

Probabilistic-Based Differential Evolution Algorithm

¹Morteza Alinia Ahandani, ²Nooreddin Jafari

^{1,2}Department of electrical engineering, Islamic Azad University, Langaroud branch, Langaroud, Iran
^{1,2}Young Researchers Club, Langaroud Branch, Islamic Azad University, Langaroud, Iran

Email: ¹ alinia@iaul.com
² jafari@iaul.com

Abstract – In this paper we propose a probabilistic-based differential evolution (DE) algorithm for solving optimization problems. One of main drawbacks of DE is its few search efforts which it carries out for exploring the search space. The proposed DE with a probability and after providing a better fitness, accepts the generated donor member in mutation stage as an offspring. So the proposed algorithm by a new offspring can obtain a better exploration of the search space without imposing any additional operator. This improved algorithm is applied on some continuous benchmark functions and is compared with some modern and recently proposed differential evolution and other evolutionary algorithms. The obtained results demonstrate that the proposed algorithm has a completely comparable performance than other versions of DE algorithms. In a later part of the comparative experiments, performance comparisons of the proposed algorithm with some other evolutionary algorithms (EAs) shows that it has a better performance than other compared EAs.

Keywords – Differential evolution; Optimization problems; Search space; Donor member

1. Introduction

The evolutionary algorithms (EAs) are all-purpose and approximate optimization methods employed to solve a wide variety of optimization problems. The EAs are applied on those of problems which there are not any analytical methods to solve them or analytical methods can not obtain the solution in a rational time. Differential evolution (DE) [1] is a simple and population-based EA on stochastic search and direct optimization derived from genetic algorithms (GAs). Its name was derived from natural biological evolutionary processes.

Simple operator to generate new offspring, easy implementation and fast convergence are three main traits of DE. These traits cause widespread applications of the DE in the fields of optimization problems such as large scale passive harmonic filters planning problem [2], large-scale economic dispatch problem [3], cancer diagnosis [4], scheduling problems [5], optimization of supply chain systems [6], and electrical engineering [7], state feedback design for aircraft landing system [8], engineering design optimization [9], analyzing constrained machining conditions [10], image thresholding converge [11]. Plagianakos et al. [12] proposed a review of major application areas of DE algorithm.

In spite of its traits, the DE has some drawbacks. This algorithm has a problem to find high accurate optimal solution. The DE similar to most other direct search methods uses greedy criterion to accept or reject a new generated offspring. In the DE, only an offspring with a

better fitness than its parents is accepted and those of candidates with a worse fitness are rejected. However greedy criterion ensures the fast convergence, it increases the probability of trapping in local minimums [13]. Also the DE has a limited amount of search moves carried out to explore whole search spaces.

To compensate these drawbacks of the DE, a wide variety of approaches have been employed in the literature. Improving the evolutionary operators of DE or using some new operators (see [10], [14] and [15]), dynamic and self-adaptive adjusting of DE evolutionary operators i.e. amplification factor of the difference vector, crossover rate and population size (see [16] and [17]) and hybridizing the DE with other local search methods (see [18] and [19]) are some utilized approaches to improve the DE performance. Also Ahandani et al. [13] proposed three modified DE algorithms to compensate limited amount of search moves in the original DE. Their proposed algorithms employed some operators to generate additional donor and offspring members. Also Ahandani and Alavi-Rad [20] introduced four new versions of the DE algorithms. Their proposed DE, on one hand, used the partitioning and shuffling concepts to compensate the limited amount of search moves of the original DE and, on the other hand, employed the opposition-based learning to accelerate the DE without making premature convergence.

This research proposes a modified version of DE algorithm for a better exploration of search space without imposing a new operator. In this DE algorithm, the

generated donor member in the mutation stage can be considered as an offspring with a probability. So based on

The DE algorithm:

Step 1: //initialization //

Generate and evaluate initial population of size N_{pop} from search space

Step 2: //mutation //

Apply mutation operators for each individual according to (1) to obtain each individual donor member (v).

Step 3: //crossover //

Apply crossover operator between each member of population and its corresponding donor member (v) according to (2) to generate an offspring (u).

