ADAPTIVE PENALTY FUNCTION FOR SOLVING CONSTRAINED EVOLUTIONARY OPTIMIZATION

[1]Omar Al Jadaan, [2]Lakshmi Rajamani, [3]C. R. Rao

[1]Dept. CSE, EC. Osmania University, Hyderabad 500-007, India.

[2]Prof. Dept. CSE, EC. Osmania University, Hyderabad 500-007, India.

[3]Prof. Dept. CSE, EC. Osmania University, Hyderabad 500-007, India.

E-mail: o_jadaan@yahoo.com, lakshmiraja@yahoo.com, crrcs@uohyd.ernet.in

ABSTRACT

A criticism of Evolutionary Algorithms might be the lack of efficient and robust generic methods to handle constraints. The most widespread approach for constrained search problems is to use penalty methods, because of their simplicity and ease of implementation. The penalty function approach is generic and applicable to any type of constraint (linear or nonlinear). Nonetheless, the most difficult aspect of the penalty function approach is to find an appropriate penalty parameters needed to guide the search towards the constrained optimum. In this paper, GA's population-based approach and Ranks are exploited to devise a penalty function approach that does not require any penalty parameter called Adaptive GA-RRWS. Adaptive penalty parameters assignments among feasible and infeasible solutions are made with a view to provide a search direction towards the feasible region. Rank-based Roulette Wheel selection operator (RRWS) is used. The new adaptive penalty and rank-based roulette wheel selection operator allow GA's to continuously find better feasible solutions, gradually leading the search near the true optimum solution. GAs with this constraint handling approach have been tested on thirteen problems commonly used in the literature. In all cases, the proposed approach has been able to repeatedly find solutions closer to the true optimum solution than that reported earlier.

Keywords: *Constrained Optimization, Constraint Handling, Genetic Algorithm, Penalty Functions, Ranking.*

1. INTRODUCTION

The general nonlinear programming (*A*) problem can be formulated as solving the *objective* function

$$\underset{x \in S \cup F \subseteq \Re n}{Min} \{f(x)\} \quad (1)$$

Where $s = \{x \in \Re^n; x_i^l \le x_i \le x_i^u, (i = 1,...,n)\}$, defines the *search space* which is a *n*-dimensional space bounded by the parametric constraints

$$x_i^l \le x_i \le x_i^u, (i = 1,...,n) \quad (2)$$

and the *feasible region* F is defined by

$$F = \left\{x \in \Re^n \mid g_j(x) \le 0, \forall j \in \{1,...,m\}\right\} \quad (3)$$

Where $g_j(x), j \in \{1,...m\}$ are constraints, which include all equality constraints after transforming them to inequality constraints using (4)

$$|h(x)| - \varepsilon \le 0 \quad (4)$$

Where ε is a small tolerance. Since the algorithm that will be discussed does not use gradient information, it does not mater if equality constraint (4) is non-differentiable.

Constraint handling methods used in classical optimization algorithms can be classified into two groups: (i) generic methods that do not exploit the mathematical structure (whether linear or nonlinear) of the constraint, and (ii) specific methods that are only applicable to a special type of constraints. Generic methods, such as the penalty function method, the Lagrange multiplier method, and the complex search method [3], [7] are popular because each one of them can be easily applied to most problems without much change in the algorithm. Nevertheless, since these methods are generic, the performance of these methods in most cases is not satisfactory. However, specific methods, such as the cutting plane method, the reduced gradient method, and the gradient projection method [3], [7], are applicable either to problems having convex feasible regions only or to problems having a few variables because of increased computational burden with large number of variables. Since genetic algorithms (GAs) are generic search methods, most applications of GAs

to constraint optimization problems have used the penalty function approach of handling constraints [8], [6]. The penalty function approach involves a number of penalty parameters, which must be set right in any problem to obtain feasible solutions. This dependency of GA's performance on penalty parameters has led researchers to devise sophisticated penalty function approaches such as multi-level penalty functions [1], dynamic penalty functions [9], and penalty functions involving temperature-based evolution of penalty parameters with repair operators [24]. All these approaches require extensive experimentation for setting up appropriate parameters needed to define the penalty function. Michalewicz describes the difficulties in each method and compares the performance of these algorithms on a number of test problems [14]. In a similar study, Michalewicz and Schoenauer concluded that the static penalty function method (Without any sophistication) is a more robust approach than the sophisticated methods [23]. This is because one such sophisticated method may work well on some problems but may not work so well in another problem.

The introduction of the penalty term enables us to transform a constrained optimization problem (A) into an unconstrained one (A'), such as the one given by (5):

$$\underset{x \in S \subseteq \Re^n}{Min} \{\psi(y)\}; \psi(y) = f(x) + r_g \phi\left(g_j(x), j = 1, ..., m\right) \quad (5)$$

Where $\phi \geq 0$ is a real-valued function which imposes a penalty controlled by a sequence of *penalty coefficients* r_g, where g is the generation counter. The general form of ϕ function includes both the generation counter (for dynamic penalty) and the population (for adaptive penalty). In the current notation, this is reflected in the penalty coefficient r_g.

This transformation (i.e.(5)) has been used widely in evolutionary constrained optimization [11], [20].

The penalty function method may work quite well for some problems; however, deciding an optimal (or near-optimal) value for r_g turns out to be a difficult optimization problem itself! If r_g is too small, an infeasible solution may not be penalized enough. Hence, an infeasible solution may be evolved by an evolutionary algorithm. If r_g is too large, a feasible solution is very likely to be found, but could be of very poor quality. A large r_g discourages the exploration of infeasible regions, even in the early stages of evolution. This is particularly inefficient for problems where feasible regions in the whole search space are disjointed. In this case, it may be difficult for an evolutionary algorithm to move from one feasible region to another unless they are very close to each other. Reasonable exploration of infeasible regions may act as bridges connecting two or more different feasible regions. The critical issue here is how much exploration of infeasible regions (i.e., how large r_g is) should be considered as reasonable.

The answer to this question is problem dependent. Even for the same problem, different stages of evolutionary search may require different r_g values.

There has been some work on the dynamic setting of r_g values in evolutionary constrained optimization [10], [11], and [15].Such work usually relies on a predefined monotonically nondecreasing sequence of r_g values. This approach worked well for some simple problems, but failed for more difficult ones because the optimal setting of r_g values is problem dependent [18]. A fixed and predefined sequence cannot treat a variety of different problems satisfactorily. In such cases trial-and-error process has to be used in order to find a proper function for r_g as is done in [10], [11]. An adaptive approach, where r_g values are adjusted dynamically and automatically by an evolutionary algorithm itself, appears to be most promising in tackling different constrained optimization problems. For example, population information can be used to adjust r_g values adaptively [21]. Different problems lead to different populations in evolutionary search, and thus lead to different r_g values. The advantage of such an adaptive approach is that it can be applied to problems where little prior knowledge is available because there is no need to find a predefined r_g value (or a sequence of r_g values) that is optimal for this problem. According to (5), different r_g values define different fitness functions. A fit individual under one fitness function may not be fit under a different fitness function. Finding a near-optimal r_g adaptively is equivalent to ranking individuals adaptively in a population. Hence, the issue becomes how to rank individuals according to their objective and penalty values.

