameters of DE are not difficult to choose. On the other hand, Gämperle et al. [6] reported that choosing the proper control parameters for DE is comparatively difficult than expected. Liu and Lampinen [7] reported that effectiveness, efficiency, and robustness of the DE algorithm are sensitive to the settings of the control parameters. The best settings for the control parameters can be different for different functions and the same function with different requirements for consumption time and accuracy. However, there still exists a lack of perfect knowledge on how to find reasonably good values for the control parameters of DE for a given function [8].

Several instances are available in literature for determining the optimal values of these control parameters. But it is observed that mostly the control settings are problem specific and different scholars have different views regarding the optimal values of these parameters [2]-[6]. Researchers therefore laid emphasis on having adaptive/self adaptive control parameters to avoid the careful and time consuming process of fine tuning. Ali and Torn [7] proposed a simple rule for adapting the F scaling factor value during the search process. Quin and Suganthan [9] proposed self-adaptive choice of mutation strategy combined with controlled random adjusting the values of F and CR. Evolutionary self-adaptation of control parameters F and CR suggested by Brest et al. [10][11] have proved good convergence in the applications. Several other modifications have been suggested in the literature [12]-[14].

3. Overview of DE

DE is an Evolutionary Algorithm (EA) proposed by Storn and Price in 1995 [2]. DE starts by randomly generating an initial population, when no preliminary knowledge about the solution space is available. The uniformly distributed random numbers are evaluated using the fitness function provided. Then the following are executed until maximum number of generation has been reached or an optimum solution is found.

Mutation Operation

For each target vector $x_{i,G}$ at generation G , an associated mutated vector is usually generated as follows:

$$v_{i,G+1} = x_{r1,G} + F(x_{r2,G} - x_{r3,G}) \quad (1)$$

where r_1 , r_2 , $r_3 \in$ (1,2…,NP) are randomly chosen integers, different from each other and also different from the running index i. F is a real and constant factor having value between [0, 2] and controls the amplification of differential variation $(x_{r2,G} - x_{r3,G})$.

Crossover Operation

After the mutation phase, the "*binominal*" crossover operation is introduced in order to increase the diversity of the perturbed parameter vectors. The parent vector is mixed with the mutated vector to produce a trial vector $u_{ji,G+1}$,

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1} & if\ rand_j(0,1) \le Cr \,\forall\, j = k \\ x_{ji,G} & otherwise \end{cases} \quad (2)$$

where j, k = {1, 2,…, D (dimension)}; k is the random parameter index, chosen once for each i, $rand_j \in [0, 1]$ and Cr is a user-specified crossover constant in the range [0, 1].

Selection Operation

Finally selection takes place where a tournament is held between the target vector and trial vector and the one with better fitness function is allowed to enter the next generation. In this way individuals in a new generation are as good as or better than the individuals in the previous generation. DE contains 4 parents, in which one is target vector; x_i and three are random vectors x_{r1}, x_{r2} and x_{r3}. The detailed pseudocode is given in Figure 1 and process is shown in Figure 2.

1: Generate uniformly distributed random population, G (generation) = 1
2: **while** termination criterion not met do
3: **for** $i = 1; i \le NP$ (Population Size); $i = i + 1$
Parents Selection
(For Original DE, for each vector x_i, select randomly three distinct vectors x_{r1}, x_{r2} and x_{r3} from the current population other than the vector x_i)
4: Randomly select r_1 , r_2 , $r_3 \in \{1,2,...,NP\}$, $r_1 \neq r_2 \neq r_3 \neq i$
5: **for** $j = 1; j \le D$ (Dimension); $j = j + 1$ do
6: Randomly select $j_{rand} \in \{1, 2... D\}$
****
Binomial Crossover for DE (DE/1/rand/bin)
7: **if** (rand () < CR or $j = j_{rand}$)
8: $u_{ji,G} = x_{r1,G} + F * (x_{r2,G} - x_{r3,G})$

9: **else**
10: $u_{ji,G} = x_{ji,G}$
11: **end if**
12: **end for**
13: **end for**
14: **for** $i = 1; i \leq NP; i = i + 1$ *do*
****
Selection
15: **if** $f(u_{ji,G}) \leq f(x_{i,G})$
16: $x_{i,G+1} = u_{ji,,G}$
17: **else**
18: $x_{i,G+1} = x_{i,,G}$
19: **end if**
****
20: **end for**
21: G = G + 1
22:**end while**