Step 4: //selection //

Evaluate the generated offspring in cost function according to (3) and if $f(u) < f(x_i)$, replace x_i with u .

Step 5: //stopping condition //

Check the stopping criteria and if are not met go to **Step 2**.

Figure 1. Steps of the DE.

this idea, the second candidate solution is obtained as an offspring without using any new operator.

The remained sections of this paper are organized as follow. In the next section, the original DE algorithm is briefly explained. In section 3, the proposed modified version of DE is described. The simulation results are presented and analyzed in section 4 and section 5 concludes the paper.

2. The Differential Evolution

The structure of the DE algorithm is shown in Fig. 1. The DE similar to other EAs, starts with an initial population extracted randomly from search space. It uses three main operators: mutation, crossover and selection. In the mutation stage, a weighted difference of two random members is added to a third member and a donor member is generated. Among different utilized operators for this stage, this study uses (1).

$$v = x_i + \lambda(x_{best} - x_i) + F(x_{r_2} - x_{r_1}) \quad (1)$$

where x_i is the current point, x_{best} is the best point that is found so far, x_{r_1} and x_{r_2} are two random points that are selected from the population, where $r_1 \neq r_2$ and F and λ are two random numbers within a definite range. The DE with this mutation operator is nominated as DE2 such as Ahandani et al. [13].

In the crossover stage, an operator applies between generated donor member and a predetermined one to generate an offspring presented in (2).

$$u_j = \begin{cases} v_j & j = \langle n \rangle_D, \langle n+1 \rangle_D, \dots, \langle n+l-1 \rangle_D \\ x_j & \text{otherwise} \end{cases} \quad (2)$$

where u_j , v_j and x_j are j^{th} gene of offspring, donor and current individuals, respectively. D is the dimension of problem, n and l are two random integer number in the set of $\{1, 2, \dots, D\}$ that denote starting point and

number of component, respectively. Also angular brackets $\langle \rangle_D$ denote a modulo function with modulus D . According to (3), to generate u , those of genes are among $n-1$ and $n+l$ must be copied from v and other genes will be copied from x .

After generation of new member, in the selection stage based on a greedy criterion, the offspring is evaluated by cost function and if the fitness of new member is better than fitness of current member, this candidate replaces the current member.

$$x_i = \begin{cases} u & f(u) < f(x_i) \\ x_i & f(u) \geq f(x_i) \end{cases} \quad (3)$$

Thus the DE generates an offspring for each member of population in its crossover stage.

3. The Probabilistic Differential Evolution

One of main shortcomings of DE was its few search efforts for exploring whole search space. This drawback beside of using the greedy criterion to accept or reject new generated members causes a fast convergence. The probabilistic-based DE (PBDE) is proposed to give independently selection possibility to donor member as an offspring. In the original DE algorithm, if the fitness of generated offspring was less than fitness of current member, the current member is preserved in the population and this search move is considered as an unsuccessful effort. The PBDE after rejection of new generated offspring by reason of its worse fitness, the generated donor for current member in the mutation stage is evaluated. While the donor member has a better fitness than current member, it is replaced with donor with a probability equal to $Prob$. Steps of the PBDE are shown in Fig. 2. So the PBDE has only an additional stage than the original DE (Step5). This stage does not apply any

The PBDE algorithm:

Step 1: //initialization //

Generate and evaluate initial population of size N_{pop} from search space

Step 2: //mutation //

Apply mutation operators for each individual according to (1) to obtain each individual and donor member (v).

Step 3: //crossover //

Apply crossover operator between each member of population and its corresponding donor member (v) according to (2) to generate an offspring (u).

Step 4: //selection //

Evaluate the generated offspring in cost function according to (3) and if $f(u) < f(x_i)$, replace x_i with u .

Step 5: //probabilistic operator //

Evaluate value of cost function in v and if $f(v) < f(x_i)$ and $rand \leq Prob$ replace x_i with v .

Step 6: //stopping condition //

Check the stopping criteria and if are not met go to **Step 2**.