A novel method for ranking individuals without specifying an r_g value is proposed. Experimental studies test the effectiveness and efficiency of this method which can be regarded as an adaptive penalty approach.

One approach to avoid setting a hard-to-set parameter r_g is to treat constrained optimization as a multi-objective optimization where constraints are regarded as an additional objective function [22], [2]. However, multi-objective optimization does not appear to be any easier than constrained optimization since one has to balance different objectives in optimization. The rest of this paper is organized as follows. Section II discusses the proposed constraint handling method, where relationship between r_g and ranking are described in more details. The concept of dominance is introduced, which is somewhat similar to, but not the same as an early work [20].The analysis of penalty methods from the point of view of balancing dominance between the objective and penalty functions has revealed what penalty methods are trying to do, and has led to the development of this new constraint handling technique which balances such dominance directly and implicitly in order to improve the effectiveness and efficiency of constrained algorithms. Section III describes the details of the implementation of the evolutionary algorithm for constrained optimization, and presents the experimental results on 13 benchmark problems. The results are also compared with best-known solutions obtained using earlier Evolutionary Algorithms implementations. Finally, Section IV ends up with conclusions and hints at future work.

2. PROPOSED CONSTRAINT HANDLING METHOD

A. Penalty Method

For a given penalty coefficient $r_g \geq 0$ let the ranking of *n* individuals be

$$\psi(y_1) \leq \psi(y_2) \leq ... \leq \psi(y_n) \qquad (6)$$

Where $\psi(y_i); i = 1,2,...,n$ is the transformation function given by (5) for a set of *n* individuals. Let us examine the adjacent pair *i* and *i* + 1 in the ranked order

$$f_i + r_g \phi_i \leq f_{i+1} + r_g \phi_i + 1, i \in \{1,...,n-1\} \quad (7)$$

Where the notations $f_i = f(x_i)$ and $\phi_i = \phi\left(g_i\left(x_i\right), j = (1,...,m)\right)$ are used for convenience. We now introduce a parameter $r_c(i)$ Which will be referred to as the *critical penalty coefficient* for the adjacent pair *i* and *i* + 1

$$r_c(i) = \frac{f_{i+1} - f_i}{\phi_{i+1} - \phi_i} \text{ For } \phi_i \neq \phi_{i+1} \qquad (8)$$

For the given choice of $r_g \geq 0$, there are three different cases, which may give rise to the inequality (7).

1) If $f_i \leq f_{i+1}$ an $\phi_i \geq \phi_{i+1}$: The comparison is said to be *dominated by the objective function* and $0 \leq r_g \leq r_c(i)$ because the objective function f plays the dominant role in determining the inequality. When individuals are feasible, $\phi_i = \phi_{i+1}$ and $r_c(i) \to \infty$.
2) If $f_i \geq f_{i+1}$ and $\phi_i < \phi_{i+1}$: The comparison is said to be *dominated by the penalty function* and $0 \leq r_c(i) \leq r_g$ because the penalty function ϕ plays the dominant role in determining the inequality.
3) If $f_i < f_{i+1}$ and $\phi_i < \phi_{i+1}$: The comparison is said to be non-dominated and $r_c(i) < 0$, neither the objective nor the penalty function can determine the inequality by itself.

When comparing nondominant and feasible individuals, the value of r_g has no impact on the inequality (7). In other words, it does not change the order of ranking of the two individuals.

However, the value of r_g is critical in the first two cases as r_g is the flipping point that will determine whether the comparison is objective or penalty function dominated. For example, if r_g increased to a value greater than $r_c(i)$ in the first case, individual *i*+1 would change from a fitter individual into a less-fit one. For the entire population, the chosen value of r_g is used for comparisons will determine the fraction of individuals dominated by the objective and penalty functions. Not all possible r_g values can influence the ranking of individuals. They have to be within a certain range, i.e. $r_g^l < r_g < r_g^u$ to influence the

ranking, where the lower bound r_g^l is the minimum critical penalty coefficient computed from adjacent individuals ranked only according to the objective function, and the upper bound r_g^u is the maximum critical penalty coefficient computed from adjacent individuals ranked only according to the penalty function. In general, there are three different categories of values.

1) $r_g < r_g^l$: All comparisons are based only on the fitness function. r_g is too small to influence the ranking of individuals. We will call this under penalization.
2) $r_g > r_g^l$: All comparisons are based only on the penalty function. r_g Is so large that the impact of the objective function can be ignored. We will call this overspecialization.
3) $r_g^l < r_g < r_g^u$: All comparisons are based on a combination of objective and penalty functions.

All penalty methods can be classified into one of the above three categories. Some methods may fall into different categories during different stages in search. It is important to understand the differences among these three categories because they indicate which function (combination of functions) is driving the search process and how search progresses. For example, most dynamic methods start with a low r_g value (i.e. $r_g < r_g^l$) in order to find a good region which may contain both feasible and infeasible individuals. Toward the end of the search, a high r_g value (i.e. $r_g^u < r_g$) is often used in order to locate a good feasible individual. Such a dynamic method would work well for problems for which the unconstrained global optimum is close to its constrained global optimum. It is unlikely to work well for problems for which the constrained global optimum is far away from its unconstrained one because the initial low r_g value would drive the search toward the unconstrained global optimum, and thus further away from the constrained one.

It has been widely recognized that neither under penalization nor over-penalization is a good constraint handling technique, and there should be a balance between preserving feasible individuals and rejecting infeasible ones [5]. In other words, ranking should be dominated by a combination of objective and penalty functions, and so the penalty coefficient r_g should be within the bounds $r_g^l < r_g < r_g^u$. It is worth noting that the two bounds are not fixed. They are problem dependent, and may change from generation to generation as they are also determined by the current population. It is clear from the analysis in this section which has been carried out by [19] that all penalty methods try to obtain the right balance between objective and penalty functions so that the search moves toward the optimum in the feasible space, not just toward the optimum in the combined feasible and infeasible space. One way to achieve such balancing effectively and efficiently is to adjust such balance directly and implicitly. This is what ranking, described in the next section, does.

3. IMPLEMENTATION OF THE EVOLUTIONARY ALGORITHM FOR CONSTRAINED OPTIMIZATION

It is clear from the analysis in this section which has been carried out by [19] that all penalty methods try to obtain the right balance between objective and penalty functions so that the search moves toward the optimum in the feasible space, not just toward the optimum in the combined feasible and infeasible space. One way to achieve such balancing effectively and efficiently is to adjust such balance directly and implicitly. This is what ranking, described in the next section, does.

