ADAPTIVE DIFFERENTIAL EVOLUTION FOR CONSTRAINED OPTIMIZATION PROBLEMS

T. T. SHAH[1], M. A. JAN[2], W. K. MASHWANI[3] AND H. WAZIR[4]
[1,2,3,4] Department of Mathematics, Kohat University of Science & Technology, KPK, Pakistan
E-mails: tayyabashah37@yahoo.com, majan@kust.edu.pk, mashwanigr8@gmail.com, hamzawazir@hotmail.com

ABSTRACT: *Adaptive differential evolution with optional external archive (JADE) is an efficient unconstrained optimization and search algorithm. It has shown very promising results for CEC'05 test suite. In this paper, JADE is modified for solving constrained optimization problems. The modification introduces a dynamic penalty function in the selection scheme of JADE to handle constraints. This develops a new modified algorithm, denoted by CJADE-D. A well-known test function suite of constrained optimization problems, CEC'06 is used to examine the performance of CJADE-D. Four different parameter settings of the adopted dynamic penalty function are used to obtain the experimental results. The results show that the performance of CJADE-D is good for most of the problems.*

Keywords: Evolutionary algorithm, differential evolution, JADE, constraint handling, dynamic penalty function.

1. INTRODUCTION

Many problems in the field of mathematics, engineering, economics, operation research etc., involve some functional constraints i.e., we are always interested in maximizing our profit and productivity with minimum cost in each sector. In our work, we consider the following constrained single-objective non-linear optimization problem (in the minimization sense):

$$\begin{aligned} &Minimize \quad f(\bar{x}), \\ &Subject\ to: \quad g_i(\bar{x}) \le 0, i = 1,2,3,\dots,p; \\ &\qquad\qquad h_j(\bar{x}) = 0, j = p+1,\dots,q, \quad (1) \end{aligned}$$

where $\bar{x} = (x_1, x_2, \dots, x_n)^T$ is the solution vector, p is the number of inequality constraints and $q - p$ is the number of equality constraints. If S is the search space and F is the feasible region, then $F \subseteq S$.

Evolutionary algorithms (EAs), mainly inspired from the natural evolution of species, have been applied on wide range of optimization and search problems [1], [2]. Basic evolutionary operators associated with EAs are mutation, crossover and selection. Good individuals of better objective function values in the population are selected to become parents of the next generation. However, EAs in their original shape are unconstrained search methods. They require additional techniques for handling constraints [3]. In last many decades, EAs have got much attention to solve constrained optimization problems, and a number of algorithms are thus proposed (e.g., see [4], [5]).

Differential Evolution (DE) is a well-known EA which was first proposed by Rainer Storn and Kenneth Price [6] in 1997. DE is a population based search technique for solving optimization problems. Its adaptive version, adaptive differential evolution with optional external archive (JADE) was introduced in [7], which implements a generalized mutation strategy and sets the parameters in an adaptive way. In [7], the performance of JADE was reported on a set of unconstrained benchmark functions. Comparison of reported results obtained from JADE and other algorithms, shows the superiority of JADE in [7].

In our research work, we have modified JADE by using penalty functions in its selection scheme. Suggested modification is used to penalize all those infeasible solutions which violates the given constraints and goal to reach feasible optimal solution. This way a new constrained variant of JADE, named as CJADE-D is introduced. Now CJADE-D is capable to solve constrained optimization problem. We analyse the performance of CJADE-D on test function suite of constrained optimization problems, CEC'06 [8].

The remainder of this paper is distributed in various sections. Section 2 briefly explains the DE algorithm. Section 3 defines its adaptive version, JADE. Section 4 gives details of penalty function methods along with our modification. Section 5 discusses the experimental results and finally, Section 6 concludes this paper.

2. DIFFERENTIAL EVOLUTION

Differential evolution (DE) is one of the most efficient and commonly used EA. It has a stochastic nature.

DE generates a random initial population of size NP in search space S, i.e., $\boldsymbol{x}_i = (x_{i,1}, x_{i,2}, \dots, x_{i,n})$, where $i = 1,2,\dots,NP$ and n is the problem dimension. These solutions are then perturbed by genetic operators to produce offspring. The parents and offspring then compete based on their fitness value to survive for next generation. Some parameters involved in DE are NP (population size), F (mutation factor) and Cr (crossover ratio). The basic evolutionary operators of DE, mutation, crossover and selection are detailed in the following.