Figure 2. Steps of the PBDE

additional operator to the DE algorithm and uses the generated donor member in Step 2.

4. Simulation Results and Discussion

In this section, after some experiments to set *Prob* parameter of PBDE algorithm, the PBDE is compared with results of three modern DE algorithms reported in Ahandani et al. [9] on 12 continuous benchmark functions, including 6 low dimensional and 6 high dimensional functions. Then a comparison among the PBDE2 and some other EAs reported in Elbeltagi et al. [21] is carried out. The initial population is randomly generated by uniform probability distribution in all search space. Performance and termination criteria are similar to those of used in Ahandani et al. [13].

Twenty trial runs are performed for each problem. To call a run successful, four threshold values, i.e., 1E-1, 5E-2, 1E-3 and 1E-10 are considered. The performances of the different algorithms are compared using two criteria:

(I) The success rate (SR), the number of runs that algorithm satisfies (4).

$$|(f - f^*)| < \varepsilon \quad (4)$$

where ε is the considered thresholds, f^* is the global minimum of the cost function and f is the cost of the best point found by the algorithm.

(II) Average number of function evaluations (FE) in all runs that algorithm satisfies (4).

Also two criteria are considered for termination of algorithms:

(I) $|(f - f^*)| < \varepsilon_{best}$, where ε_{best} is the best considered value of thresholds.

(II) The objective function value does not improve in several consecutive iterations.

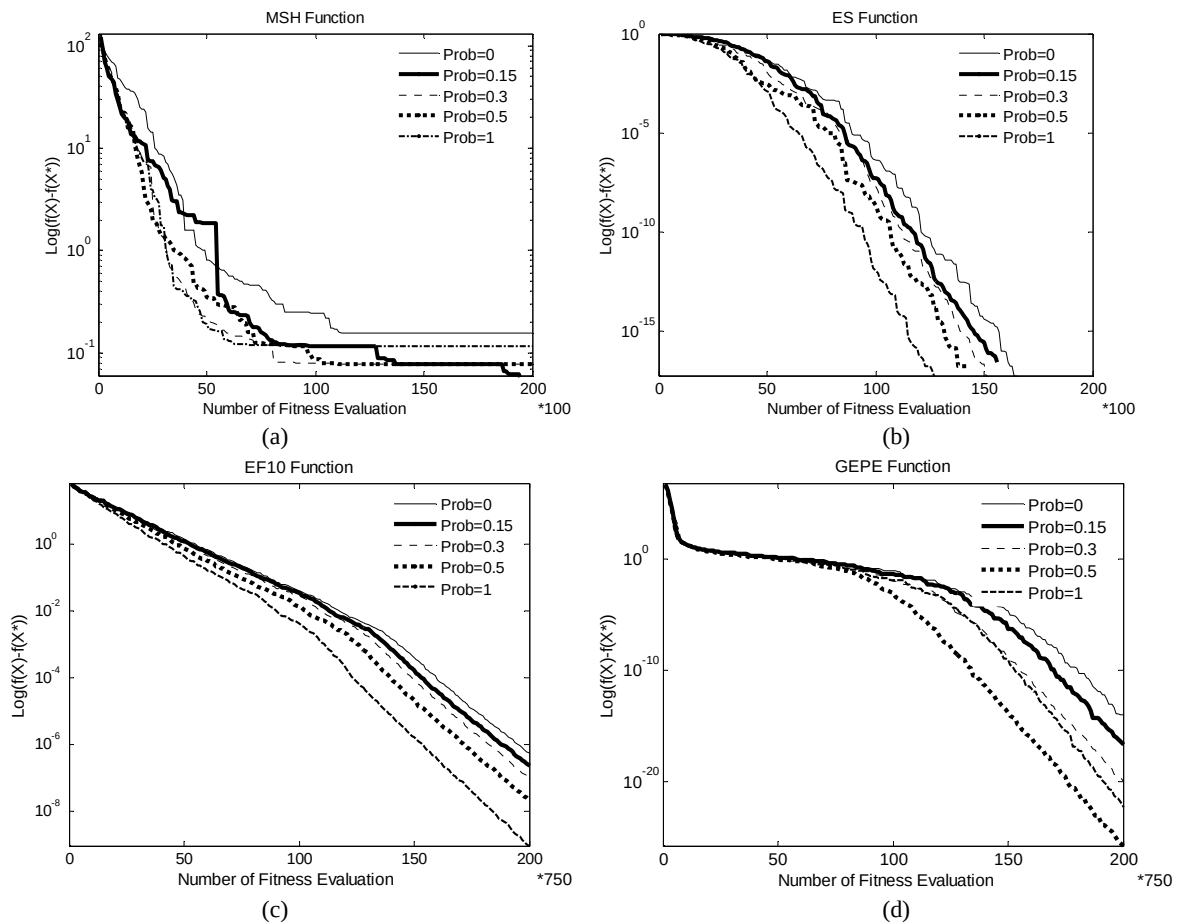
Also size of initial population is considered equal to 100 for those of function with a dimension less than 10 and 200 for those of function with 10 number of variables. Values of λ and F are considered equal to two random number in the range of [0,1] which are considered different for each variable. The obtained results of applying the PBDE2 on benchmark functions and its comparison with other algorithms are shown in Tables 1 to 4.