A. Ranking

To overcome the difficulty of determining the optimal r_g a different approach is suggested in this section to balance the dominance of the objective and penalty functions. The following fitness function is introduced

$$fitness(x) = f(x) + rank_f + rank_g * \sum_{j=1}^{m} \phi(x) \quad (9)$$

Where $rank_f$ is the rank of the objective function values, which takes values in the range of [1 – *population size*]. $rank_g$ Is the rank of the sum of the constraints violation for each solution, which takes values from [(*population size* + 1)−(2* *population size*)]. What (9) above mounts to is that minimum fitness value and less constraints violation inevitably leads to best fitness value. By using rank-based roulette wheel selection [16], [17], self-adapting is achieved without any extra computational cost. More importantly, the motivation of ranking comes from the need for

balancing objective and penalty functions directly and implicitly in optimization. Equation (9) provides a convenient way of balancing the dominance in a ranked set. The algorithm is listed below.

B. An Illustrative Example

In this section the ranking based on (9) is discussed for a simple NLP problem. The purpose of this section is to show the important effects of the ranking and not to solve the NLP problem. First, the results of an actual search are presented, and then the imposed search direction is discussed.

The problem is as follows. A quadratic function is to be minimized and the feasible region formed by g_1 and g_2 constraints is a narrow crescent-shaped region (approximately 0.7% of the total search space) with the optimum solution lying on the first constraint. The problem is stated as follows:

$$f(x) = \left(x_1^2 + x_2 - 11\right)^2 + \left(x_1 + x_2^2 - 7\right)^2$$

Subject to:

$$g_1(x) = -4.84 + \left(x_1 - 0.05\right)^2 + \left(x_2 - 2.5\right)^2 \le 0$$

$$g_2(x) = -x_1^2 - \left(x_2 - 2.25\right)^2 \_ 4.84 \le 0 \qquad (10)$$

$$0 \le x_1 \le 6\ ,\ 0 \le x_2 \le 6$$

For the corresponding unconstrained problem the optimal solution is $x^* = [3, 2]$. For the constrained problem (10) the optimal solution is $x^* = (2.246826, 2.381865)$ with a function value equal to $f^* = 13.59085$. The population size is set to 10, the maximum number of generations is 50, the mutation probability is set to 0.01, and crossover probability is set to 0.9. The initial generation is shown in Figure 1. The rank of initial generation according to (9) is also given in the figure. The best solution in the initial generation corresponds to ranking 1. Figure 1shows clearly that the search direction is towards the feasible region in the initial generation.

Figure 2 shows the ratio of feasible solutions, the mean normalized Euclidean distance and the ratio between the true optimum and the best-found feasible objective value. To avoid premature convergence it is crucial to have sufficient diversity in the population. Such diversity in the population is achieved by using rank-based roulette wheel selection [16], [17]. The first feasible solution is found in generation 5. In the early generations (5 to 8) the number of feasible individuals increases rapidly. In generation 10 all individuals are feasible and the improvement in the objective function by generation 30 is very close to the optimum since the population has converged too fast. To make this variation of the search direction clearer, two populations with different ratios of feasible individuals are studied in figures 3 and 4. In both figures an infeasible individual is created by a mutation. This individual is better than all the feasible individuals in terms of the objective value, $f(x)$. Due to the rank in Equation (9), this infeasible individual becomes the best individual. The search is then directed out of the feasible region towards the unconstrained optimum and as a result better feasible solutions are found.

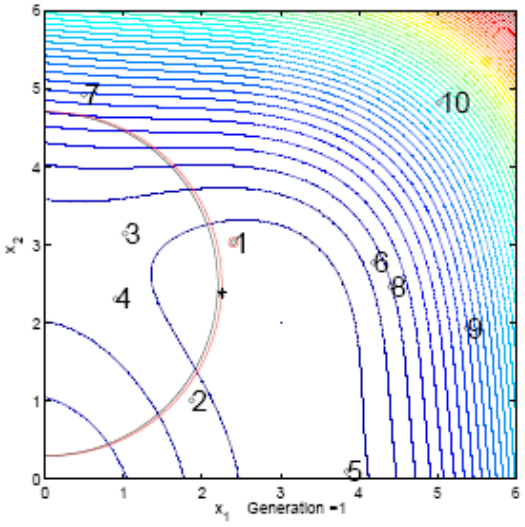

Fig.1. Initial generation (black circles) shown in decision space. The red circle indicates the best solution based on equation (9), the black star is the optimum solution

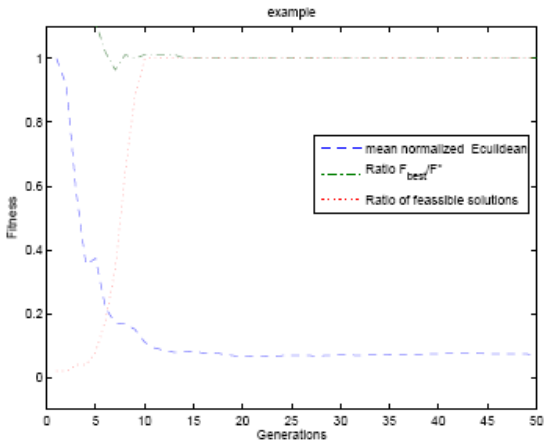

Fig. 2. Search results for problem (10)

This example shows the dynamic behavior of the search direction. The variation of the search direction has a positive effect for the population diversity. Thus, no special operation to preserve the diversity in the population is required for most

cases. In the next section the method is evaluated using a set of selected test problems gathered from the literature.

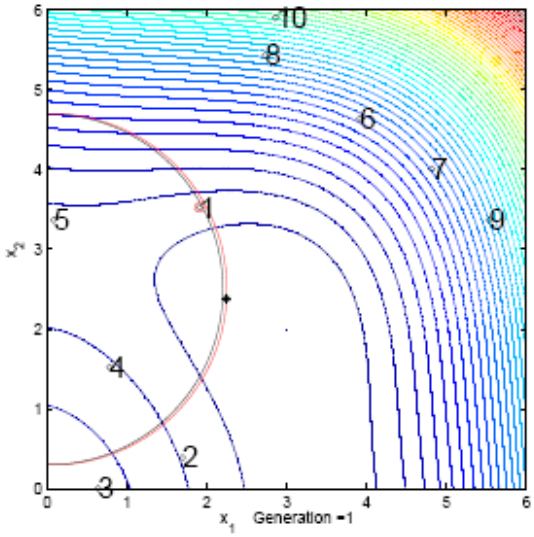

Fig. 3. Rank according to Equation (9) for a population.

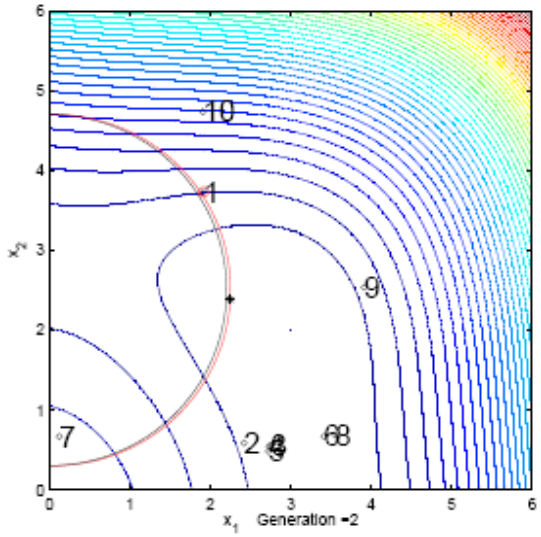

Fig. 4. Rank according to Equation (9) for a population.