***A. Mutation*:** At each generation G, the mutated vector $\boldsymbol{v}_{i,G}$ is generated corresponding to each individual vector $\boldsymbol{x}_{i,G}$ called target vector. Following are some mutation strategies that are frequently used by researchers:

I. DE/rand/1:

$$\boldsymbol{v}_{i,G} = \boldsymbol{x}_{r1,G} + F * (\boldsymbol{x}_{r2,G} - \boldsymbol{x}_{r3,G}), \quad (2)$$

II. DE/best/1:

$$\boldsymbol{v}_{i,G} = \boldsymbol{x}_{best,G} + F * (\boldsymbol{x}_{r1,G} - \boldsymbol{x}_{r2,G}), \quad (3)$$

III. DE/current-to-best/1:

$$\boldsymbol{v}_{i,G} = \boldsymbol{x}_{i,G} + F * (\boldsymbol{x}_{r1,G} - \boldsymbol{x}_{r2,G}) + F * (\boldsymbol{x}_{best,G} - \boldsymbol{x}_{i,G}), \quad (4)$$

where $r1, r2$ and $r3$ are distinct random integers chosen from the set $1,2,\dots,NP$. F is a mutation factor and $\boldsymbol{x}_{best,G}$ is the best individual at current generation. Mutation strategy *"DE/rand/1"* mutates a random solution with one difference vector as shown in Eq. (2). Similarly, *"DE/best/1"* mutates a best solution with one difference vector as shown in Eq. (3). *"DE/current-to-best/1"* mutates a current solution with a

difference vector of random solutions and a best solution as shown in Eq. (4).

***B. Crossover*:** After mutation operation, offspring vector or trial vector $\boldsymbol{u}_{i,G} = (u_{1,i,G}, u_{2,i,G}, \dots, u_{n,i,G})$ is generated by applying binomial crossover operation, i.e.,

$$u_{j,i,G} = \begin{cases} v_{j,i,G}, if\ rand(0,1) \le Cr\ or\ j = j_{rand} \\ x_{j,i,G}, \qquad otherwise, \end{cases} \quad (5)$$

where $Cr \in [0,1]$ is crossover probability and $j = 1,2,\dots,n$.

***C. Selection*:** In this step, a better solution vector from the parent vector $\boldsymbol{x}_{i,G}$ and offspring vector $\boldsymbol{u}_{i,G}$ is selected on the basis of their fitness values, i.e., in minimization problems, the solution vector is selected by:

$$\boldsymbol{x}_{i,G+1} = \begin{cases} \boldsymbol{u}_{i,G}, & if\ f(\boldsymbol{u}_{i,G}) < f(\boldsymbol{x}_{i,G}) \\ \boldsymbol{x}_{i,G}, & otherwise. \end{cases} \quad (6)$$

It then becomes the parent vector of next generation. Above mentioned evolutionary process continues till the specified stopping criteria is satisfied.

3. ADAPTIVE DIFFERENTIAL EVOLUTION WITH OPTIONAL EXTERNAL ARCHIEVE (JADE)

The mutation strategies of DE defined in Eqs. (2), (3) and (4) was supposed to be the robust and greedy strategies. These greedy strategies sometimes may cause premature convergence. Trial-and-error search for suitable mutation strategy and to set suitable parameters is also time consuming in original DE.

A new adaptive DE algorithm, JADE is introduced in [7], which implements mutation strategy *"DE/current-to-pbest/1"* (a generalization of *"DE/current-to-best/1")*, and adjusts the parameters F (mutation factor) and Cr (crossover probability) in an adaptive way. Suggested mutation strategy in JADE utilizes the information of best solutions as well as the information of other good solutions. Any of the top $100p\%$, $p \in (0,1]$, solutions in the current population can be randomly chosen in *"DE/current-to-pbest"*. These solutions are utilized as a single best solution in *"DE/current-to-best"*. Also, it may vary the population which can control the premature convergence [7]. JADE follows the same crossover and selection operations in its evolutionary process, as used in original DE [6], and defined in Section 2.

4. PENALTY FUNCTION METHODS

Like other EAs, JADE is an unconstrained optimization search algorithm. Therefore, it requires some other techniques to solve constrained optimization problems, i.e., techniques to deal with violated solutions. Penalty functions are the most frequently used methods of handling constraints. These methods have received more attention due to simplicity.