Figure 1. Pseudocode of basic DE

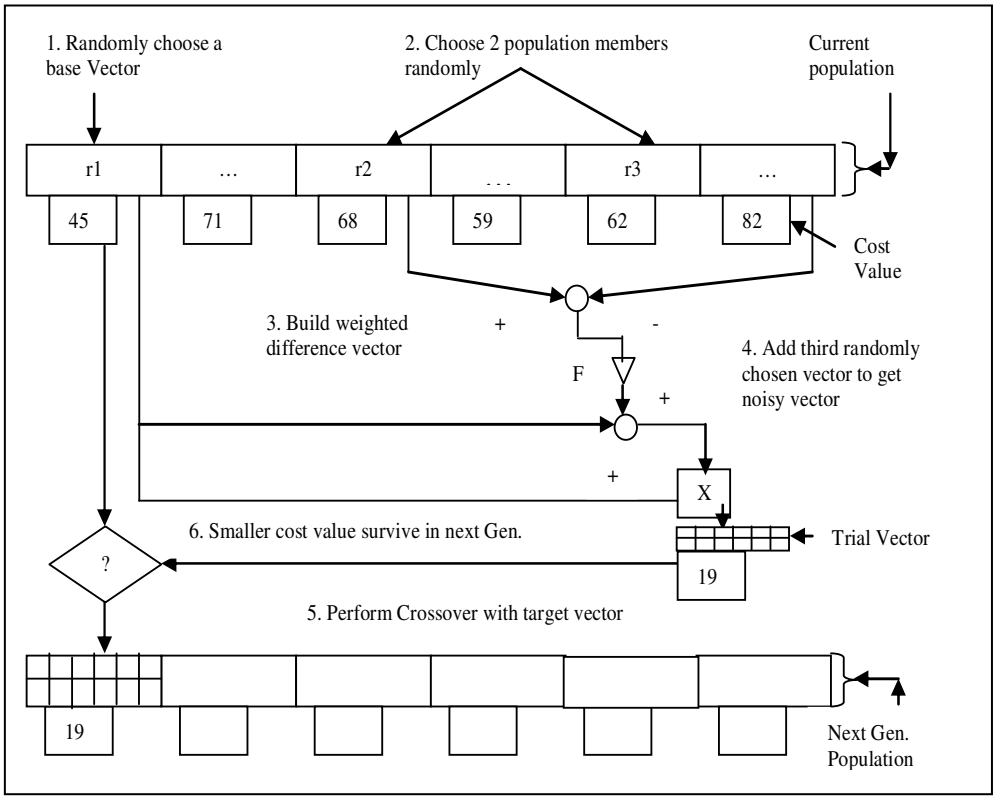


Figure 2. Process of DE

4. Motivation

The basic motivation behind the proposed algorithm SaMSDE is to balance the exploration and exploitation in basic DE. So, the proposed algorithm incorporates exponential distributions to produce mutation steps with varying lengths and suitably adjusts the current step length. Mutation with large step size is likely to produce large variations which would facilitate better exploration of the undiscovered regions of the search space while small step size usually produces small variations that are better for exploitation of the already found solutions. The appropriateness of small or large steps changes dynamically depending on the current stage and maturity of the ongoing search process as well as the properties of the search space. So, dynamic adaptation of mutation step size is quite promising and efficient to solve continuous optimization problems.

5. Self Adaptive Mutation Step size

The only structural difference between the proposed SaMSDE algorithm and the basic DE is during mutation. The scaling factor (F) in eq. (1) is replaced by $MS_{ij} * r_{ij}$.