C. *Experimental Results and Discussions*

The thirteen benchmark functions chosen from [12], [14] contain characteristics that are representative of what can be considered "difficult global optimization problems" for an evolutionary algorithm. To get an estimate of how difficult it is to generate feasible points through a purely random process, Mezura and Carlos [13] computed the ρ ½ metric (as suggested by Michalewicz and Schoenauer [23]) using the following expression:

$$\rho = \frac{|F|}{|S|} \qquad (11)$$

where $|S|$ is the number of random solutions generated ($|S|$ = 1000000 in their case), and $|F|$ is the number of feasible solutions found (out of the $|S|$ total solutions randomly generated).

The values of ρ for each of the functions chosen are shown in Table I, where n is the number of decision variables, LI is the number of linear inequalities, NI the number of nonlinear inequalities, LE is the number of linear equalities, and NE is the number of nonlinear equalities.

Prob.	n	Type of fun.	ρ	LI	NI	LE	NE
g01	13	quadratic	0.0003%	9	0	0	0
g02	20	nonlinear	99.9973%	2	0	0	0
g03	10	nonlinear	0.0026%	0	0	0	1
g04	5	quadratic	27.0079%	4	2	0	0
g05	4	nonlinear	0.0000%	2	0	0	3
g06	2	nonlinear	0.0057%	0	2	0	0
g07	10	quadratic	0.0000%	3	5	0	0
g08	2	nonlinear	0.8581%	0	2	0	0
g09	7	nonlinear	0.5199%	0	4	0	0
g10	8	linear	0.0020%	6	0	0	0
g11	2	quadratic	0.0973%	0	0	0	1
g12	3	quadratic	4.7697%	0	9^3	0	0
g13	5	nonlinear	0.0000%	0	0	1	2

TABLE I
VALUES OF ρ FOR THE 13 TEST PROBLEMS CHOSEN.

Problem	Study	Pop size	Max gen	Max mutation step
g01	This study [4]	130 130	500 N/A	0.09 -
g02	This study [4]	200 -	1000 -	0.05 -
g03	This study [4]	200 -	500 -	0.04 -
g04	This study [4]	200 50	1000 5000	0.02 -
g05	This study [4]	400 -	2000 -	0.02 -
g06	This study [4]	200 -	2000 -	0.02 -
g07	This study [4]	200 100	1000 3500	0.025 -
g08	This study [4]	200 -	100 -	0.025 -
g09	This study [4]	200 70	400 5000	0.028 -
g10	This study [4]	200 80	500 4000	0.0015 -
g11	This study [4]	200 -	300 -	0.025 -
g12	This study [4]	200 -	50 -	0.025 -
g13	This study [4]	200 50	1000 7000	0.002 -

TABLE II
GA PARAMETERS USED FOR THE RESULTS PRESENTED IN TABLE III

Problems g02, g03, g08, and g12 are maximization problems. They are transformed into minimization problems using $-f(x)$. For each of the benchmark problems, 50 independent runs are performed. All experiments are performed in MATLAB.

In [4] Deb tested his method on nine different problems of which test problems #1 to test problem #6. These problems are a subset of the test problems considered in this study. The results obtained by the proposed method in this paper are compared to the results obtained by Deb on all these six test problems. Furthermore, Deb stated that "in all cases, the proposed approach has been able to repeatedly find solutions closer to the true

optimum solution than that reported earlier." Therefore, a fair comparison to the results reported in [4] should give good indication of the performance of the method presented in this paper.

The source code may be obtained from the authors upon request

Problem	Study	Best	Median	Worst
g01	This study	-15.000	-15.000	-14.0298
	[4]	-15.000	-15.000	-13.000
g02	This study	-0.8036	-0.8036	-0.7996
	[4]	-	-	-
g03	This study	-1	-1	-1
	[4]	-	-	-
g04	This study	-30665.5	-30665.5	-30665.3
	[4]	-30665.5	-30665.5	-29846.7
g05	This study	5126.5	5126.5	5129.2
	[4]	-	-	-
g06	This study	-6961.8	-6961.8	-6933.7
	[4]	-	-	-
g07	This study	24.327	24.374	24.642
	[4]	24.372	24.409	25.075
g08	This study	0.095825	0.095825	0.095723
	[4]	-	-	-
g09	This study	680.630	680.632	681.885
	[4]	680.634	680.642	680.651
g10	This study	7055.8	7566.8	88845.3
	[4]	7060.2	7220.0	10230.8
g11	This study	0.750	0.750	0.750
	[4]	-	-	-
g12	This study	1.000	1.000	1.000
	[4]	-	-	-
g13	This study	0.0053	0.0035	0.0000
	[4]	0.05395	0.24129	0.50776

TABLE III
RESULTS FOR THE PROPOSED ALGORITHM (ADAPTIVE GA–RRWS)COMPARED TO BEST RESULTS [4]. THE INDEPENDENT NUMBER OF RUNS IS 50

The GA parameters used in this study are presented in Table II for each problem. Then, the results for this algorithm are compared to the best results reported in [4].

First, some typical search results for the problems are presented in Figure 5 to Figure 17. These figures show the ratio of feasible solutions, the mean normalized Euclidean distance and the ratio between the optimal solution and the best-found feasible solution.

Obviously, the easiest problem is the test problem g12. Near optimal solutions are found in early generations. Surprisingly, the most difficult of these thirteen problems for this method are test problems g05, g07 and g13 where the ρ metric is 0% as shown in Table I. The results for adaptive GA-RRWS algorithm are now compared to the best results for all tested methods in [4] and summarized in Table III. The problems g02, g03, g05, g06, g08, g11, and g12 are not a part of the test problems studied in [4].

As can be seen from Table III, adaptive GA-RRWS algorithm outperforms the Deb method [4] in all the problems except problem g13 where the algorithm fails to get solutions close to the optimum which requires further research and modification.

The best found results of the 50 independent runs for adaptive GA-RRWS algorithm are almost better than the results reported by Deb. For these problems the variation in the best results found is less for the proposed method than those reported in [4]. It should be mentioned, however, that the results presented by Deb are based on tournament selection with a niching method that required two extra parameters. Furthermore, the maximum number of generations for the results of test problem g01 in [4] is not known. Hence it is difficult to make a fair comparison of the results on this problem.

Problem	Deb's method			Proposed method			$Gain = \frac{Y-X}{Y} * 100$
	Pop_size	Max_gen	Y	Pop_size	Max_gen	X	
g04	50	5000	250000	200	1000	20000	20%
g07	100	2500	350000	200	1000	200000	42.857%
g09	70	5000	350000	200	400	80000	77.14%
g11	80	4000	320000	200	500	10000	68.75%
Average							52.25%

TABLE IV
COMPARISON BETWEEN DEB'S METHOD [4] AND THE PROPOSED ALGORITHM (ADAPTIVE GA–RRWS) FROM THE NUMBER OF FITNESS EVALUATIONS POINT OF VIEW. WHERE X, Y = MAXIMUM FITNESS EVALUATIONS.

