2006 IEEE Congress on Evolutionary Computation
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

A Population-Based, Parent Centric Procedure for Constrained Real-Parameter Optimization

Ankur Sinha, Aravind Srinivasan and Kalyanmoy Deb
Kanpur Genetic Algorithms Laboratory (KanGAL)
Indian Institute of Technology Kanpur
Kanpur, PIN 208 016, India
Email: {ankursi,aravinds,deb}@iitk.ac.in

Abstract— **Despite the existence of a number of procedures for constrained real-parameter optimization using evolutionary algorithms, there is still the need for a systematic and unbiased comparison of different approaches on a carefully chosen set of test problems. In this paper, we suggest a parent centric procedure for constrained real-parameter optimization. The algorithm so developed is applied to a set of 24 test problems and the results are presented. The proposed procedure is able to find the exact optimum within the specified number of function evaluations for 22 of the 24 test problems. In the remaining two problems, the proposed algorithm shows steady progress towards the respective optima, but it was unable to solve within the specified number of evaluations. It is also noteworthy that the algorithm was able to find solutions, better than the ones specified in the original problem description (`http://www.ntu.edu.sg/home/EPNSugan/`) for a number of test problems.**

I. Introduction

This paper is written for the special session devoted to comparing different constrained real-parameter optimization methods on a set of 24 test problems. In this paper, we employ a population-based, steady-state optimization algorithm for the purpose. The algorithm is developed based on adaptation of a population-based algorithm-generator [1]. The generator requires specification of four plans of the optimization process: (i) selection plan, (ii) generation plan, (iii) replacement plan and (iv) update plan. These plans are designed based on essential aspects needed in solving various optimization problems, such as importance of diversity preservation and need for creation of offspring solutions based on diversity in parent solutions (also used in other successful real parameter optimization schemes, such as evolution strategy [2], [4] and differential evolution [5]).

The test problems involve equality and inequality constraints, linear and non-linear constraints, low to high dimensionalities etc. It is our intuition that to solve such wide variety of test problems to a reasonable level of satisfaction, the algorithm has to be simple and not specifically designed to solve a particular problem. In the following section, we describe the proposed procedure. In Section 5, we present the simulation results in tabular form and in Section 8, we discuss the performance of our algorithm on different test problems.

II. Description of the Algorithm

The optimization algorithm used here is derived from the population-based algorithm generator suggested elsewhere [1]. The algorithm-generator requires four plans to be specified and generates a steady-state optimization procedure:

1) Selection Plan (SP): Strategy used to select a fixed number of parents for recombination from the current population.
2) Generation Plan (GP): Methodology used to create offspring solutions from parents chosen in the selection plan.
3) Replacement Plan (RP): Strategy used to select a fixed number of members from population that will compete with the newly generated offspring solutions for inclusion in the population.
4) Update Plan (UP): Strategy used to decide the winners from a set consisting of offspring solutions and members obtained from replacement plan that will eventually get included in the current population.

The above division of an algorithm into different plans allows one to design each essential feature of an optimization task independently. With a set of population members, the first task (SP) is to choose a set of good solutions (parents) so that they can be utilized to create new solutions in GP. The use of a suitable probability distribution around parent solutions to create new offspring solutions would be one way to implement a GP. Once the offspring solutions are created they can be accepted in the fixed-size population by first choosing a set of possible population members for deletion using a RP and then designing a scheme for updating the population in UP. This is where an elite preserving strategy can be implemented.

The equality constraints ($h(\vec{x}) = 0$), as suggested, have been transformed into inequality constraints of the form $|h(\vec{x})| - \epsilon \leq 0$ and a solution is regarded as feasible if the following conditions are satisfied:

$$g_j(\vec{x}) \leq 0 \quad for \quad j = 1, 2, ..., q \tag{1}$$

$$|h(\vec{x})| - \epsilon \leq 0 \quad for \quad j = q+1, q+2, ..., m \tag{2}$$

It is well known that the equality constraints are tough to satisfy as they provide a narrow window to the algorithm to operate. To avoid this, equality constraints are handled

0-7803-9487-9/06/$20.00/©2006 IEEE

here in such a way that initially the size of the window is larger and slowly it narrows down to the required accuracy as the number of function evaluations increases. The following function has been used to define the ϵ value during intermediate function evaluations. This value is referred as ϵ_t where t denotes the number of function evaluations.

$$\epsilon_t = \epsilon(1 + 1000e^{-200(t/t_{max})}) \tag{3}$$

It is clear that when $t = t_{max}$ then $\epsilon_t = \epsilon$, so after maximum function evaluations, only those solutions would be feasible which satisfy the equlatiy constraint with an accuracy of ϵ. To begin with, the window would be larger and initially the algorithm would not have any problem in exploring the search space. The values in the above equation have been intutively adjusted just to meet the requirements but this can be replaced with even efficient parameters. By introducing this equation in the algorithm we are enhancing the exploration of the search space for problems having equality constraints and it makes the algorithm less susceptible to quickly loose its diversity or get caught in a local minima, hence increasing its reliability.