$$v_{i,G+1} = x_{r1,G} + MS_{ij} * r_{ij}(x_{r2,G} - x_{r3,G}) \quad (3)$$

MS_{ij} is the scaling factor that keeps the knowledge of separately for every dimension, j of every individual x_i, r_{ij} randomly range in [0, 1].
MS_{ij} is initialized to 1 during the beginning of the search process. As the search progresses across several local

minima and plateaus or flat regions, the MS_{ij} values are automatically adjusted by the adaptation scheme in order to take care of the current situation. Large values for MS_{ij} would expand the product $MS_{ij} * r_{ij}$ in order to promote large mutation steps for better exploration of the search space and quickly get rid of local minima. On the contrary, small values (less than unity) for MS_{ij} would shrink the product $MS_{ij} * r_{ij}$ (in (3)) and thus facilitate small mutation steps ensuring exploitation in the vicinity of current search points. Whether exploitation or exploration would be better at current search stage might not be apparent or could not be predicted beforehand. So SaMSDE employs two different exponential distributions in order to produce different range of scaling factor (i.e. MS_{ij}) values for explorations and exploitations. The adjusted scaling factors (i.e., MS_{ij} values) are the weighted average of the current scaling factor values and the random values generated anew from the two exponential distributions. The procedure is further explained in the following pseudocode (Figure.4) along with the two distributions used for explorations and exploitations. To adapt the MS_{ij} values, SaMSDE generates two gaussian random values, r[1] and r[2] from the ranges (0, -10] and (0, 10]. Now, for every individual solution two different offspring solutions are generated by (3): by using $MS_{ij,1} = 2^{r[1]}$ and $MS_{ij,2} = 2^{r[2]}$. Since r[1] < 0 and r[2]> 0, the scaling factor $MS_{ij,1} = 2^{r[1]}$ would generate small steps for better exploitation, while $MS_{ij,2} = 2^{r[2]}$ would produce large steps for more search space exploration. SaMSDE evaluates the fitness of each offspring and accepts the better one. Also, it moves MS_{ij} from its current value to the new scaling factor ($MS_{ij,1}$ or $MS_{ij,2}$) based on which one produces better offspring using the formula:

$$SF_{ij} = r_1 * MS_{ij} + (1 - r_1) * MS_{ij,K} \quad (4)$$

where r1, r2∈ rand[-0.5, 0.5]. These two distributions produce the scaling factor values for exploration and exploitation respectively. Exponential distributions provide wide range of values as scaling factors facilitating very small to very large jumps ensuring from little to large explorations and exploitations.

1. *Follow Steps* 1 *to* 4 *from Figure* 1.
2. ***for*** each $i \in NP$
3. ***for*** j = 1 to D
{
4. ***for*** k = 1 to 2
{
5. $r[k] = \text{rand}[0, \tau_k]$
6. $MS_{ij,k} = 2^{r[k]}$
7. $v_{ij,k} = v_{ij}$ computed from (3) by using $MS_{ij} = MS_{ij,k}$
8. $temp\_v_i = \boldsymbol{v_i}$, but v_{ij} replaced by $v_{ij,k}$
9. $f[k] = f(temp\_v_i)$
}
10. find k such that $f[k]$ is the minimum over $f[temp\_v_i]$
11. *Calculate* MS_{ij} *using eq.* (4)
}
12. *Follow steps* 5 *to* 22*, but replace step* 8 *by eq.* (3)

Figure 3. Pseudocode for the adaptation of mutation step size in DE

6. Self Adaptive Mutation Step size

Experimental Setting

In this section, the proposed SaMSDE algorithm is validated on a set of seven benchmarks taken from [17]. In order to investigate the performance of the proposed SaMSDE we compared it with the DE algorithm and PSO. DE, proposed SaMSDE are implemented in Dev-C++ and the experiments are conducted on a computer with 2.00 GHz Intel (R) core (TM) 2 duo CPU and 2- GB of RAM. In order to make a fair comparison of DE and the proposed algorithm, we fixed the same seed for random number generation so that the initial population is same for both the algorithms. For each problem, the SaMSDE is independently run 30 times. The parameter setting is taken as follows:

Mutation strategy	*DE/rand/1/bin*
Population size NP	*50*
Scaling factor (F)	*0.5*
Crossover probability	*0.9*
Iterations	*5000*
Value to Reach (VTR)	10^{-06}
Maximum NFE	*300000*

Performance Criteria

Four performance criteria are selected to evaluate the performance of the algorithms. These criteria are described as follows:

Mean Fitness and Standard Deviation

The average of function fitness value that an algorithm can find, using predefined maximum NFEs, is recorded in each run and then average of the function fitness values are calculated. Also the average and standard deviation of the fitness values are calculated.

NFEs [18]

The number of fitness function evaluations (NFEs) is recorded when the VTR is reached before to reach maximum NFE. i.e we set the termination criteria as $\left|f_{optimal} - f_{global}\right| \le VTR$ and record average NFE of successful run over 30 runs.