Table 1 shows the effect of different values of *Prob* on the performance of the PBDE2. The obtained results demonstrate that by increasing *Prob*, the number of function evaluations to reach the considered accuracy is decreased. On ML5 function *Prob* = 0 and *Prob* = 0.3 have the best success rate but *Prob* = 0.3 is faster than *Prob* = 0. Also on all other low-dimensional functions, the PBDE2 with *Prob* = 1 has the fastest performance. On function with 10 variables, *Prob* = 1 obtains the fastest performance, except on GR function. On this function, *Prob* = 0.15 has the best success rate and *Prob* = 0.3 obtains the best number of function evaluations. With incorporation between two considered measures to compare different values of *Prob* i.e. number of function evaluations and success rate, it is considered *Prob* = 0.3 is a proper selection.

Fig. 3 shows convergence characteristic curves for the all considered values for *Prob* in the PBDE2 algorithm, in terms of the fitness value of the median run of algorithm, with over 10 independent runs on MSH, ES, EF10 and GEPE functions. These curves show that how the PBDE2 by increasing the number of function evaluations

Table 1. The results of experiments for adjusting parameter of *Prob* for 1E-3 threshold value. Values in parentheses show the function dimension.

Fun (Dim)	0		0.15		0.3		0.5		1	
	SR	FE	SR	FE	SR	FE	SR	FE	SR	FE
Low-Dimensional Functions										
MSH(2)	100	8543	100	12040	100	8457	100	12085	100	7245
ML5(5)	40	4248	10	802	40	2544	20	1409	20	1364
ES(2)	100	9546	100	8612	100	7368	100	7273	100	5953
GP(2)	100	3547	100	3014	100	2589	100	2658	100	2299
TP3(2)	100	3642	100	2854	100	2535	100	2577	100	2382
Him(2)	100	2959	100	3382	100	2867	100	2683	100	2425
GR(10)	80	170162	100	315282	80	206572	80	476941	70	317786
EF10(10)	100	108694	100	105012	100	95248	100	95099	100	83037
ACK(10)	100	32823	100	31464	100	30114	100	29204	100	26610
GEPE(10)	100	89501	100	86385	100	82273	100	81050	100	76203
ZAK(10)	100	88034	100	74060	100	62428	100	52710	100	37804
ROS(10)	0	-	0	-	0	-	0	-	0	-

**Figure 3.** Median convergence characteristics for PBDE2 over 4 functions (2 low dimensional functions with 20000 function evaluations and 2 high dimensional functions with 10 variables and 150000 function evaluations). (a) MSH function. (b) ES function. (c) EF10 function. (d) GEPE function.

converges toward the optimal solution, and how major values of *Prob* obtain a faster performance.