For all the test problems the algorithm has consistently found solutions closer to the optimal solution for all 50 runs.

Deb used 50 Independent runs. The maximum function evaluations equal to product of the maximum number of generations and population size. The table IV shows why the proposed approach is outperforming the approach used by Deb. Deb did not show the maximum number of generations for problem g01, so we cannot compare with it.

From the table IV, this comparison between Deb's method and the proposed method shows that the proposed method has outperformed Deb's method with gain 52.25%.

4. CONCLUSIONS AND FUTURE WORK

This paper has proposed a new constraint handling technique based on ranking called Adaptive GA-RRWS. Ranking is motivated by the analysis of penalty methods from the point of view of dominance. The balance between the objective and penalty functions is achieved through rank and the fitness function (9). The introduction of rank and using that rank in the fitness function enables the algorithm to bias toward the global optimum. The proposed method could get solutions closer to the optimum solution on twelve test problems and fails in getting any good solution on problem g13 which requires further research and modification.

The new constraint-handling technique was tested on a set of 13 benchmark problems. Experimental results have been presented and have outperformed Deb's method with gain 52.25%. The future work of this study includes the application of ranking to other types of evolutionary algorithms. The Adaptive GA-RRWS algorithm does not

introduce any specialized variation operators, and does not require a priori knowledge about a problem since it uses adaptive penalty coefficient r_g in a penalty function.

REFRENCES:

[1] X. Qi A. Homaifar, S.H.-V. Lai. *"Constrained optimization via genetic algorithms"*. Simulation, 62(4):242–254, 1994.

[2] E. Camponogara and S. N. Talukdar. *"A genetic algorithm for constrained and multiobjective optimization"*. In Jarmo T. Alander, editor, 3rd Nordic Workshop Genetic Algorithms and Their Appl.(3NWGA), pages 49–62, Finland, 1997. Univ. Vaasa.

[3] K. Deb. *"Optimization for Engineering Design: Algorithms and Examples"*. Prentice-Hall, New Delhi, 1995.

[4] K. Deb. *"An efficient constrained handling method for genetic algorithms"*. Computer Methods in Applied Mechanics and Engineering, 1999.

[5] M. Gen and R. Cheng. *"Genetic Algorithms and Engineering Design"*. Wiley, New York, 1997.

[6] D. E. Goldberg. *"Genetic Algorithms in Search, Optimization, and Machine Learning"*. Addison-Wesley, New York, NY, 1989.

[7] K.M. Ragsdell G.V. Reklaitis, A. Ravindran. *"Engineering Optimization Methods and Applications"*. Wiley, New York, 1983.

[8] J. H. Holland. *"Adaptation in Natural and Artificial Systems"*. University of Michigan Press, Ann Arbor, MI, 1975.

[9] C. R. Houck J.A. Joines. *"On the use of nonstationary penalty functions to solve nonlinear constrained optimization problems with GAs"*. In Z. Michalewicz, editor, International Conference on Evolutionary Computation, pages 579–584, Piscataway, 1994. IEEE Press.

[10] J. Joines and C. Houck. *"On the use of nonstationary penalty functions to solve nonlinear constrained optimization problems with GAs"*. In IEEE Int. Conf. Evolutionary Computing, 1994.

[11] S. Kazarlis and V. Petridis. *"Varying fitness functions in genetic algorithms: Studying the rate of increase in the dynamic penalty terms"*. In Parallel Problem Solving from Nature, pages 211–220, 1998.

[12] S. Koziel and Z. Michalewicz. *"Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization"*. Evol. Comput., 7(1):19–44, 1999.

[13] Coello C.A.C. Mezura-Montes, E. *"A simple multimembered evolution strategy to solve constrained optimization problems"*. IEEE Trans, Evolutionary Computation, 9(1):1–17, 2005.

[14] Z. Michalewicz. *"Genetic algorithms, numerical optimization, and constraints"*. In Eshelman, editor, the Sixth International Conference on Genetic Algorithms, pages 151–158, San Mateo, 1995. Morgan Kauman.

[15] Z. Michalewicz and N. Attia. *"Evolutionary optimization of constrained problems"*. In A. V. Sebald and L. J. Fogel, editors, 3rd Annu. Conf. Evolutionary Programming, page 98108, River Edge, 1994. World Scientific.

[16] Omar Al Jadaan, Lakshmi Rajamani, C. R. Rao. *"Improved selection operator for GA"*. Journal of Theoretical and Applied Information Technology, 4(4):269277, 2008.

[17] Omar Al Jadaan, Lakshmi Rajamani, C. R. Rao. *"Parametric study to enhance genetic algorithm performance, using ranked based roulette wheel selection method"*. In InSciT2006, volume 2, pages 274–278, Merida, Spain, 2006.

[18] C. R. Reeves. *"Genetic algorithms for the operations researcher"*. INFORMS J. Comput., 9(3):231–247, 1997.

[19] Thomas P. Runarsson and Xin Yao. *"Stochastic ranking for constrained evolutionary optimization"*. IEEE Transactions on Evolutionary Computation, 4(3):284294, 2000.

[20] W. Siedlecki and J. Sklansky. *"Constrained genetic optimization via dynamic reward-penalty balancing and its use in pattern recognition"*. In Int. Conf. Genetic Algorithms, pages 141–149, 1989.

[21] A. E. Smith and D. W. Coit. *"Handbook on Evolutionary Computation, chapter Penalty functions"*, pages C5.2:1–C5.2:6. Oxford Univ. Press, Oxford, U.K., 1997.

[22] P. D. Surry and N. J. Radcliffe. *"The COMOGA method: Constrained optimization by multiobjective genetic algorithms"*. Contr. Cybern., 26(3), 1997.

[23] M. Schoenauer Z. Michalewicz. *"Evolutionary algorithms for constrained parameter optimization problems"*. Evolutionary Computation, 4(1):1–32, 1996.

[24] N. Attia Z. Michalewicz. *"Evolutionary optimization of constrained problems"*. In L.J. Fogel A.V. Sebald, editor, the Third Annual Conference on Evolutionary Programming, pages 98–108, Singapore, 1994. World Scientific.

Algorithm 1 Adaptive GA-RRWS

1: Initialize Population *P*
2: Generate random population − size *N*
3: Evaluate objective values and constraints
4: Calculate the rank of objective values R_f for each solution $R_f \in [1-N]$
5: Calculate the rank of the sum of the constraints violation R_f for each solution $R_f \in [N+1-(2N)]$
6: Assign fitness values based on (9) for each solution in *P*
7: Generate offspring Population *Q* from *P*
8: {Ranked based Roulette Wheel Selection
9: Recombination and Mutation
10: Evaluate objective values and constraints }
11: *For g=1 to G do*
12: Calculate the rank of objective values R_f for each solution in the combined population $P \cup Q, R_f \in [1-2N]$
13: Calculate the rank of the sum of the constraints violation R_f for each solution in the combined population $P \cup Q, R_g \in [(2N+1)-(4N)]$
14: Assign fitness values based on (9) for each solution in the combined population $P \cup Q$
15: *P* = Select the best *N* members of the combined population based on Fitness values to make the population of the next generation
16: *Q* = Create next generation from *P* {
17: Ranked based Roulette Wheel Selection
18: Recombination and Mutation
19: Evaluate objective values and constraints }
20: **end for**

APPENDIX
TEST FUNCTION SUITE

All benchmark functions with the exception of g13 are described in [14]. They are summarized here for completeness.