Convergence Graphs [18]

The convergence graphs show the mean fitness performance of the total runs, in the respective experiments.

Acceleration rate (AR) in % [19]

This criterion is used to compare the convergence speeds between SaMSDE and DE. It is defined as follows:

$$AR = \frac{NFE_{one\ algorithm} - NFE_{other\ algorithm}}{NFE_{one\ algorithm}}\% \quad (5)$$

Simulated Results

The simulated results based on the above experimental setting are given in Table-1, Table-2 and Table-3. In Table-1 we have taken the results on the basis of average error. In this case NFEs are fixed at 105 to estimate the average of minimum fitness function value in 30 runs. From the Table-1 it can be clearly observed that for all benchmark functions SaMSDE gives better results than PSO and DE. A two tail sample t-test [20]-[21] is also applied to analyze the statistical significance of the proposed algorithm. We have checked the significant difference of SaMSDE with respect to DE at 5% level of significance. For statistical analysis, we consider the null hypothesis and alternative hypothesis as:

H_0: There is no significant difference between the mean fitness value of SaMSDE and others algorithms.

H_1: Mean fitness value of SaMSDE is better than others algorithms.

The calculated t-value of all function is greater than t-table value that shows the significant better performance in the comparison of DE.

In the Table-2, we fixed VTR as given in experimental setting and then calculated the average NFE of 30 runs. From Table-2 we can see that the proposed SaMSDE gives the better results for every function in the comparison to the other algorithms. From the Table-2 it is clear that the proposed SaMSDE is faster than PSO & DE by 39.96% & 29.33% respectively. Best and worst function values in 30 runs are shown in Table-3.

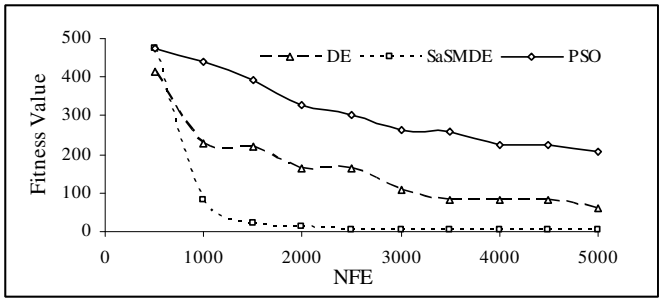
Figure 4(a)-(b) shows the convergence graph of SaMSDE with the comparisons of DE and PSO. NFEs taken to estimate the average of minimum fitness function value in 30 runs are also presented graphically in Figure 5.

Table 1. Comparison of SaMSDE with PSO, DE algorithm in terms of NFEs and AR(%), Here 3/1 implies SaMSDE vs PSO, and 3/2 implies SaMSDE vs DE

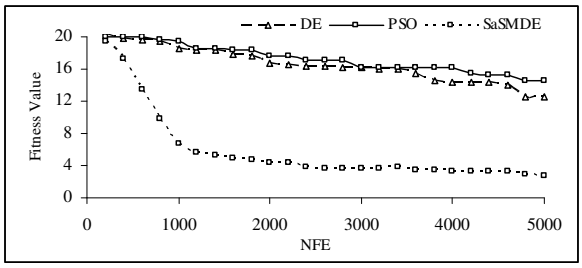
F	PSO	DE	SaMSDE	AR (3/1)	AR (3/2)
F_1	43600	26380	16360	62.477064	37.983321
F_2	97630	74370	62890	15.097818	15.436332
F_3	468200	330840	160600	65.698419	51.456898
F_4	61880	76680	54362.5	12.148513	29.104721
F_5	29110	19500	17880	38.577808	8.3076923
F_6	53220	52590	37190	30.120256	29.283134
F_7	50040	33540	22220	55.595524	33.750745
Average AR(%)				**39.96**	**29.33**

Table 2. Best and (Worst) Function values in 30 runs

F	PSO	DE	SaMSDE
F_1	6.04453e-005 0.000100962	8.69549e-005 0.000100901	8.33852e-005 0.00010096
F_2	5.56861e-006 9.98884e-006	6.06769e-005 9.44291e-005	4.78849e-006 9.93995e-006
F_3	0.00993023 0.0100924	0.000685359 0.00099867	0.086481 0.0987919
F_4	7.86592e-005 9.89971e-005	4.68213e-006 9.74202e-006	5.57054e-006 7.79099e-006
F_5	0 0	0 0	0 0
F_6	8.7398e-005 0.000100506	8.79372e-005 9.95211e-005	8.80526e-005 0.000100549
F_7	9.19165e-005 0.000100997	9.12616e-006 0.000100094	2.37059e-005 9.89772e-005