Table 2 and 3 compare the performance of the PBDE2 and results reported in Ahandani et al. [13] on 6 low-dimensional functions and 6 high-dimensional functions with 10 variables. Ahandani et al. [13] proposed three modified versions of DE algorithm i.e. bidirectional DE (BDE), shuffled DE (SDE) and shuffled bidirectional DE (SBDE) and compared them with original DE algorithm. The obtained results demonstrate that the PBDE2 has a comparable performance than other algorithms. For example, the PBDE2 has a faster performance than the DE2 and BDE2 algorithms on MSH function.

Also the PBDE2 has a better success rate than BDE2 and fastest performance on ML5 function.

Also results of Table 3 show that the PBDE2 has a better success rate than the DE2 and BDE2 algorithms on GR function; however it has a slow performance on this function. Also the PBDE2 has a better success rate than the DE2 and BDE2 on ZAK function. On EF10, ACK and GEPE functions, the PBDE2 has a similar success rate but a slower performance. In an overall consequence, the PBDE2 obtains a better, or at least comparable, performance than original DE2 and BDE2 algorithms. On the other side, the SDE2 and SBDE2 have a faster performance than the PBDE2 algorithm; however their success rate is completely comparable.

Table 2. The comparison of PBDE2 and results reported of Ahandani et al. [13] in terms of success rate and number of function evaluation for low-dimensional functions.

Algorithm	Threshold							
	1E-1		5E-2		1E-3		1E-10	
	SR	FE	SR	FE	SR	FE	SR	FE
MSH								
DE2	100	6370	100	6775	100	8875	100	22750
BDE2	100	6911	100	7510	100	10437	100	24151
SDE2	100	1500	100	1500	100	1920	100	3660
SBDE2	100	3550	100	3550	100	4381	100	9805
PBDE2	100	6071	100	6421	100	8457	100	15652
ML5								
DE2	60	2430	40	1795	40	2880	40	6085
BDE2	50	2848	35	2175	35	3415	35	7306
SDE2	80	2280	60	1760	60	2560	60	4760
SBDE2	100	13730	100	16630	100	37670	100	137490
PBDE2	60	1587	50	1423	40	2544	40	4725
ES								
DE2	100	2830	100	3150	100	4715	100	8510
BDE2	100	3805	100	4285	100	4285	100	4285
SDE2	100	2100	100	2100	100	2100	100	4100
SBDE2	100	5057	100	5057	100	5057	100	82964
PBDE2	100	3293	100	3619	100	7368	100	9848
GP								
DE2	100	1415	100	1620	100	2445	100	5520
BDE2	100	1993	100	2220	100	3710	100	8353
SDE2	100	2100	100	2100	100	2100	100	4100
SBDE2	100	3450	100	3450	100	3450	100	6750
BDE2	100	995	100	1117	100	2589	100	5749
TP3								
DE2	100	1340	100	1340	100	1830	0	-
BDE2	100	1340	100	1340	100	1830	0	-
SDE2	100	2100	100	2100	100	2100	0	-
SBDE2	100	3010	100	3010	100	4000	0	-
PBDE2	100	1823	100	1963	100	2535	0	-
Him								
DE2	100	970	100	1020	100	1910	100	5890
BDE2	100	1155	100	1472	100	3106	100	8995
SDE2	100	1260	100	1260	100	1260	100	2460
SBDE2	100	2615	100	2615	100	2615	100	5032
PBDE2	100	955	100	1195	100	2867	100	6912

Table 4 shows a comparison among the proposed algorithm and results of Elbeltagi et al. [21]. Elbeltagi et al. [21] have compared four continuous algorithms i.e. the GAs, memetic algorithms (MAs), particle swarm optimization (PSO) and shuffled frog leaping (SFL) on two continuous benchmark functions. Results of table 4 demonstrate that the PBDE2 has a better performance than other algorithms. The BDE2 has a success rate of 100 on all cases. The MAs and the PSO have only a comparable performance on GR and EF0 functions, respectively.