1) g01

$$Min\ f(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=1}^{13} x_i$$

Subject to:

$$g_1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \le 0$$
$$g_2(x) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \le 0$$
$$g_3(x) = 2x_1 + 2x_3 + x_{11} + x_{12} - 10 \le 0$$
$$g_4(x) = -8x_1 + x_{10} \le 0$$
$$g_5(x) = -8x_2 + x_{11} \le 0$$
$$g_6(x) = -8x_3 + x_{12} \le 0$$
$$g_7(x) = -2x_4 + x_5 + x_{10} \le 0$$
$$g_8(x) = -2x_6 + x_7 + x_{11} \le 0$$
$$g_9(x) = -2x_8 + x_9 + x_{12} \le 0$$

Where the bounds are $0 \le x_i \le 1 (i = 1,...,9)$, $0 \le x_i \le 100 (i = 10,10,12)$ and $0 \le x_{13} \le 1$ the global minimum is at $x^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$ where six constraints are active (g_1, g_2, g_3, g_7, g_8, *and* g_9 and $f(x^*) = -15$.

2) g02

$$MAX\ f(x) = \left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right|$$

Subject to:

$$g_1(x) = 0.75 - \prod_{i=1}^{n} x_i \le 0$$
$$g_1(x) = \sum_{i=1}^{n} x_i - 7.5n \le 0$$

Where n = 20 and $0 \le x_i \le 10 (i = 1,...,10)$.

The global maximum is unknown; the best we found is $f(x^*) = 0.803619$ (which, to the best of our knowledge, is better than any reported value), g_1 constraint is close to being active ($g_1 = -10^{-8}$).

3) g03

$$Max\ f(x) = (\sqrt{n})^n \prod_{i=1}^{n} x_i$$

Subject to:

$$h_1(x) = \sum_{i=1}^{n} x_i^2 - 1 = 0$$

Where n = 10 and $0 \le x_i \le 1 (i = 1,...,n)$ the global maximum is at $x_i^* = \frac{1}{\sqrt{n}} (i = 1,...,n)$; where $f(x^*) = 1$.

4) g04

$$Min\ f(x) = 5.3578547x_3^2 + 0.08356891x_1x_5$$

$+37.293239x_1 - 40792.141$

Subject to:

$g_1(x) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \le 0$

$g_2(x) = -85.334407 - 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 \le 0$

$g_3(x) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \le 0$

$g_4(x) = -80.51249 - 0.0071317x_2x_5 + 0.0029955*x_1x_2 - 0.0021813x_3^2 + 90 \le 0$

$g_5(x) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x3x4 - 25 \le 0$

$g_6(x) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 - 20 \le 0$

Where

$78 \le x_1 \le 102$, $33 \le x_2 \le 45$, $27 \le x_i \le 45$ (i = 3, 4, 5)

the optimum solution is x^* = (78, 33, 29.995256025682, 45, 36.775812905788) where $f(x^*) = -30665.539$. Two constraints are active (g_1 and g_6).

5) g05

$Min\ f(x) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3$

Subject to:

$g_1(x) = -x_4 + x_3 - 0.55 \le 0$

$g_2(x) = -x_3 + x_4 - 0.55 \le 0$

$h_3(x) = 1000 \sin(-x_3 - 0.25) + 1000 \sin(-x_4 - 0.25) + 894.8 - x_1 = 0$

$h_4(x) = 1000 \sin(x_3 - 0.25) + 1000 \sin(-x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$

$h_5(x) = 1000 \sin(x_4 - 0.25) + 1000 \sin(-x_4 - x_3 - 0.25) + 1294.8 = 0$

Where $0 \le x_{1,2} \le 1200, -0.55 \le x_{3,4} \le 0.55$. The best known solution [12] x^* = (679.9453, 1026.0670, 1.188764, 0.3962336) Where $f(x^*) = 5126.4981$.

6) g06

$Min\ f(x) = (x_1 - 10)^3 + (x_2 - 20)^3$

Subject to:

$g_1(x) = -(x_1 - 5)^5 - (x_2 - 5)^2 + 100 \le 0$

$g_2(x) = (x_1 - 6)^2 - (x_2 - 5)^2 + 82.81 \le 0$

Where $13 \le x_1 \le 100$ and $0 \le x_2 \le 100$. The optimum solution is $x^* = (14.095, 0.84296)$. Both constraints are active.

7) g07

$Min\ f(x) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_10 - 7)^2 + 45$

Subject to:

$g_1(x) = 105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \le 0$

$g_2(x) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \le 0$

$g_3(x) = -8x_1 + 2x_2 + 5x_9 - 2x_10 - 12 \le 0$

$g_4(x) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \le 0$

$g_5(x) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \le 0$

$g_6(x) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \le 0$

$g_7(x) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \le 0$

$g_8(x) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_10 \le 0$

Where $-10 \le x_1 \le 10, (i = 1, ..., 10)$.

The optimum solution is x^* = (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, and 8.375927). Where $f(x^*)$ =24.3062091. Six constraints are active (g_1, g_2, g_3, g_4, g_5 and g_6).

8) g08

$$Min\ f(x) = \frac{\sin(2\pi x_1)^3 \sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$$

Subject to:

$g_1(x) = x_1^2 - x_2 + 1 \le 0$

$g_2(x) = 1 - x_1 + (x_2 - 4)^2 \le 0$

Where $0 \le x_{1,2} \le 10$. The optimum is located at

x^* = (1.2279713, 4.2453733) where $f(x^*)$ = 0.095825. The solution lies within the feasible region.

9) g09

$$Min\ f(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x(3)^2 + 3(x_4 - 11)^2 + 10x\frac{6}{5} + 7x\frac{2}{6} + x\frac{4}{7} - 4x_6 * x_7 - 10x_6 - 8x_7$$

Subject to:

$$g_1(x) = -127 + 2x\frac{2}{1} + 3x\frac{4}{2} + x_3 + 4x\frac{2}{4} + 5x_5 \le 0$$

$$g_2(x) = -282 + 7x_1 + 3x_2 + 10x\frac{2}{3} + x_4 - x_5 \le 0$$

$$g_3(x) = -196 + 23x_1 + x\frac{2}{2} + 6x\frac{2}{6} - 8x_7 \le 0$$

$$g_4(x) = 4x\frac{2}{1} + x\frac{2}{2} - 3x_1x_2 + 2x\frac{2}{3} + 5x_6 - 11x_7 \le 0$$

Where $-10 \le x_i \le 10$ **for** $(i = 1,...,7)$**.**

The optimum solution is x^***=(**2.330499, 1.951372, 0.477544, 4.365726, 0.6244870, 1.03131,1.594227**)** Where $f(x^*)$ = 680.6300573. Tow constraints are active (g_1 and g_2).