(a)



(b)

Figure 1. Convergence graph of Function (a) F_4 and (b) F_6

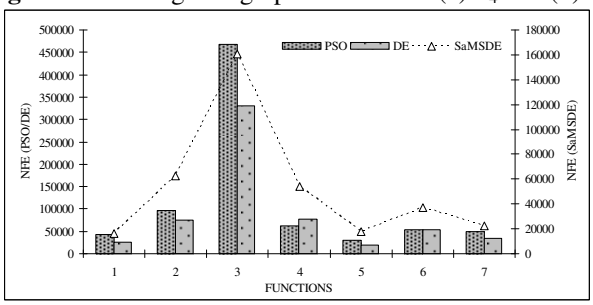


Figure 2. NFEs taken to estimate the average of minimum fitness function value in 30 runs

6. Conclusions

In the present study we propose a scheme that dynamically adapts the mutation step size for better exploration and exploitation of the search space. The self adaptive mutation step size helps in increasing the diversity which in turn helps in balancing exploration and exploitation of the search space which finally helps in improving the solution quality and the convergence rate of an algorithm. This is evident from the empirical studies done in the present study. We are continuing our work towards the theoretical development of the proposed algorithms and extending them for solving constrained and real optimization problems..

Appendix A

Function	Function Definition	D	Range	Optimum
Sphere	$f_1(x) = 10n + \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i))$	30	[-5.12, 5.12]	0
Rastringin	$f_2(x) = \sum_{i=1}^{n} x_i^2$	30	[-5.12, 5.12]	0
Rosenbrock's	$f_3(x) = \sum_{i=1}^{n-1} [100\ (x_{i+1}^2 - x_i^2) + (1 - x_i^2)]$	10	[-30, 30]	0
Griekwank	$f_4(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$	30	[-600, 600]	0
Step	$f_5(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$	30	[-100, 100]	0
Ackly	$f_6 = -20 * \exp\left(-.2\sqrt{1/n\sum_{i=1}^{n} x_i^2}\right) - \exp\left(1/n\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	30	[-32, 32]	0
Colliville	$f_7(x) = 100(x_2 - x_1^2)^2 + (1-x_1)^2 + 90(x_4 - x_3^2)^2 + (1-x_3)^2 + 10.1(x_2-1)^2 + (x_4-1)^2 + 19.8(x_2-1)(x_4-1)$	4	[-10, 10]	0

References

[1] Goldberg D., Genetic Algorithms in Search Optimization and Machine Learning. Addison-Wesley, (1989).

[2] Storn R. and Price K. V., Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, J. GlobalOptimization, vol. 11, pp. 341–359, (1997).

[3] Price K. V., Storn R., and J. Lampinen, Differential Evolution: A Practical Approach to Global Optimization. Springer, (2005).

[4] ——, "Critical values for the control parameter of the differential evolution algorithms," in *MENDEL 2002, 8th International Conferenceon Soft Computing*, R. Matou˘sek and P. O˘smera, Eds. Brno: University of Technology, 2002, pp. 62–67.

[5] R. G¨amperle, S. D. M¨uller, and P. Koumoutsakos, "A parameter study for differential evolution," in *Advances in Intelligent Systems Fuzzy Systems,*

Evolutionary Computing, A. Grmela and N. E. Mastorakis, Eds. Athens: WSEAS Press, 2002, pp. 293–298.

[6] T. B¨ack and H. P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary Computation*, vol. 1, no. 1, pp. 1–23, 1993.

[7] M. M. Ali and A. T¨orn, "Population set based global optimization algorithms: Some modifications and numerical studies," *Computers and Operations Research*, vol. 31, pp. 1703–1725, 2004.

[8] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algortithm," *Soft Computing*, vol. 9, pp. 448–462, 2005.

[9] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, vol. 2, pp. 1785–1791, 2005.