Penalty functions were first introduced by Courant [9] in 1940s. After that, Carroll [10] and Fiacco and McCormick [11] worked a lot on penalty functions and introduce various penalty models. The objective of this approach is to convert a constrained optimization problem into an unconstrained optimization problems by adding, in minimization sense (or subtracting, in maximization sense) a penalty term to the objective function. Generally a penalty function can be formulated as follows [3]:

$$F_p(\bar{x}) = f(\bar{x}) \pm [\sum P1_i * G_i + \sum P2_j * H_j], \quad (7)$$

where F_P is the new penalized objective function, $P1_i$ and $P2_j$ are penalty factors. $G_i = \max[0, g_i(\bar{x})]^{\beta}$ and $H_j = |h_j(\bar{x})|^{\gamma}$ are functions of inequality and equality constraints, respectively In this paper, we use a dynamic penalty function proposed by Joines and Houck [12]. It is discussed as follows.

4.1. Dynamic Penalty Function

During the evolutionary process, a dynamic penalty function increases the penalty with the increase in generation number G to penalize infeasible solutions. Joines and Houck [12] proposed the following dynamic penalty function in 1994:

$$f_p(\bar{x}) = f(\bar{x}) + (CG)^{\alpha}\, V(\beta, \bar{x}\,), \quad (8)$$

where $\overline{\boldsymbol{x}} = (x_1, x_2, \dots, x_n)^T$ is the solution vector, α and β are constants defined by users. Joines and Houck [12] suggested $\alpha = 1\ or\ 2$, $\beta = 1\ or\ 2$, and $C = 0.5$, $V(\beta, \bar{x})$ can be described as:

$$V(\beta, \bar{x}\,) = \sum_{i=1}^{p} g_{vj}(\bar{x})^{\beta} + \sum_{j=p+1}^{q} h_{vj}(\bar{x}), \quad (9)$$

where

$$g_{vi}(\bar{x}) = \begin{cases} 0, & if\ \bar{x}\ is\ feasable\ for\ i = 1,2,\dots,p \\ |g_i(\bar{x})|, & otherwise, \end{cases}$$

and

$$h_{vj}(\bar{x}) = \begin{cases} 0, & if\ \bar{x}\ is\ feasable\ for\ j = p+1,\dots,q \\ |h_j(\bar{x})|, & otherwise. \end{cases}$$

In this work, we define four different versions of the above dynamic penalty function based on the values of adopted parameters, i.e.,

Version 1: $\alpha = 1, \beta = 1, C = 0.5$
Version 2: $\alpha = 1, \beta = 2, C = 0.5$
Version 3: $\alpha = 2, \beta = 1, C = 0.5$
Version 4: $\alpha = 2, \beta = 2, C = 0.5$

5. EXPERIMENTAL SETTINGS AND DISCUSSION

In this section, first we comment on the experimental settings, and then discuss our findings from the conducted experiments.

5.1. Experimental Settings

Performance of CJADE-D is tested on 24 test functions with constraints of CEC 2006 [8]. Details of these functions are given in Table 1. Classification of the problems on the basis of number of variables is also shown in Table 2. The algorithm is executed 25 times independently with population of size 100 for each problem. The algorithm stops after 5000 generations (or when $FES = 500{,}000$) in each run. All simulations are conducted in MATLAB® (Ra2011). Moreover, the experiments are conducted with four different values of α and β (as mentioned in Section 4 above and used in Eq. 9). All other parameters' settings are fixed for each problem.

Table 1: Details and properties of 24 test problems [8].

Problem	n	Type of function	ρ	INE	EQ	a
g01	13	quadratic	0.0111%	9	0	6
g02	20	nonlinear	99.997%	2	0	1
g03	10	polynomial	0.0000%	0	1	1
g04	5	quadratic	52.123%	6	0	2
g05	4	cubic	0.0000%	2	3	3
g06	2	cubic	0.0066%	2	0	2
g07	10	quadratic	0.0003%	8	0	6
g08	2	nonlinear	0.8560%	2	0	0
g09	7	polynomial	0.5121%	4	0	2
g10	8	linear	0.0010%	6	0	6
g11	2	quadratic	0.0000%	0	1	1

g12	3	quadratic	4.7713%	1	0	0
g13	5	nonlinear	0.0000%	0	3	3
g14	10	nonlinear	0.0000%	0	3	3
g15	3	quadratic	0.0000%	0	2	2
g16	5	nonlinear	0.0204%	38	0	4
g17	6	nonlinear	0.0000%	0	4	4
g18	9	quadratic	0.0000%	13	0	6
g19	15	nonlinear	33.476%	5	0	0
g20	24	linear	0.0000%	6	14	16
g21	7	linear	0.0000%	1	5	6
g22	22	linear	0.0000%	1	19	19
g23	9	linear	0.0000%	2	4	6
g24	2	linear	79.655%	2	0	2

where n is the problem dimension, $\rho = |F|/|S|$ is the approximated ratio between the feasible region and the search space S, INE represents inequality constraints, EQ represents equality constraints and a represents active constraints at $\bar{x}$ [8].