10) g10

$$Min\ f(x) = x_1 + x_2 + x_3$$

Subject to:

$$g_1(x) = -1 + 0.0025(x_4 + x_6) \le 0$$

$$g_2(x) = -1 + 0.0025(x_5 + x_7 - x_4) \le 0$$

$$g_3(x) = -1 + 0.01(x_8 - x_5) \le 0$$

$$g_4(x) = -x_1x_6 + 833.33252x_4 + 100x_1 - 83333.333 \le 0$$

$$g_5(x) = -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \le 0$$

$$g_6(x) = -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \le 0$$

Where $100 \le x_1 \le 10000, 1000 \le x_{2,3} \le 1000$, and $10 \le x_i \le 1000$, for $(i = 4,...,8)$.

The optimum solution is x^*=(579.3167, 1359.943, 5110.071, 182.0174, 295.5985, 217.9799, 286.4162, 395.5979) where $f(x^*)$ =7049.3307. Three constraints are active (g_1, g_2 and g_3).

11) g11

$$Min\ f(x) = x\frac{2}{1} + (x_2 - 1)^2$$

Subject to:

$$h(x) = x_2 - x\frac{2}{1} = 0$$

Where $-1 \le x_{1,2} \le 1$. The optimum solution is:

$x^* = \pm\sqrt{2}, 1/2$ where $f(x^*)$ =0.75.

12) g12

$$Min\ f(x) = (100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2)/100$$

Subject to:

$$g(x) = (x_1 - p)^2(x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \le 0$$

Where $0 \le x_{1,2,3} \le 10$ and p, q, r =1,…, 9 the feasible region of the search space consists of disjointed g^3 spheres.

A point (x_1, x_2, x_3) is feasible if and only if there exist p, q, r such that the above inequality holds. The optimum is located at x^* = (5,5,5) where $f(x^*)$ =1.

The solution lies within the feasible region.

13) g13

$$Min\ f(x) = \exp^{x_1x_2x_3x_4x_5}$$

Subject to:

$$h_1(x) = x\frac{2}{1} + x\frac{2}{2} + x\frac{2}{3} + x\frac{2}{4} + x\frac{2}{5} - 10 = 0$$

$$h_2(x) = x_2x_3 - 5x_4x_5 = 0$$

$$h_3(x) = x\frac{3}{1} + x\frac{3}{2} + 1 = 0$$

Where $-2.3 \le x_{1,2} \le 3.2, -3.2 \le x_{3,4,5} \le 3.2$.

The optimum solution is x^*=1.717143, 1.595709, 1.827247, 0.7636413, 0.763645 where $f(x^*)$ =0.0539498.

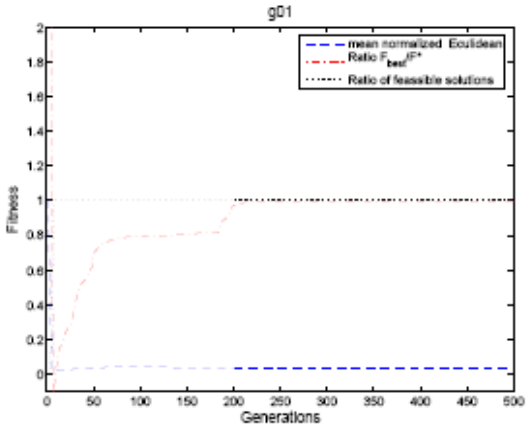

Fig. 5. Typical result for problem g01.

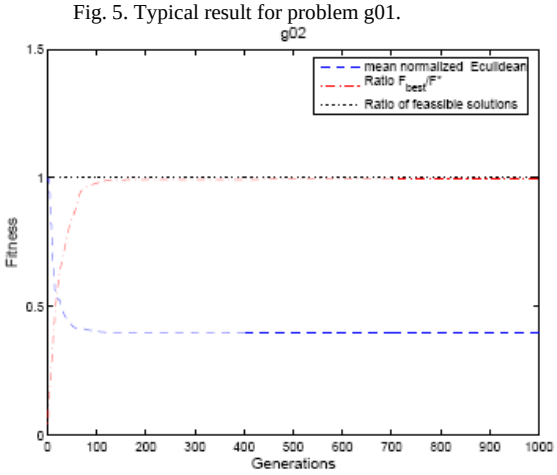

Fig. 6. Typical result for problem g02.

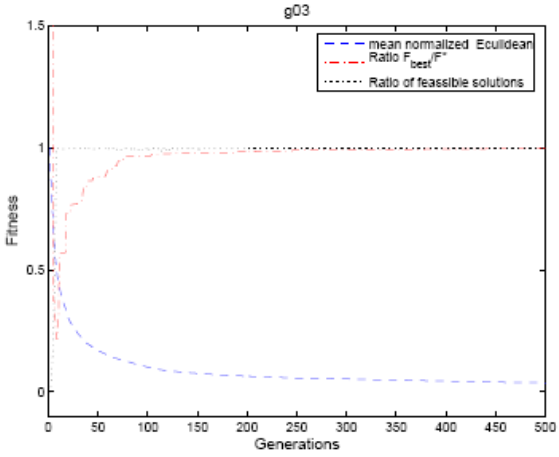

Fig. 7. Typical result for problem g03.

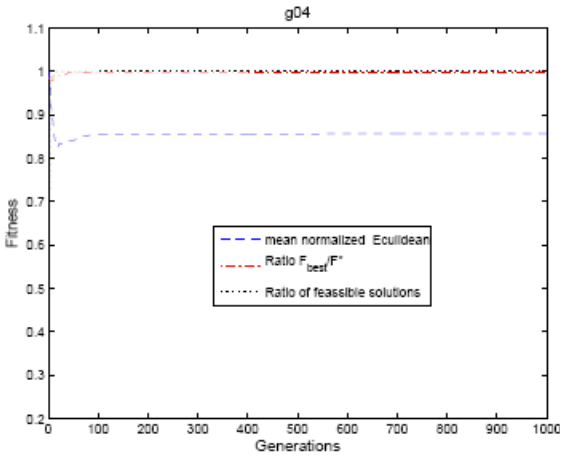

Fig. 8. Typical result for problem g04.

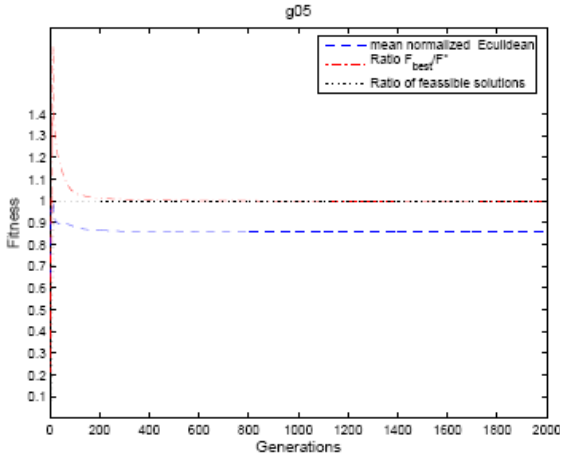

Fig. 9. Typical result for problem g05.

