Self-Adaptive Fitness Formulation for Constrained Optimization

Raziyeh Farmani and Jonathan A. Wright

***Abstract*—A self-adaptive fitness formulation is presented for solving constrained optimization problems. In this method, the dimensionality of the problem is reduced by representing the constraint violations by a single infeasibility measure. The infeasibility measure is used to form a two-stage penalty that is applied to the infeasible solutions. The performance of the method has been examined by its application to a set of eleven test cases from the specialized literature. The results have been compared with previously published results from the literature. It is shown that the method is able to find the optimum solutions. The proposed method requires no parameter tuning and can be used as a fitness evaluator with any evolutionary algorithm. The approach is also robust in its handling of both linear and nonlinear equality and inequality constraint functions. Furthermore, the method does not require an initial feasible solution.**

***Index Terms*—Constraint handling, dynamic, fitness, genetic algorithm, penalty, self-adaptive.**

NOMENCLATURE

$c(\mathbf{X})$	Constraint violation value.
$c_{\max,j}$	Scaling factor for violation of constraint j.
$F(\mathbf{X})$	Fitness.
$f(\mathbf{X})$	Objective function.
$\ddot{f}(\mathbf{X})$	Penalized objective function value after the second penalty.
$\dot{f}(\mathbf{X})$	Penalized objective function value after the first penalty.
$g(\mathbf{X})$	Inequality constraint.
$h(\mathbf{X})$	Equality constraint.
$\imath(\mathbf{X})$	Solution's infeasibility.
X	Vector of decision variables.
$\check{\mathbf{X}}$	"Best" individual.
$\hat{\mathbf{X}}$	"Worst" infeasible individual.
$\breve{\mathbf{X}}$	Individual with the highest objective function value.
δ	A small tolerance.
γ	Scaling factor.

I. INTRODUCTION

IN THE LAST two decades, genetic algorithms have received much attention regarding their potential as global optimization techniques. More recently, the solution of constrained optimization problems has been addressed by many researchers ([1], [5], [6], [9], [13], [21], [23]). Penalty function methods are among the most common methods used to solve constrained optimization problems. In these methods, a penalty term is added to the objective function, the penalty increasing with the degree of constraint violation (static penalty) or the degree of constraint violation and generation number (dynamic penalty) ([11], [12]). In general the weakness of penalty methods is that they often require several parameters (to adjust the relative weights of each constraint in the penalty, and the weight of the penalty against the objective function). However, due to their simplicity and ease of implementation they are the most common methods used in solving real world problems.

Michalewicz and Janikow [17] presented the GENOCOP method which is based on designing specialized operators that incorporate knowledge of the constraints. This method uses projection operators that map feasible points back to feasible boundaries. The approach is only applicable to linear constraints and an objective function with a feasible starting point. To overcome this limitation Michalewicz and Attia [15] introduced a hybrid optimization system for general nonlinear programming problems (GENOCOP II). Later, Michalewicz *et al.* [18] developed the GENOCOP III method which is based on the idea of repairing infeasible solutions and also incorporating some concepts of coevolution. Schoenauer and Michalewicz [25], constructed further operators that maintain solutions on nonlinear analytical constraint surfaces.

In order to avoid generating and rejecting a large number of infeasible solutions, specialized operators can be used. In the Greedy decoder method, the chromosome does not directly encode a solution in the feasible region but rather a set of parameters that is used by the decoder to generate a feasible solution. Because the decoder must be guaranteed to never produce infeasible solutions, it is often extremely difficult to design. Hajela and Yoo [10] overcame this problem in an alternative approach that is able to handle both nonlinear, equality and inequality constraints. The strategy is based on a preconditioning of the infeasible solutions prior to the genetic transformation. The approach is conceptually analogous to the theory of emulation of the immune system and is effective in evolving feasible solutions. In this approach both feasible and infeasible solutions should be present in the population at any generation of the search. The basis for this scheme is that those segments of the binary chromosome, that contribute to calculating the objective function are minimally altered, while those segments of the chromosome that contribute to constraint violations are replaced by corresponding segments from feasible chromosomes.

Another class of constraint handling methods involves the biasing feasible over infeasible solutions. Schoenauer

Manuscript received January 24, 2003; revised May 29, 2003. This work was supported in part by the U.K. Engineering and Physical Science Research Council under Platform Grant GR/R14712/01 and Grant GR/M23601/01.

R. Farmani is with the School of Engineering and Computer Science, University of Exeter, Exeter, Devon EX4 4QF, U.K. (e-mail: R.Farmani@exeter.ac.uk).

J. A. Wright is with the Department of Civil and Building Engineering, Loughborough University, Loughborough, Leicestershire LE11 3TU, U.K. (e-mail: J.A.Wright@lboro.ac.uk).