The constraint violation is the summation of the violations of all the equality and inequality constraints. A solution x^i is said to 'constraint dominate' [9] a solution x^j if any of the following conditions are true:

1) Solution x^i is feasible and solution x^j is not.
2) Solution x^i and x^j are both infeasible but solution x^i has a smaller constraint violation.
3) Solution x^i and x^j are both feasible but the objective value of x^i is less than that of x^j.

The algorithm starts with an initial population (generated randomly) of size N. In the *Selection Plan*, stochastic remainder selection [10] is used to form a pool from which *index parents* are selected for crossover. To create an offspring, a parent is chosen from the pool which becomes the index parent and the other two parents are chosen from the population randomly. We shall describe this crossover operator (PCX) a little later.

In stochastic remainder selection, the fitness assignment scheme is done in the following manner by keeping in mind the above three constraint handling principles:

1) If all the members of the population are infeasible then the fitness to a member x^i is assigned as

$$fitness_{x^i} = 1 - \frac{v_{x^i} - v_{min}}{v_{max} - v_{min}} \tag{4}$$

v_{x^i} = summation of the violations of member x^i
v_{min} = minimum violation among all the members
v_{max} = maximum violation among all the members

2) Otherwise
For an infeasible member

$$fitness_{x^i} = 1 - \frac{(t + v_{x^i}) - o_{min}}{o_{max} - o_{min}} \tag{5}$$

For a feasible member

$$fitness_{x^i} = 1 - \frac{o_{x^i} - o_{min}}{o_{max} - o_{min}} \tag{6}$$

t = worst objective value among feasible members
o_{min} = best (minimum) objective value among feasible members.
o_{max} = worst (maximum) objective value among feasible members.
o_{x^i} = objective value of the member x^i.

The above fitness evalation scheme assigns a higher value to a better solution.

After these fitness values are computed, they are degraded by a niching procedure which does not use any niching parameter. The fitness degradation scheme is identical to that of the sharing function method [12], except that the niche radius σ_{share} is not a problem parameter, rather is set by an adaptive procedure. The extent of the search space dictated by the population-minimum and maximum values is divided into exactly N small hypercubes and the size of the hypercube is defined as σ_{share}.

In the *Generation Plan*, we create an offspring solution from the chosen parent solutions. We use the parent-centric recombination (PCX) operator [6] with modification for the purpose of recombination and produce an offspring. Here a planar version of PCX operator has been used, where 3 parents are used to create a child on the plane formed by the three parents. The following is the recombination operator which is used to create a child from the parents.

$$\vec{C} = \vec{X_p} + \omega_\xi \vec{D} + \omega_\eta \frac{\vec{P_2} - \vec{P_1}}{2} \tag{7}$$

The terms used in the above equation are defined as follows:

- $\vec{X_p} \in \mathcal{B}$ is *index* parent
- $\vec{G}$ is the mean of μ parents
- $\vec{D} = \vec{X_p} - \vec{G}$
- $\vec{P_1}$ and $\vec{P_2}$ are parents chosen randomly from the population
- ω_ξ and ω_η are the two parameters which are updated by the one-fifth success rule [11]. It starts with unity and its value is decreased by a factor $c_d = 0.817$ if the success rate in the generation is less than $1/5$ otherwise it is increased by a factor $c_d^{-1} = 1/0.817$. If the value of the parameters become less than 0.5 then it is again assigned a value of 1.0.

In each generation λ children are created. After the generation plan, next we use the *Replacement Plan* to choose r solutions from the population. In the present scheme, we choose these solutions randomly from the entire population. We then form a pool (of size $\lambda + r$) consisting of r solutions chosen from the population by the replacement plan and λ newly created offspring solutions by the generation plan. The current population is then updated using the *Update Plan*, in

which we replace the r solutions chosen in the replacement plan by the best r solutions of the pool. This operation ensures an elite-preservation strategy. The best ones are chosen by the constraint domination principle described above.