Table 2: Classification of problems according to the number of values.

Classification of Problems		
Low	Medium	High
Number of Variables		
2-4	5-9	10-20
g05 g06 g08 g11 g12 g15 g24	g04 g09 g10 g13 g16 g17 g18 g21 g23	g01 g02 g03 g07 g14 g19 g20 g22

5.2. DISCUSSION ON NUMERICAL RESULTS

Tables 3, 4, 5 and 6 show the best, worst, mean, median and standard deviations values obtained from our experiments. The last columns of these tables also show the known optimal values. Table 7 presents a comparison of the four paradigms of CJADE-D.

Experimental results in Table 3 conclude that version 1 is effective for g01, g02, g07, g09, g12, 16, g19, and g24. In this version penalty rate is low. Parameters setting of version 2 is good for the problems g01, g02, g05, g07, g09, g12, g16 and g19, as shown in Table 4. Table 5 and 6 show the results obtained by using version 3 and 4, respectively. Both Tables show the effectiveness of the two versions on same problems, i.e., g01, g02, g09, g12, g16 and g19. Each version is effective for different problems due to different penalty coefficients.

The results in Table 7 show that the performance of modified JADE, CJADE-D is good for more than 50% problems in general. For comparison of different versions, we consider best values and mean values. Version 1 gives good results for the problems g07, g09, g12, g16, g19, g24. Version 2 shows better results for the problems g01, g02, g05, g12, g15, g17, g18. Version 3 and 4 are not so

encouraging as their performance is better only for g01, g09, g12, g16, and g19. In these two versions, penalty coefficients are not suitable for the rest of the problems. Although some problems are hard and have large search space, so they require more tuning in penalty parameters. Bold results shows the better results.

5.3. Discussion on Graphical Results

Following figures show convergence graphs of few functions. Each graph shows average values of the best solutions obtained across 5000 generations and 25 runs. The four different versions of CJADE-D are differentiated with different colours and marker styles. It can easily be viewed from these figures that versions 1 and 2 are good for most of the problems.

6. CONCLUSION

In this paper, we modified the adaptive DE algorithm, i.e., JADE for handling constraints in constrained optimization. A dynamic penalty function is introduced in the selection scheme of JADE to penalize infeasible solutions, and thus a constrained version of JADE, denoted by CJADE-D is developed. The performance of CJADE-D is reported on well-known test function suite CEC'06. Moreover, four different versions of the algorithm are introduced by using four different settings of parameters used in the dynamic penalty function. We noticed from our experimental results that Versions 1 and 2 are the most efficient versions of the dynamic penalty function because of their better performance for various problems. Version 3 and 4 show their effectiveness for the same problems that are already optimized by first two versions. Experimental results also show the effectiveness of this work. However, this work leaves plenty of room for improvement. All these results are preliminary work on CEC'06. In future, we intend to employ some more dynamic penalty functions in JADE to see the effectiveness of each one of them in its framework. We also anticipate to show the performance of CJADE-D according to the evaluation criteria as explained in [8].