5. Conclusions

This paper proposed a probabilistic-based DE algorithm named PBDE. The PBDE is an effort to associate the generated member in the mutation stage of DE for utilization independently as an offspring. The DE beside of its considerable traits such as simple operator to generate new offspring, easy implementation and fast convergence has a limited amount of search moves to explore whole search space. The PBDE2 apply an additional move to original DE without utilizing a new operator. The proposed algorithm was applied on some continuous benchmark functions and was compared with

some modern and recently proposed DE and other EAs. The obtained results demonstrated that the PBDE2 had a completely comparable performance than other versions of DE algorithms. Also it outperformed other compared EAs in the considered paper.

Acknowledgements

We Thanks the Young Researchers Club, Langaroud Branch, for supporting this research.

References

- [1] R. Storn, K.Price, Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11 (1997) 341-359.
- [2] Y-P Chang, C Low, An ant direction hybrid differential evolution heuristic for the large scale passive harmonic filters planning problem, *Expert Systems Applications*, 7 (2007) 4157-4174.
- [3] J-P Chiou, Variable scaling hybrid differential evolution for large-scale economic dispatch problems, *Electrical Power Systems Research*, 77 (2007) 212-218.

Table 3. The comparison of PBDE2 and results reported of Ahandani et al. [13] in terms of success rate and number of function

evaluation for high-dimensional functions.

Algorithm	Threshold							
	1E-1		5E-2		1E-3		1E-10	
	SR	FE	SR	FE	SR	FE	SR	FE
GR(10)								
DE2	95	8700	95	12100	20	4020	10	5740
BDE2	100	17307	100	30832	10	32149	0	-
SDE2	100	14800	100	16400	80	25000	80	50900
SBDE2	100	31030	100	43170	100	112930	100	218850
PBDE2	100	77016	100	120382	80	206572	80	427104
EF10(10)								
DE2	100	20875	100	23050	100	34375	100	66645
BDE2	100	28289	100	30996	100	47413	100	93132
SDE2	100	23100	100	26100	100	40900	100	86500
SBDE2	100	24427	100	26312	100	36816	100	70555
PBDE2	100	60192	100	66824	100	95248	100	188446
ACK(10)								
DE2	100	6855	100	7415	100	11330	100	28020
BDE2	100	9421	100	10179	100	15830	100	39539
SDE2	100	7400	100	8000	100	14100	100	42200
SBDE2	100	12938	100	12938	100	17685	100	42441
PBDE2	100	18927	100	20520	100	30114	100	76302
GEPE(10)								
DE2	100	20350	100	21605	100	25710	100	35265
BDE2	100	50056	100	52918	100	65509	100	93515
SDE2	100	14200	100	15500	100	19500	100	35500
SBDE2	100	21954	100	22757	100	26732	100	36532
PBDE2	100	63396	100	66238	100	82273	100	107054
ZAK(10)								
DE2	100	3440	100	3705	100	5755	0	-
BDE2	100	59659	100	64529	85	84459	10	12407
SDE2	100	8500	100	9300	100	17700	100	50500
SBDE2	100	26259	100	26880	100	33642	100	64071
PBDE2	100	44035	100	47709	100	62428	100	123852
ROS(10)								
DE2	95	644060	45	398760	0	-	0	-
BDE2	85	729670	75	778460	0	-	0	-
SDE2	25	6020	20	6420	0	-	0	-
SBDE2	20	22689	10	5855	10	11909	0	-
PBDE2	20	223184	0	-	0	-	0	-

Table 4. A comparison among the proposed algorithms and results of Elbeltagi et al. [21]

Comparison criteria	Algorithm	Number of variables						
		GR				EF10		
		10	20	50	100	10	20	50
%Success	PBDE2	100	100	100	100	100	100	100
	GAs (Evolver)	50	30	10	0	20	0	0
	MAs	90	100	100	100	100	70	0
	PSO	30	80	100	100	100	80	60
	SFL	50	70	90	100	80	20	0
Mean Solution	PBDE2	0.0476	0.0454	0.0469	0.0479	0.0476	0.0479	0.0491
	GAs (Evolver)	0.06	0.097	0.161	0.432	0.455	1.128	5.951
	MAs	0.014	0.013	0.011	0.009	0.014	0.068	0.552
	PSO	0.093	0.081	0.011	0.011	0.009	0.075	2.895
	SFL	0.08	0.063	0.049	0.019	0.058	2.252	6.469

[4]H.A. Abbass, An Evolutionary Artificial Neural Networks Approach for Breast Cancer Diagnosis, Artificial Intelligence in Medicine, 25 (2002)

265-281.