In performing a single generation of above mentioned procedure, we have exhausted λ function evaluations (same as the number of offspring solutions produced). The iteration continues until a prescribed number of function evaluations is achieved or a pre-defined termination criterion is met. We use the polynomial mutation [8] as the mutation operator with a mutation probability $p_m = 1/n$, where n is the number of real variables.

III. PC Configuration

- System: RedHat Linux 9.0 (i386 GNU/Linux)
- CPU: P-III 1.0 GHz
- RAM: 256 MB
- Language: ANSI-C
- Compiler Used: GCC version-3.2.2

IV. Parameter Setting

1) Population Size: N = 100. For problem 2, 100 population members does not provide a high success rate , although a steady progress towards the optimum is observed. Here, for problem 2, we use $N = 200$.
2) Number of offsprings created in each generation: λ = 10
3) Number of parent solutions selected randomly for replacement: r = 2

V. Results Obtained

The results achieved have been presented in Tables I, II, III, IV, and V. The first four tables show the function error values, $(f(\vec{x}) - f(\vec{x^*}))$, for all the functions at 5×10^3, 5×10^4, and 5×10^5 function evaluations. 25 runs were done for each test problem and the best, median and worst results have been presented along with the mean value and the standard deviation. In the table, c represents the number of violated constraints at the median solution. The sequence of three numbers indicate the number of constraint violations greater than 1.0, 0.01 and 0.0001 respectively. $\overline{v}$ is the mean value of the violations of all the constraints at the median solution. The numbers in the parenthesis after the error value of the best, median and worst solution indicates the number of constraints which are not satisfied at the corresponding location.

Tables I, II, III, and IV show some negative error values which means that the objective value was better than the best value because of the violation of the constraints. Moreover in some of the problems the best value obtained is better than the values suggested in the special session. This would make the final error values negative. But to avoid confusion the final errors have been reported as 0.0000 in those cases.

Table V shows the number of function evaluations needed to achieve the fixed accuracy level ($(f(\vec{x}) - f(\vec{x^*})) < 0.0001$), success rate, feasibile rate and success performance. The mean and the standard deviation for the number of function evaluations has also been presented.

VI. Convergence Map

The convergence plots for the median run of the total runs and with a termination criterion of maximum function evaluations (500,000) for each test problem are shown in Figures 1, 2, 3 and 4. The semi-log graphs show $\log_{10}((f(\vec{x}) - f(\vec{x}^*)))$ vs FES and $\log_{10}(\overline{v})$ vs FES for each problem. Points, with error values very close to zero or negative, could not be shown in the graph as the logarithm of the function is not defined for values less than or equal to zero. Problem 19, where a feasible solution was obtained while generating the initial population does not appear in the $\log_{10}(\overline{v})$ vs FES graph as $\overline{v}$ is always zero in such a case and its logarithm could not be produced. For other problems also the graph ends as soon as the error value becomes zero.

VII. Discussion of Results

Objective values better than the ones provided have been obtained in problems 3, 4, 5, 6, 8, 10, 11, 13, 14, 15, 16, 17, 18, 19, 21, 23, and 24. In case of the problems having equality constraints this can be attributed to the ϵ window which has been provided to the algorithm to operate. Out of the above mentioned 17 problems the problems having equality constraints are 3, 5, 11, 13, 14, 15, 21, and 23. On all other problems the algorithm has been able to produce results better than that available.

Relatively high values of ω_ξ and ω_η have been taken in order to span the entire search space. Smaller values would ensure a faster convergence. The parameters used in the algorithm are very simple to understand and can be effectively put to use. However, default parameters have been used for the entire results presented here and the algorithm is able to successfully solve the test problems with a good success rate.

VIII. Algorithm Complexity

Table VI shows the complexity of the algorithm. Here $T0$ is the time (in milli-seconds) taken to execute the provided sample program. $T1$ is the average computing time (in milli-seconds) per problem for $10,000$ evaluations of the given 24 test problems. $T2$ is the average computing time (in milli-seconds) per problem for the algorithm with 10,000 evaluations for the given 24 test problems.