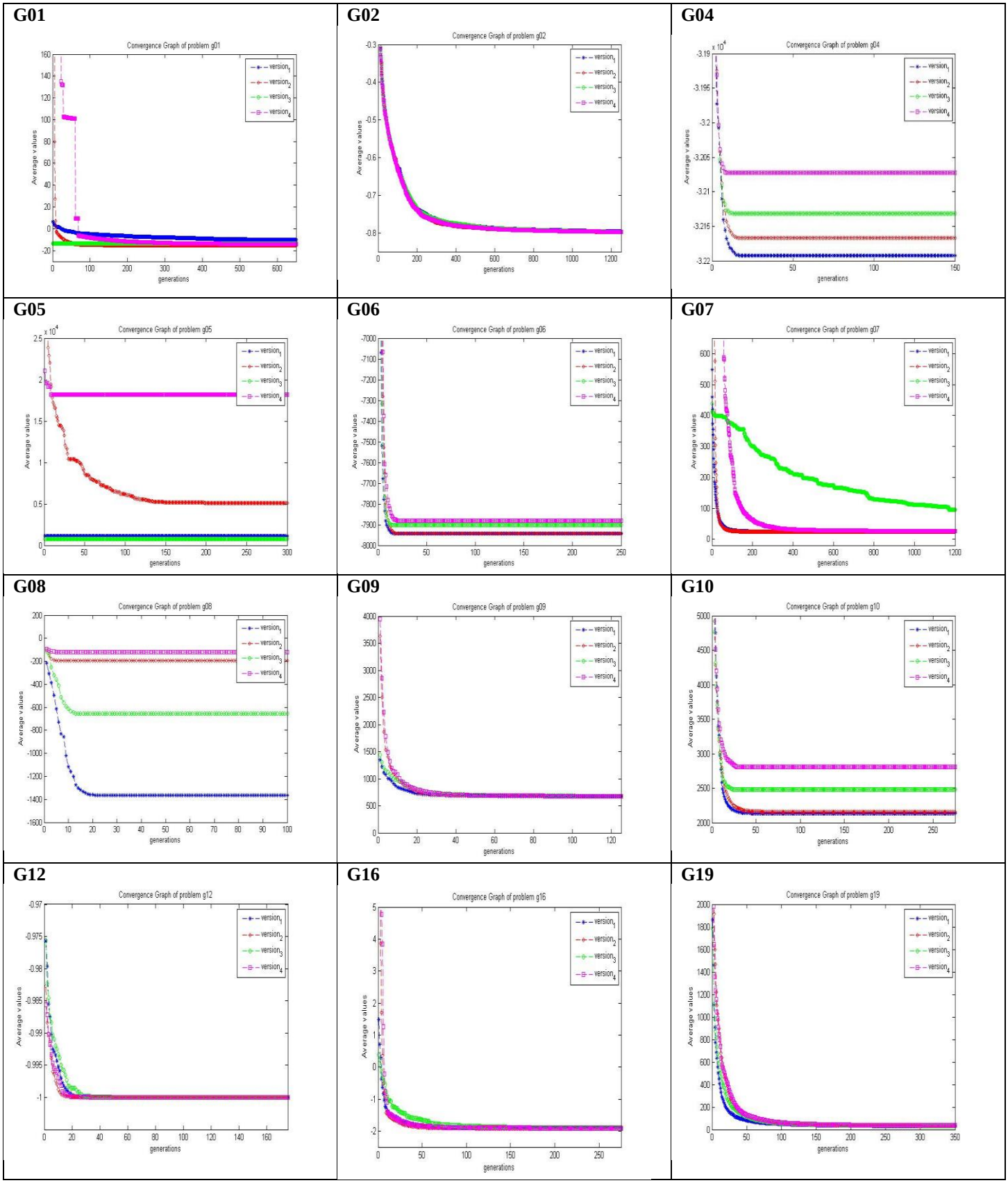

Figure 1: Ccnvergence Graphs of Test Functions

Table 3: Statistical Results of 24 Problems by Using Version 1

Problems	Optimum	Best Achieved	Worst	Median	Mean	St. dev
g01	-15.000000	-14.999339	-14.829761	-14.664120	-12.969465	5.894e+000
g02	-0.803619	-0.802628	-0.799956	-0.801388	-0.801371	7.083e-004
g03	-1.000000	-99675.835134	-99710.493976	-99690.148408	-99689.240788	8.574e+000
g04	-30665.539000	-32196.152588	-32188.751696	-32192.360668	-32192.275678	2.138e+000
g05	5126.498000	1362.819559	848.675941	1074.561136	1081.791600	1.215e+002
g06	-6961.814000	-7962.000000	-7941.880483	-7943.279110	-7944.403151	3.903e+000
g07	24.306000	24.306209	24.545117	24.306209	24.321855	4.837e-002
g08	-0.095825	-1296.806712	-1423.972572	-1346.209597	-1348.735625	3.397e+001
g09	680.630000	680.630057	680.630057	680.630057	680.630057	2.344e-013
g10	7049.331000	2131.846129	2155.026553	2140.014032	2140.683593	4.949e+000
g11	0.750000	0.443553	0.437688	0.440110	0.440157	1.834e-003
g12	-1	-1	-1	-1	-1	0.000e+000
g13	0.053950	1.201125	2.601442	1.775458	1.837891	3.967e-001
g14	-47.764411	-1420.02254	-1661.53550	-1570.92955	-1561.62470	6.536e+001
g15	961.715172	771.641627	665.402777	709.110988	709.556725	2.332e+001
g16	-1.905155	-1.905155	-1.905155	-1.905155	-1.905155	4.532e-016
g17	8876.980680	453.620550	2213.706348	1506.524787	1475.586258	4.239e+002
g18	-0.865735	32.185920	133.707982	90.087847	87.105215	2.959e+001
g19	32.655593	32.655593	36.372887	32.655593	33.110947	9.493e-001
g20	0.096737	99.408663	76.735985	89.579922	88.649597	5.305e+000
g21	193.778349	475.884809	251.604811	345.31174	345.569134	6.946e+001
g22	382.902205	-	-	-	-	-
g23	-400.002500	-3022.66568	-2199.25821	-2439.89209	-2458.25713	1.864e+002
g24	-5.508013	-5.508013	-5.761019	-5.631511	-5.628632	7.837e-002

.