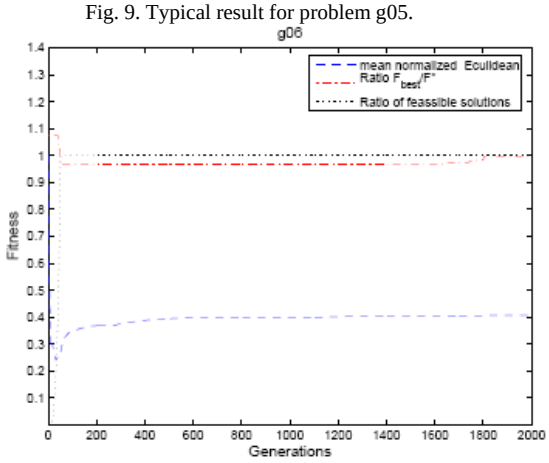

Fig. 10. Typical result for problem g06.

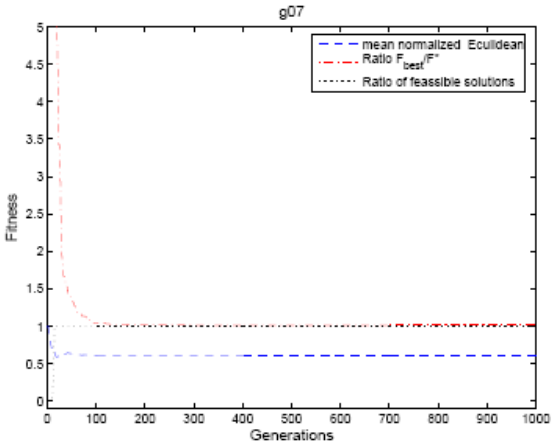

Fig. 11. Typical result for problem g07.

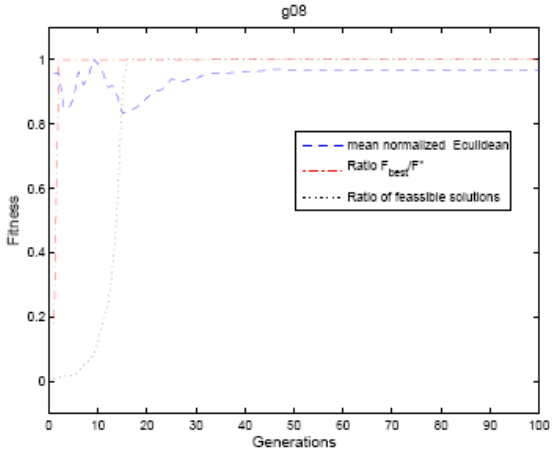

Fig. 12. Typical result for problem g08.

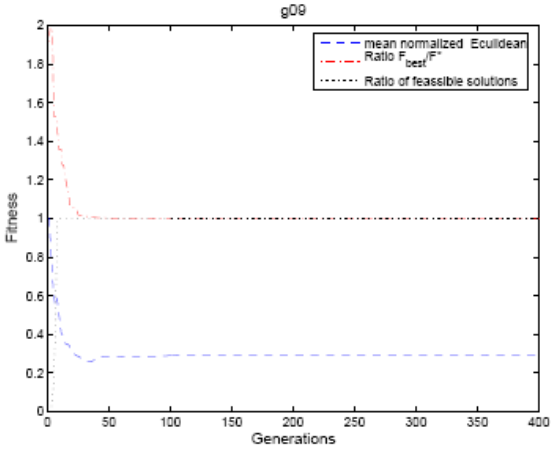

Fig. 13. Typical result for problem g09.

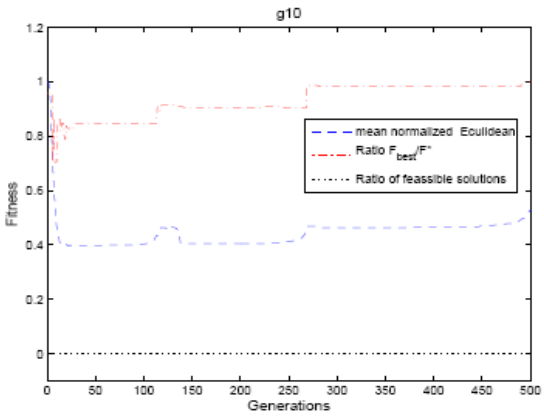

Fig. 14. Typical result for problem g10.

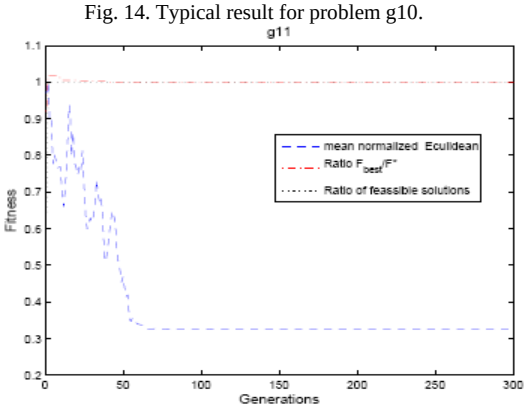

Fig. 15. Typical result for problem g11.

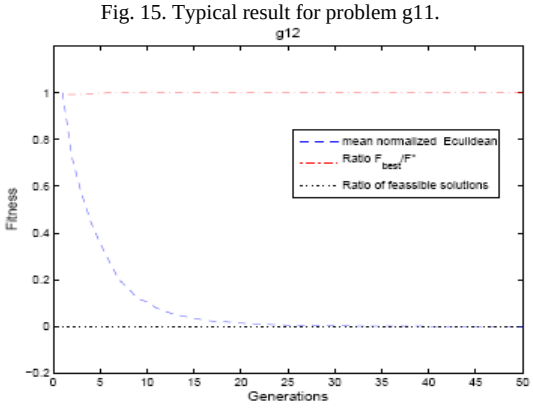

Fig. 16. Typical result for problem g12.

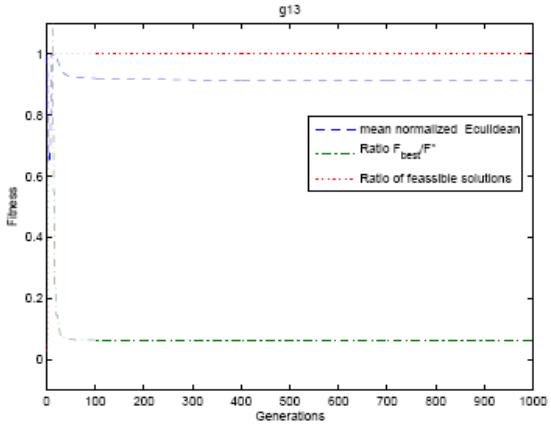

Fig. 17. Typical result for problem g13.