IX. Conclusions

In this paper we have presented a parent centric [1], population based procedure for constrained optimization. 24 different benchmark test problems have been tried and the algorithm has been successful in solving most of the problems. The algorithm has been developed with a simple approach which makes it robust to handle a wide variety of problems. Extensive simulation results have demonstrated that the algorithm has been successful in attaining the desired

FES		g01	g02	g03	g04	g05	g06
5×10^3	Best	5.7195(0)	0.4251(0)	0.0219(1)	41.7102(0)	0.3639(3)	63.7697(0)
	Median	7.5600(0)	0.5044(0)	0.3165(1)	124.3401(0)	50.0966(3)	451.0048(0)
	Worst	8.9692(0)	0.5628(0)	0.6794(1)	287.8847(0)	920.5715(3)	1055.1352(0)
	c	0,0,0	0,0,0	0,0,1	0,0,0	0,3,4	0,0,0
	$\overline{v}$	0.0000	0.0000	0.0099	0.0000	4.1336	0.0000
	Mean	7.6496	0.5046	0.3044	131.9307	175.0356	481.6039
	Std	0.9145	0.0303	0.1569	58.3098	251.4995	225.1081
5×10^4	Best	0.0000(0)	0.2166(0)	0.0000(0)	0.0000(0)	0.0000(0)	0.0000(0)
	Median	0.0000(0)	0.3103(0)	0.0000(0)	0.0000(0)	0.0000(0)	0.0000(0)
	Worst	0.0136(0)	0.4828(0)	0.0000(0)	0.0000(0)	109.6522(3)	0.0000(0)
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	$\overline{v}$	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	Mean	0.0012	0.3095	0.0000	0.0000	7.4733	0.0000
	Std	0.0003	0.0292	0.0000	0.0000	9.2474	0.0000
5×10^5	Best	0.0000(0)	0.0000(0)	0.0000(0)	0.0000(0)	0.0000(0)	0.0000(0)
	Median	0.0000(0)	0.0000(0)	0.0000(0)	0.0000(0)	0.0000(0)	0.0000(0)
	Worst	0.0000(0)	0.0712(0)	0.0000(0)	0.0000(0)	0.0000(0)	0.0000(0)
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	$\overline{v}$	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	Mean	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	Std	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

TABLE I

ERROR VALUES ACHIEVED WHEN FES= 5×10^3 , FES= 5×10^4, FES= 5×10^5 FOR PROBLEMS 1-6.

FES		g07	g08	g09	g10	g11	g12
5×10^3	Best	103.3952(0)	0.0000(0)	5.5400(0)	1582.5174(0)	-0.0101(1)	0.0000(0)
	Median	429.1698(0)	0.0000(0)	20.2502(0)	4401.9779(0)	-0.0098(1)	0.0065(0)
	Worst	1022.1652(0)	0.0000(0)	58.5712(0)	9184.8916(0)	-0.0023(1)	0.0195(0)
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,1	0,0,0
	$\overline{v}$	0.0000	0.0000	0.0000	0.0000	0.0098	0.0000
	Mean	400.0089	0.0000	23.5748	4516.2436	-0.0090	0.0053
	Std	235.2697	0.0000	14.1171	1619.6275	0.0019	0.0056
5×10^4	Best	0.8331(0)	0.0000(0)	0.0000(0)	4.1367(0)	0.0000(0)	0.0000(0)
	Median	1.2676(0)	0.0000(0)	0.0000(0)	21.1173(0)	0.0000(0)	0.0000(0)
	Worst	2.0361(0)	0.0000(0)	0.0018(0)	162.5342(0)	0.0000(0)	0.0000(0)
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	$\overline{v}$	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	Mean	1.2620	0.0000	0.0001	35.3208	0.0000	0.0000
	Std	0.3047	0.0000	0.0001	37.8958	0.0000	0.0000
5×10^5	Best	0.0000(0)	0.0000(0)	0.0000(0)	0.0000(0)	0.0000(0)	0.0000(0)
	Median	0.0000(0)	0.0000(0)	0.0000(0)	0.0000(0)	0.0000(0)	0.0000(0)
	Worst	0.0000(0)	0.0000(0)	0.0000(0)	0.0000(0)	0.0000(0)	0.0000(0)
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	$\overline{v}$	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	Mean	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	Std	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

TABLE II

ERROR VALUES ACHIEVED WHEN FES= 5×10^3 , FES= 5×10^4, FES= 5×10^5 FOR PROBLEMS 7-12.