Table 4: Statistical Results of 24 Problems by Using Version 2.

Problems	Optimum	Best Achieved	Worst	Median	Mean	St. dev
g01	-15.000000	-15.000284	-15.000227	15.000256	-15.000255	1.341e-005
g02	-0.803619	-0.803558	-0.801573	-0.802930	-0.802732	5.386e-004
g03	-1.000000	-97628.264744	-98141.663749	-97824.812948	-97829.864487	1.147e+002
g04	-30665.539000	-32172.599949	-32159.838470	-32167.317368	-32166.981986	4.028e+000
g05	5126.498000	5126.449507	5126.419072	5126.427912	5126.430191	7.047e-003
g06	-6961.814000	-7945.007066	-7941.988298	-7942.721877	-7943.096876	9.390e-001
g07	24.306000	24.301221	24.303034	24.301836	24.301923	4.719e-004
g08	-0.095825	-59.518724	-1044.936574	-83.862457	-192.699596	2.477e+002
g09	680.630000	680.627899	680.626873	680.627363	680.627328	1.939e-004
g10	7049.331000	2145.939349	2164.720439	2157.985385	2157.301978	4.833e+000
g11	0.750000	0.350202	0.333494	0.335778	0.336515	3.549e-003
g12	-1	-1	-1	-1	-1	0.000e+000
g13	0.053950	0.068077	0.859996	0.324918	0.381841	2.267e-001
g14	-47.764411	-32.497629	586.298716	312.251181	319.366876	1.737e+002
g15	961.715172	961.711596	961.708990	961.710588	961.710500	5.112e-004
g16	-1.905155	-1.905389	-1.905350	-1.905364	-1.905366	1.070e-005
g17	8876.980680	8852.251703	8954.536857	8881.277758	8888.782128	3.668e+001
g18	-0.865735	-0.866058	-0.673399	-0.866052	-0.858339	3.853e-002
g19	32.655593	32.655263	34.208323	32.655366	33.246513	8.143e-001
g20	0.096737	13904.386462	7574.299709	11443.349785	11047.776752	1.731e+003
g21	193.778349	304.126717	65.564523	187.746073	194.813211	7.753e+001
g22	382.902205	-	-	-	-	-
g23	-400.002500	-1019.367279	-247.298193	-800.793787	-769.276861	1.772e+002
g24	-5.508013	-5.542029	-5.737026	-5.630906	-5.640892	5.519e-002

Table 5: Statistical Results of 24 Problems by Using Version 3.

Problems	Optimum	Best Achieved	Worst	Median	Mean	St. dev
g01	-15.000000	-14.536362	-12.623072	-13.596435	-13.576798	4.884e-001
g02	-0.803619	-0.803096	-0.800685	-0.801478	-0.801542	5.705e-004
g03	-1.000000	-94725.828501	-96062.676195	-95625.001578	-95640.658296	3.251e+002
g04	-30665.539000	-32153.561671	-32082.964317	-32128.197631	-32121.669005	2.027e+001
g05	5126.498000	1091.155898	491.063763	686.671960	680.857312	1.209e+002
g06	-6961.814000	-7923.373172	-7880.499303	-7895.403131	-7897.210547	1.173e+001
g07	24.306000	24.306209	353.914092	27.803982	41.412885	6.517e+001
g08	-0.095825	-32.344341	-1224.941978	-811.359476	-710.303801	3.781e+002
g09	680.630000	680.630057	680.630057	680.630057	680.630057	2.378e-013
g10	7049.331000	2272.958937	2686.759763	2511.481525	2500.159555	9.615e+001
g11	0.750000	0.252099	0.234373	0.236317	0.237485	4.210e-003
g12	-1	-1	-1	-1	-1	0.000e+000
g13	0.053950	0.736388	1.775409	1.259539	1.238076	2.540e-001
g14	-47.764411	-1444.352863	-1725.637015	-1550.658350	-1570.498290	7.113e+001
g15	961.715172	691.913649	561.887878	611.009098	615.130240	3.292e+001
g16	-1.905155	-1.905155	-1.905155	-1.905155	-1.905155	2.066e-011
g17	8876.980680	830.311696	2659.975004	1353.220608	1417.665163	5.346e+002
g18	-0.865735	14.012411	72.095287	47.307690	46.079484	1.551e+001
g19	32.655593	32.655593	36.071229	32.655593	33.503592	1.032e+000
g20	0.096737	56.510557	43.196474	50.512803	50.586572	3.257e+000
g21	193.778349	313.537859	97.700696	205.226167	207.261941	5.419e+001
g22	382.902205	-	-	-	-	-
g23	-400.002500	-2804.065489	-1543.316592	-2152.572383	-2097.655354	3.290e+002
g24	-5.508013	-5.884972	-6.062107	-5.957645	-5.965597	5.447e-002