[5]A.C. Nearchou, A differential evolution approach for the common due date early/tardy job scheduling problem, Computers and Operations Research, 32 (2008) 1329-1343.

[6]T.K. Sharma, M. Pant, V.P. Singh, Modified mutation in differential evolution algorithm to optimize supply chain system, Journal of Nature Inspired Computing, 1 (2012) 1-8.

[7]S. Amanullah, I. Danish, F. Mehmood, Comparison of conventional helix with genetic algorithm optimized helix, Advances in Electrical Engineering Systems, 1 (2012) 186-188

[8]A. ghoreishi, A. Ahmadvand, State feedback design aircraft landing system with using differential evolution algorithm, Advances in Computer Science and its Applications, 1 (2012) 16-20.

[9]P. Kumar, differential evolution with interpolation based mutation operators for engineering design optimization, Advances in Computer Science and its Applications, 2 (2012) 221-231.

- [10] Y. Singh, P. Chauhan, Analyzing constrained machining conditions in turning operations by differential evolution, *Advances in Computer Science and its Applications*, 2 (2012) 201-206.
- [11] S. Kumar, 2D maximum entropy method for image thresholding converge with differential evolution, *Advances in Computer Science and its Applications*, 2 (2012) 189-192.
- [12] V.P. Plagianakos, D.K. Tasoulis, M.N. Vrahatis, A review of major application areas of differential evolution, *Advances in differential evolution*, 143 (2008) 197-238.
- [13] M.A. Ahandani N.P. Shirjoposht, R. Banimahd, Three Modified Versions of Differential Evolution Algorithm for Continuous Optimization, *Soft Computing*, 15 (2010) 803-830.
- [14] M.M. Ali, Differential evolution with preferential crossover. *European Journal of Operational Research*, 181 (2007) 1137-1147.
- [15] S. Das, A. Abraham U.K. Chakraborty, A. Konar, Differential evolution using a neighborhood-based mutation operator, *IEEE Transactions on Evolutionary Computation*, 13 (2009) 526-553.
- [16] A.K., Qin V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, *IEEE Transactions on Evolutionary Computation*. 13 (2009), 398-417.
- [17] N.S., Teng, J. Teo, M.H.A. Hijazi, Self-adaptive population sizing for a tune-free differential evolution, *Soft Computing*, 13 (2009) 709-724.
- [18] A. Caponio, F. Neri, V. Tirronen, Super-fit control adaptation in memetic differential evolution frameworks, *Soft Computing.*, 13 (2009) 811-831.
- [19] F. Neri, V.Tirronen, Scale factor local search in differential evolution, *Memetic Computing.*, 1, (2009) 153-171.
- [20] M.A. Ahandani, H Alavi-Rad, Opposition-based learning in the shuffled differential evolution algorithm, *Soft Computing*, 16 (2012) 1303-1137
- [21] E. Elbeltagi, T. Hegazy, D. Grierson, Comparison among five evolutionary-based algorithms, *Advanced Engineering Informatics*, 19 (2005) 43-53.



M.A. Ahandani was born in 1983 in Lahijan, Iran. He received his B.Sc. and M.Sc degrees in Electronics and control engineering, from Azad University of Lahijan and Tabriz University in 2006 and 2009. He has just completed his M.Sc. working on soft computing and scheduling. Since 2010, he has been a lecturer at Islamic Azad University, Langaroud branch, Langaroud, Iran.

His research interests include timetabling problems, soft computing and evolutionary algorithms. Mr. Ahandani is author over ten papers in international journals and international conferences. He is peer reviewer several journals such as *Soft Computing*, *Information Science*, *Swarm and Evolutionary Computation*, and *Computational Optimization and Applications*.