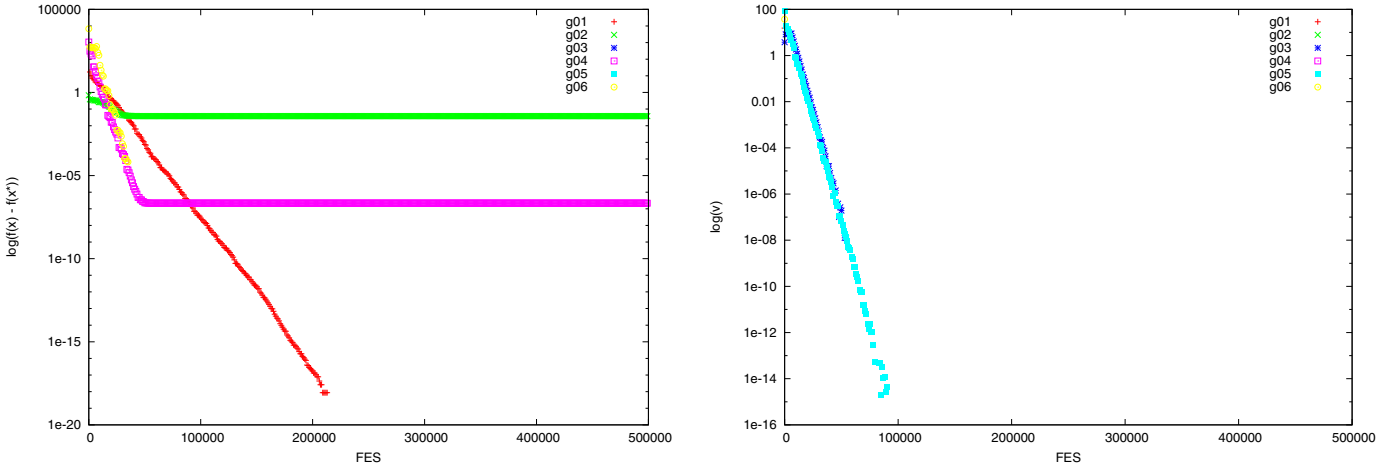

Fig. 1. Convergence Graph for Problems 1-6

FES		g13	g14	g15	g16	g17	g18
5×10^3	Best	0.0382(3)	-272.4364(3)	-0.0067(2)	0.0356(0)	-14.2485(4)	0.2479(0)
	Median	0.9251(3)	-179.3711(3)	1.0686(2)	0.0911(0)	70.8509(4)	0.4556(0)
	Worst	2.7294(3)	-134.7809(3)	6.6074(2)	0.1879(0)	355.9080(4)	0.8444(0)
	c	0,3,3	3,3,3	0,0,2	0,0,0	3,4,4	0,0,0
	$\overline{v}$	0.1069	5.9201	-0.0080	0.0000	0.6642	0.0000
	Mean	1.0219	-188.0254	2.2583	0.1012	114.6437	0.4892
	Std	0.6173	36.3780	2.3592	0.0421	108.3202	0.1471
5×10^4	Best	0.0000(0)	0.0000(0)	0.0000(0)	0.0000(0)	-16.6664(3)	0.0006(0)
	Median	0.7572(3)	1.9586(2)	0.0000(0)	0.0001(0)	66.6311(4)	0.0033(0)
	Worst	0.9337(3)	3.1329(3)	2.5347(2)	0.0002(0)	287.4956(4)	0.1997(0)
	c	0,0,3	0,0,3	0,0,0	0,0,0	0,0,4	0,0,0
	$\overline{v}$	0.0005	0.0071	0.0095	0.0000	0.0014	0.0000
	Mean	0.7100	1.9482	0.5002	0.0001	80.8104	0.0490
	Std	0.1730	0.7955	0.7962	0.0000	81.9456	0.0837
5×10^5	Best	0.0000(0)	0.0000(0)	0.0000(0)	0.0000(0)	0.0000(0)	0.0000(0)
	Median	0.0000(0)	0.0000(0)	0.0000(0)	0.0000(0)	0.0000(0)	0.0000(0)
	Worst	0.0000(0)	0.0000(0)	0.0000(0)	0.0000(0)	0.0000(0)	0.0000(0)
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	$\overline{v}$	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	Mean	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	Std	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

TABLE III

ERROR VALUES ACHIEVED WHEN FES= 5×10^3 , FES= 5×10^4, FES= 5×10^5 FOR PROBLEMS 13-18.