Digital Object Identifier 10.1109/TEVC.2003.817236

and Xanthakis [26] presented a behavioral memory method which considers all constraints in a sequence; when sufficient number of feasible individuals satisfy one constraint from the sequence the next constraint from the sequence is considered. The success of the whole process is highly dependent on the genetic diversity maintained during the initial steps, thus, ensuring a uniform sampling of the feasible region. Powell and Skolnick [21] presented a method that would make all infeasible solutions have fitness values less than the worst feasible solution. Deb [6] suggested a modification of Powell and Skolnick [21] method which does not require any penalty parameters. This method uses a tournament selection operator, where two solutions are compared at a time. In this method, any feasible solution is preferred to any infeasible solution; among two feasible solutions, the one having a better objective function value is preferred and among two infeasible solutions, the one having smaller constraint violation is preferred.

Multiobjective algorithms have been used in the solution of constrained single objective optimization problems, by treating the constraints as one or more of the objectives. Surry and Radcliffe [28] introduced the constrained optimization by multiobjective genetic algorithm (COMOGA) method in which all members of the population are ranked on the basis of the constraint violations. Such a rank, together with the value of the objective function, lead to a two-objective optimization problem. Cheng and Li [2] presented an alternative constrained multiobjective optimization method. The approach integrates a Pareto genetic algorithm and fuzzy penalty function. In this method, the rank of a solution is determined by knowing the solution's status (feasible or infeasible), the distance from the Pareto optimal set, and position in infeasible region. The fuzzy-logic penalty function method, having a discrete membership function, can express the rank order of solutions in a Pareto optimization and transform a multiobjective constrained optimization into an unconstrained problem. Parmee and Purchase [20] and Coello [3] used the vector evaluated genetic algorithm (VEGA) (Schaffer [24]) to handle each of the constraints as an objective. In this approach, the population is divided into subpopulations, each representing one of the constraints, which will be guided during the search. Coello [3] eliminated some of the drawbacks of the Parmee and Purchase [20]. However, one of the drawbacks of Coello's [3] method is the selection of the most appropriate number and size of each subpopulation. Also, care needs to be given to the way that subpopulations are guided so certain subpopulations do not dominate the search.

Although interest in treating constrained optimization problems as a multiobjective optimization problem is growing ([7], [8], [29]), the approach appears less robust than for constrained single objective algorithms. The difficulty arises from the fact that for highly constrained problems, simply considering constraints as objectives or assigning the infeasible individuals a lower fitness than the feasible individuals might not introduce enough pressure to direct the search toward the region of the optimum.

Hybrid methods combine evolutionary techniques with deterministic optimization procedures for numerical optimization problems. Myung and Kim [19] presented a two-phase evolutionary programming method based on a hybrid method. During the first phase, an evolutionary algorithm is used to optimize the function. In the second phase of the optimization, Lagrange multipliers are used to place emphasis on the violated constraints whenever the best solution does not fulfill the constraints. By updating the Lagrange multipliers, the trial solutions are driven to the optimal point, where all constraints are satisfied.

Adaptive constraint handling methods have also been developed. These are appealing due to their ability to adjust their own parameters and make use of information in the population. Coit *et al.* [5] and Coit and Smith [4] introduced an adaptive penalty technique which makes use of feed back obtained during the search along with a dynamic distance metric for genetic optimization of constrained combinatorial problems. The approach uses a near-feasibility threshold (NFT) for each constraint or set of constraints. The penalty function encourages the genetic algorithm to explore within the feasible region and the NFT neighborhood of the feasible region. Ben Hamida and Schoenauer [1] presented an adaptive segregational algorithm for constrained optimization that operates in three stages. First, the global information of the population is used to adjust the penalty coefficient, then a recombination strategy is used to mate feasible individuals with infeasible individuals, and finally, segregational selection is used to favor a given number of feasible individuals.

Koziel and Michalewicz [13] presented the homomorphous mapping approach for solving constrained optimization problems. The method incorporates a homomorphous mapping between an n-dimensional cube and the feasible search space. This approach introduces an additional problem-dependent parameter to partition the interval [0, 1] into subintervals of equal length such that the equation of each constraint has, at most, one solution in every subinterval. The method loses the locality feature of the mapping for nonconvex feasible search spaces and a small change in the coded solution may result in a large change in the solution itself. This process requires additional computational effort for finding all the intersection points for a line segment with the boundaries of the feasible region. The disadvantages of the homomorphous method are that it requires an initial feasible solution and that all infeasible solutions are rejected. Another limitation is the need for problem-dependent parameters in the method.