Table 6: Statistical Results of 24 Problems by Using Version 4.

Problems	Optimum	Best Achieved	Worst	Median	Mean	St. dev
g01	-15.000000	-15.000001	-13.000000	-15.000000	-14.839069	5.535e-001
g02	-0.803619	-0.802444	-0.800467	-0.801246	-0.801357	5.252e-004
g03	-1.000000	-78856.859004	-83959.929738	-81187.338343	-81291.317782	1.439e+003
g04	-30665.539000	-32168.512026	-32026.624363	-32077.009058	-32079.515834	3.315e+001
g05	5126.498000	62092.254197	6781.278291	19497.361357	21944.880460	1.248e+004
g06	-6961.814000	-7920.674774	-7826.417087	-7879.821153	-7879.538177	2.395e+001
g07	24.306000	24.306205	26.202911	25.231804	25.216320	5.694e-001
g08	-0.095825	-24.963049	-1393.643235	-58.273993	-239.138354	3.378e+002
g09	680.630000	680.630049	680.630040	680.630044	680.630044	2.401e-006
g10	7049.331000	2537.690323	3138.317494	2759.787554	2775.485299	1.418e+002
g11	0.750000	0.217363	0.200290	0.203082	0.204685	4.559e-003
g12	-1	-1	-1	-1	-1	0.000e+000
g13	0.053950	0.645049	4.363228	2.208718	2.301202	8.344e-001
g14	-47.764411	23.263266	-182.264671	-108.969973	-100.020889	4.950e+001
g15	961.715172	973.457666	961.183346	965.103847	965.384840	2.795e+000
g16	-1.905155	-1.905156	-1.905155	-1.905156	-1.905156	1.255e-007
g17	8876.980680	12611.343529	47483.511831	21432.801930	24763.906874	1.016e+004
g18	-0.865735	1828.373331	23801.181979	10637.993935	10979.150095	6.118e+003
g19	32.655593	32.655592	36.992855	34.016016	34.001811	1.307e+000
g20	0.096737	7822.895737	3917.001376	5716.608482	5553.804265	8.296e+002
g21	193.778349	6454.548967	494.249368	3023.426610	3026.544647	1.898e+003
g22	382.902205	-	-	-	-	-
g23	-400.002500	-747.026423	2958.256832	182.593228	441.421732	9.284e+002
g24	-5.508013	-5.580891	-5.999593	-5.854817	-5.838697	9.868e-002