FES		g19	g20	g21	g22	g23	g24
5×10^3	Best	493.9014(0)	4.9451(20)	228.1415(5)	554.1268(19)	-81.8485(4)	0.0000(0)
	Median	848.0931(0)	8.3783(20)	475.8850(5)	8068.0194(19)	288.1156(4)	0.0030(0)
	Worst	1494.5449(0)	11.6850(20)	790.2094(5)	18245.1313(19)	590.7865(5)	0.0064(0)
	c	0,0,0	2,12,20	0,4,4	19,19,19	0,4,4	0,0,0
	$\overline{v}$	0.0000	54.9031	4.0066	519530.3560	8.0524	0.0000
	Mean	875.8381	8.4831	479.5546	8264.8937	255.5581	0.0030
	Std	252.0896	1.7936	159.3575	5337.2902	194.1190	0.0013
5×10^4	Best	15.6681(0)	1.1085(19)	0.0000(0)	2866.4223(19)	36.7242(4)	0.0000(0)
	Median	32.3321(0)	2.1397(20)	0.0000(0)	10146.2777(19)	145.5710(4)	0.0000(0)
	Worst	59.2246(0)	4.8243(20)	0.0000(0)	18659.9288(19)	386.9674(4)	0.0000(0)
	c	0,0,0	0,5,18	0,0,0	17,19,19	0,0,4	0,0,0
	$\overline{v}$	0.0000	48.9721	0.0000	12.4385	8.0050	0.0000
	Mean	33.1020	2.4949	0.0000	10560.8451	191.4602	0.0000
	Std	10.7445	1.0806	0.0000	4392.4907	136.0416	0.0000
5×10^5	Best	0.0000(0)	0.0313(9)	0.0000(0)	2191.7665(13)	0.0000(0)	0.0000(0)
	Median	0.0000(0)	0.0675(12)	0.0000(0)	9888.6409(15)	0.0000(0)	0.0000(0)
	Worst	0.0000(0)	0.1227(10)	0.0000(0)	18476.3220(11)	0.0000(0)	0.0000(0)
	c	0,0,0	0,2,7	0,0,0	5,6,6	0,0,0	0,0,0
	$\overline{v}$	0.0000	0.0289	0.0000	11.1027	0.0000	0.0000
	Mean	0.0000	0.0691	0.0000	10305.7552	0.0000	0.0000
	Std	0.0000	0.0219	0.0000	4421.5326	0.0000	0.0000

TABLE IV

ERROR VALUES ACHIEVED WHEN FES= 5×10^3 , FES= 5×10^4, FES= 5×10^5 FOR PROBLEMS 19-24.