[10] J. Brest, B. Bo˘skovi´c, S. Greiner, V. ˘Zumer, and M. Mau˘cec, "Performance comparison of self-adaptive and adaptive differential evolution algorithms," *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, vol. 11, no. 7, pp. 617–629, 2007.

[11] J. Brest, S. Greiner, B. Bo˘skovi˘c, M. Mernik, and V. ˘Zumer, "Selfadapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, pp. 646–657, 2006.

[12] Salman, A. Engelbrecht, and M. Omran, "Empirical analysis of self-adaptive differential evolution," *European Journal of Operational Research*, vol. 183, no. 2, pp. 785–804, 2007.

[13] M. Omran, A. Salman, and A. Engelbrecht, "Self-adaptive Differential Evolution," Proceedings of the 2005 International Conference on Computational Intelligence and Security, pp. 192–199, 2005.

[14] Teo, J., Exploring Dynamic Self-adaptive Populations in Differential Evolution, Soft Computing - A Fusion of Foundations, Methodologies and Applications, Vol. 10 (8), pp. 673 – 686, (2006).

[15] Chakraborty, U. K., Advances in Differential Evolution, (Ed.) Springer-Verlag, Heidelberg, (2008).

[16] Mohammad Shafiul Alam et. al, Self-adaptation of Mutation Step Size in Artificial Bee Colony Algorithm for Continuous Function Optimization, In Proc. 13th International Conference on Computer and Information Technology, pp. 69 - 74.

[17] Liyuan J., Wenyin G., HongbinW.: An Improved Self-adaptive Control Parameter of Differential Evolution for Global Optimization. ISICA 2009, CCIS 51, pp. 215–224, Springer-Verlag Berlin Heidelberg (2009).

[18] Suganthan, P., Hansen, N., Liang, J.: Problem Definitions and Evaluation Criteria for the CEC2005 Special Session on Real-Parameter Optimization. (2005).

[19] Rahnamayan, S., Tizhoosh, H., Salama, M.: Opposition-Based Differential Evolution. IEEE Transactions on Evolutionary Computation 12(1), 64–79 (2008).

[20] Zhu, R. ―Statistical Analysis Methods, China Forestry Publishing House, Beijing, China (1989).

[21] Zhang, M., Luo, W. and Wang, X. Differential Evolution with Dynamic Stochastic Selection for Constrained Optimization, Information Science: An International Journal, vol 178, pp 3043-3074 (2008).

Vitae

Tarun Kumar Sharma did his MCA in 2001, M.Tech (IT) in 2009 and presently pursuing Ph.D from Indian Institute of Technology (IIT) Roorkee, India. He has almost 9 years of teaching experience in Engineering College. His key areas are Evolutionary Computing; Software Engineering; Computer based Optimization Techniques; ERP. His research interest includes swarm intelligence algorithms and their applications in various complex engineering design problems. His publications are in Journals and International Conferences of repute. He volunteered in SocPros-2011, an First International Conference on Soft Computing for Problem solving. He is peer reviewer of many IEEE conferences and International Journals. He is student member of Machine Intelligence Research (MIR) Labs, WA, USA.

Millie Pant is working as an assistant professor in Department of Paper Technology, Indian Institute of Technology (IIT), Roorkee, India since 2007. Her research interest includes evolutionary and swarm intelligence algorithms and their applications in various complex engineering design problems. Her publications are in Journals and International Conferences of repute. She has published over 100 referred on evolutionary algorithms (GA, PSO, DE and ABC) and their applications in electrical design problems, image processing papers. She has been program committee member of over 10 International events and Program Committee Chair of SoCProS-211. She is Program Committee Chair of the 7th International Conference on Bio-Inspired Computing: Theories and Application (BIC-TA 2012) and SoCProS-2012 (International Conference on Soft Computing for Problem Solving).

V.P. Singh received the Bachelor's degree from Meerut University, India in the year 1970, Master's and

Ph. D degrees in Applied mathematics from the University of Roorkee (now, Indian Institute of Technology Roorkee), India, in 1972, and 1978 respectively. Prof. Singh is currently with SCET Saharanpur, as a Director General. Previously he was Professor in the Department of Paper Technology, IIT Roorkee. His special fields of interests include Applied and Industrial Mathematics, Mathematical Modelling of Pulp washing problems. He has number of publications in journal of repute and has been reviewer for number of Journals and Conferences.