Table 7: Comparisons of Best and Mean Values

Problems		Version 1	Version 2	Version 3	Version 4
g01	Best	-14.999339	**-15.000284**	-14.536362	**-15.000001**
	Mean	-12.969465	**-15.000255**	-13.576798	**-14.839069**
g02	Best	-0.802628	**-0.803558**	-0.803096	-0.802444
	Mean	-0.801371	**-0.802732**	-0.801542	-0.801357
g03	Best	-99675.835134	-97628.264744	-94725.828501	-78856.859004
	Mean	-99689.240788	-97829.864487	-95640.658296	-81291.317782
g04	Best	-32196.152588	-32172.599949	-32153.561671	-32168.512026
	Mean	-32192.275678	-32166.981986	-32121.669005	-32079.515834
g05	Best	1362.819559	**5126.449507**	1091.155898	62092.254197
	Mean	1081.791600	**5126.430191**	680.857312	21944.880460
g06	Best	-7962.000000	-7945.007066	-7923.373172	-7920.674774
	Mean	-7944.403151	-7943.096876	-7897.210547	-7879.538177
g07	Best	**24.306209**	24.301221	**24.306209**	24.306205
	Mean	**24.321855**	24.301923	41.412885	25.216320
g08	Best	-1296.806712	-59.518724	-32.344341	-24.963049
	Mean	-1348.735625	-192.699596	-710.303801	-239.138354
g09	Best	**680.630057**	680.627899	**680.630057**	680.630049
	Mean	**680.630057**	680.627328	**680.630057**	680.630044
g10	Best	2131.846129	2145.939349	2272.958937	2537.690323
	Mean	2140.683593	2157.301978	2500.159555	2775.485299
g11	Best	0.443553	0.350202	0.252099	0.217363
	Mean	0.440157	0.336515	0.237485	0.204685
g12	Best	**-1.000000**	**-1.000000**	**-1.000000**	**-1.000000**
	Mean	**-1.000000**	**-1.000000**	**-1.000000**	**-1.000000**
g13	Best	1.201125	0.068077	0.736388	0.645049
	Mean	1.837891	0.381841	1.238076	2.301202
g14	Best	-1420.02254	-32.497629	-1444.352863	23.263266
	Mean	-1561.62470	319.366876	-1570.498290	-100.020889
g15	Best	771.641627	**961.711596**	691.913649	973.457666
	Mean	709.556725	**961.710500**	615.130240	965.384840
g16	Best	**-1.905155**	-1.905389	**-1.905155**	-1.905156
	Mean	**-1.905155**	-1.905366	**-1.905155**	-1.905156
g17	Best	453.620550	**8852.251703**	830.311696	12611.343529
	Mean	1475.586258	**8888.782128**	1417.665163	24763.906874
g18	Best	32.185920	**-0.866058**	14.012411	1828.373331
	Mean	87.105215	**-0.858339**	46.079484	10979.150095
g19	Best	**32.655593**	32.655263	**32.655593**	32.655592
	Mean	**33.110947**	33.246513	**33.503592**	34.001811
g20	Best	99.408663	13904.386462	56.510557	7822.895737
	Mean	88.649597	11047.776752	50.586572	5553.804265
g21	Best	475.884809	304.126717	313.537859	6454.548967
	Mean	345.569134	194.813211	207.261941	3026.544647
g23	Best	-3022.66568	-1019.367279	-2804.065489	-747.026423
	Mean	-2458.25713	-769.276861	-2097.655354	441.421732
g24	Best	**-5.508013**	-5.542029	-5.884972	-5.580891
	Mean	**-5.628632**	-5.640892	-5.965597	-5.838697

REFRENCES

[1] C. A. C. Coello, D . A. Van Veldhuizen and G. B. Lamont. (2nd.). *Evolutionary Algorithms for Solving Multi-Objective Problems.* Norwell: Springer US, (2002)

[2] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Berlin, Germany: Springer-Verlag, (2003)

[3] C. A. C. Coello. Theoretical and numerical constraint handling techniques used with evolutionary algorithms: A survey of the state of the art. Comput. Methods Appl. Mech. Eng., 191, 1245-1287, (2002).

[4] Z. Michalewicz and M. Schoenauer. Evolutionary algorithm for constrained parameter optimization problems. Evol. Comput., 1-32, (1996).

[5] Z. Michalewicz, K. Deb, M. Schmidt and T. Stidsen. Test-case generator for nonlinear continuous parameter optimization techniques, IEEE Trans. Evol. Comput., 197-215, (2000).

[6] R. Storn and K.V. Price. Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces. J. Global Opim., 11, 341-359, (1997).

[7] J. Zhang and A.C. Sanderson. JADE: Adaptive Differential Evolution with Optional External

Archive. IEEE Congr. Evol. Comput., 13, 5, 945-958, (2009).

[8] J. J. Liang, T. P. Runarssaon and P.N. Suganthan. Problems definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization, Journal of Applied Mechanics, technical report, 41, (2006).

[9] R. Courant. Variational methods for the solution of problems of equilibrium and vibrations. Bull. Am. Math. Soc., 49, 1-23, (1943).

[10] C.W. Carroll. The created response surface technique for optimizing nonlinear restrained systems. Operations Research, 169-184, (1961).

[11] A.V. Fiacco, G.P. McCormick. Extensions of SUMT for nonlinear programming: Equality constraints and extrapolation. Manage. Sci.12, (11), 816-828, (1968).

[12] J. Joines and C. Houck. On the use of non-stationary penalty functions to solve non-linear con-strained optimization problems with Gas. Proceedings of the First IEEE Conference on Evol. Comput., IEEE World Congress on Comput, Orlando, Fl, 2, 579-584, (1994)