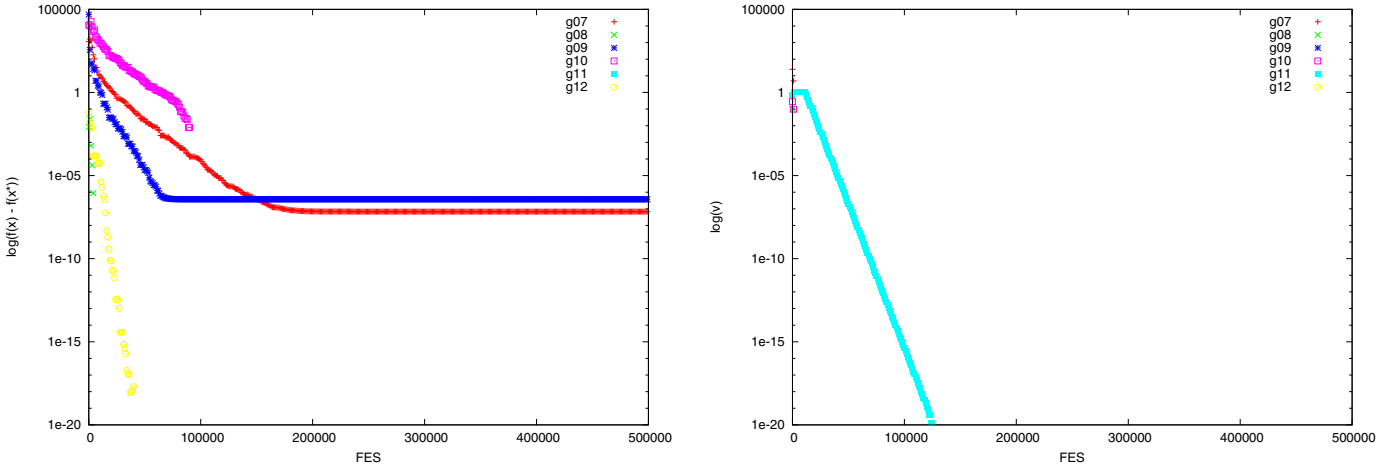

Fig. 2. Convergence Graph for Problems 7-12

TABLE V

NUMBER OF FES REQUIRED TO ACHIEVE A GIVEN ACCURACY LEVEL FOR PROBLEMS $1 - 24$

Prob	1^{st}	13^{th}	25^{th}	Mean	Std	Feas. Rate	Succ. Rate	Succ. Perf
g1	32420	47340	62026	55204.44	3420.82	100	100	55204.44
g2	52900	88463	500000	87900.00	12310.24	100	64	127900.00
g3	28330	33590	47300	34937.00	2202.33	100	100	34937.00
g4	25310	31270	40140	30989.00	1893.00	100	100	30989.00
g5	29740	84740	125088	94765.00	7568.73	100	100	94765.00
g6	32560	33670	36180	33821.00	77.00	100	100	33821.00
g7	72920	91550	258840	117121.00	7733.66	100	100	117121.00
g8	1710	3030	3510	2826.00	228.00	100	100	2826.00
g9	32100	44400	58700	46527.00	4057.66	100	100	46527.00
g10	57770	92180	109970	89028.00	1437.33	100	100	89028.00
g11	18540	31220	46620	38688.00	2727.33	100	100	38688.00
g12	3710	8650	11940	8960.00	640.00	100	100	8960.00
g13	34310	45948	77946	53735.00	5425.00	100	100	53735.00
g14	37900	58230	84340	59237.00	925.66	100	100	59237.00
g15	27550	39400	91400	46936.00	6462.00	100	100	46936.00
g16	19130	30680	36790	30395.00	2451.66	100	100	30395.00
g17	88410	129483	188072	136110.00	36700.00	100	100	136110.00
g18	48350	62350	96180	70027.00	4559.00	100	100	70027.00
g19	75210	145762	187734	129676.00	5908.00	100	100	129676.00
g20	500000	500000	500000	-	-	0	0	-
g21	31150	40850	46784	38216.66	426.34	100	100	38216.66
g22	500000	500000	500000	-	-	0	0	-
g23	66330	157420	221464	167118.88	6533.27	100	100	167118.88
g24	4800	9800	13690	11646.00	681.33	100	100	11646.00

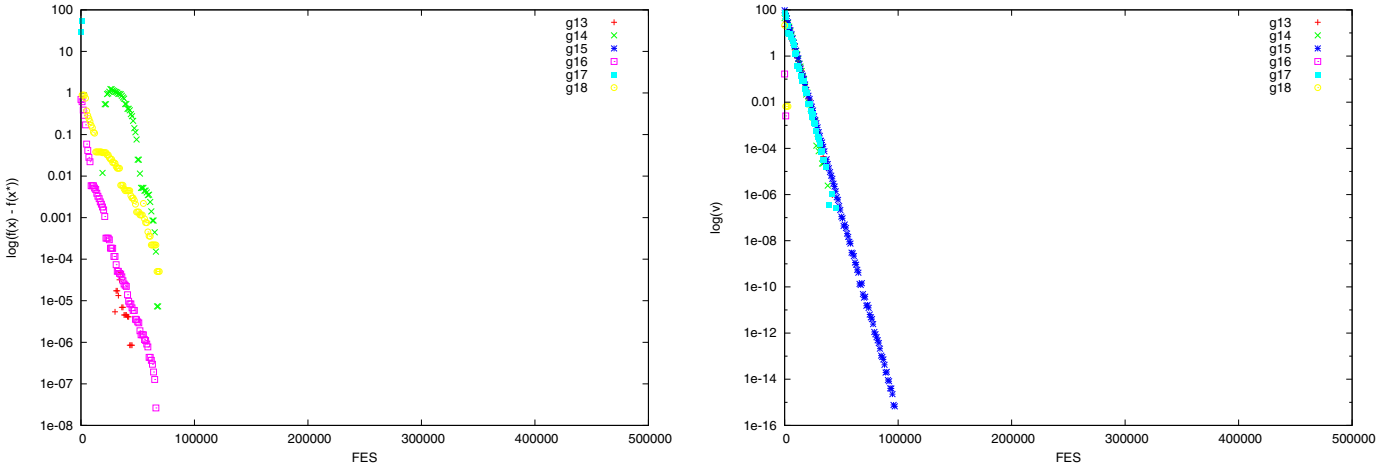

Fig. 3. Convergence Graph for Problems 13-18

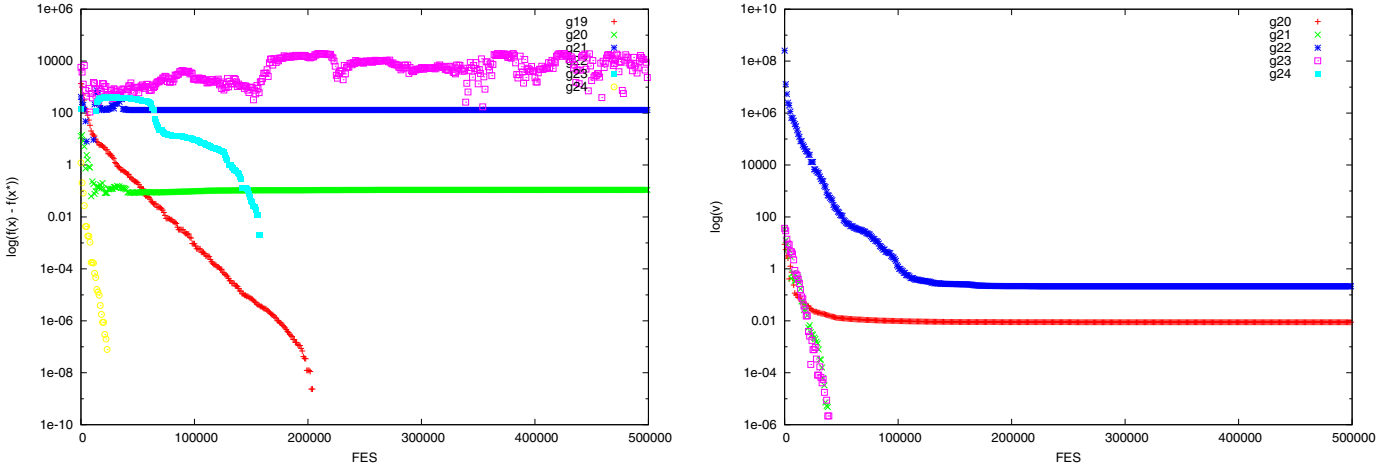

Fig. 4. Convergence Graph for Problems 19-24

$T0$	$T1$	$T2$	$(T2-T1)/T0$
500	23	1216	2.386

TABLE VI
COMPUTATIONAL COMPLEXITY (TIME IN MILLISECONDS)

accuracy in most of the problems. In the problems where desired accuracy was not met the algorithm showed a steady improvement in the objective function value. The algorithm also turns out to be reliable in most of the cases with a 100% success rate.

REFERENCES

[1] K. Deb. A population-based algorithm-generator for real-parameter optimization. *Soft Computing*, in press.

[2] N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strageties: The covariance matrix adaptation. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 312–317, 1996.

[3] A. Sinha, S. Tiwari, and K. Deb A Population-Based, Steady-State Procedure for Real-Parameter Optimization. In *Proceedings of the 2002 Congress on Evolutionary Computation (IEEE CEC-02)*, Honolulu, HI, USA, 2002.

[4] H.-P. Schwefel. Collective intelligence in evolving systems. In W. Wolff, C. J. Soeder, and F. Drepper, editors, *Ecodynamics – Contributions to Theoretical Ecology*, pages 95–100. Berlin: Springer, 1987.

[5] R. Storn and K. Price. Differential evolution – A fast and effi cient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, 1997.

[6] K. Deb, D. Joshi, and A. Anand. Real-coded evolutionary algorithms with parent-centric recombination. In *Proceedings of the 2002 Congress on Evolutionary Computation (IEEE CEC-02)*, Honolulu, HI, USA, 2002.

[7] L. Eshelman. The CHC adaptive search algorithm. how to have safe search when engaging in nontraditional genetic recombination. *Foundations of Genetic Algorithms*, pages 265–283, 1991.

[8] K. Deb and M. Goyal. A combined genetic adaptive search (GeneAS) for engineering design. *Computer Science and Informatics*, 26(4):30–45, 1996.

[9] K. Deb An effi cient constraint handling method for Genetic Algorithms. *Computer Methods in Applied Mechanics and Engineering*, 2000

[10] K. Deb *Multi-objective optimization using evolutionary algorithms* Chichester, UK: Wiley, 2001.

[11] T. Back, U. Hammel and H. Schwefel *IEEE Transaction on Evolutionary Computation*, Vol. 1. No. 1, pages 3-17, New York: IEEE, 1997.

[12] D.E. Goldberg and J. Richardson Genetic algorithm with sharing for multimodal function optimization. *Proceedings 2nd International Conference Genetic Algorithms and Applications.*, Cambridge, MA, July 1987, pages 41–49