Runarsson and Yao [23] introduced a stochastic ranking method in which the objective function values are used for ranking the solutions in the infeasible region of the search space. A probability parameter is used to determine the likelihood of two individuals in the infeasible space being compared with each other. Although the method proved to be effective in solving a wide range of constrained optimization problems, it was also sensitive to the choice of probability parameter.

Most of these constraint handling methods are problem dependent. They often require user supplied parameters to be adjusted in order to obtain good performance from the method. Some of the methods are also able to handle only specific constraint types and, therefore, lack generality. Some of the approaches limit the search to the feasible search space. However, a good search should approach the optimum solution from

both sides of the feasible/infeasible border [22]. Smith and Coit [27] pointed out that there is need for development of completely adaptive penalty functions that require no user specified constants and development of improved adaptive operators to exploit characteristics of the search as they are found.

Wright and Farmani [30] presented a fitness formulation which addresses the limitations of some of the existing constraint handling methods. In particular, it does not require parameter tuning and can be used without an initial feasible solution being given. This paper describes the further development of this algorithm as a self-adaptive penalty method. The paper also describes the evaluation of the algorithm's performance in solving eleven standard test functions taken from the literature; the performance of the algorithm is compared with a number of previous studies.

II. Fitness Formulation

The fitness formulation described here is based on the method for constraint minimization that was proposed by Wright and Farmani [30]. The method has been formulated to ensure that slightly infeasible solutions with a low objective function value remain fit. This is seen as a benefit to solving highly constrained problems that have solutions on one or more of the constraint bounds. In contrast, solutions farthest from the constraint bounds are seen as containing little genetic information that is of use and are, therefore, penalized.

The infeasibility values are represented by the sum of the normalized constraint violation values. The infeasibility measure has the properties that it increases in value with both the number of active constraints and the magnitude of each constraint violation. The infeasibility measure is used to form a two-stage penalty applied to the infeasible solutions. The first-penalty stage ensures that the worst of the infeasible solutions has a penalized objective function value that is higher or equal to that of the best solution in the population (all other solutions in the population are also penalized but by a lesser amount, depending on their feasibility). The second penalty increases the penalized objective function value of the worst of the infeasible solutions to twice the objective value of the best solution. The remaining infeasible individuals are penalized exponentially in proportion to their infeasibility. The approach is implemented in three stages. First, an infeasibility is assigned to each individual, second, the "best" and "worst" individuals in the population are identified, and finally, the two-part penalty function is applied to the infeasible solutions. There are two main advantages to this approach. First, it does not require any parameter tuning, and second, it is able to find the global optimum starting with a completely infeasible population of solutions. The method is also able to solve a range of constrained optimization problems, having both nonlinear equality and inequality constraints. It was shown, that this approach gives results that are better or comparable to those of existing methods. The use of a fixed weight for the second penalty also proved to be effective, although for test problem G4, improved results were obtained by reducing the weight of the second penalty [30].

The aim of this work is to eliminate the fixed weight for the second penalty which could cause problems if solutions are clustered in a small part of the search space. The second penalty increases the first penalized objective function values such that the second penalized objective function value of the "worst" individual is twice that of the best. In this paper, the fitness formulation has been modified so that after applying the second penalty, the penalized objective function value of the "worst" individual is equal to that of the individual with maximum objective function value in the current population. This modification makes the method more dynamic (not only does the "best" vary for each population but the individual with maximum objective function value also varies from one population to another). This helps spread fitness for populations that are clustered in a small part of the search space, but without violating the basic principles behind the fitness formulation.

III. Self-Adaptive Fitness Formulation

The dynamic fitness formulation described here has the advantage of being a self-adaptive method. The approach does not require parameter tuning and can be used without any initial feasible solution being given (this being an advantage in real world applications having many optimization variables). The approach is also robust in its handling of both linear and nonlinear equality and inequality constraint functions. The methodology described here applies to the minimization of an objective function $f(\mathbf{X})$

$$f(\mathbf{X}) = f(x_1, \ldots, x_n). \tag{1}$$

Subject to inequality constraints

$$g_j(\mathbf{X}) \leq 0, \quad (j = 1, \ldots, q) \tag{2}$$

and equality constraints

$$h_j(\mathbf{X}) = 0, \quad (j = q+1, \ldots, m). \tag{3}$$

The algorithm has three stages. First, each individual is assigned an infeasibility, second, the bounding solutions of the search space are identified, and finally, the infeasible solutions are penalized.

A. Chromosome Infeasibility

The infeasibility of an individual should represent both the number of active constraints and the extent to which each constraint is violated. A measure of infeasibility that has these properties is the sum of the normalized constraint values for all violated constraints. This can be evaluated in two stages. First, the feasible constraint values are reset as zero and infeasible values as positive [(4)]; in this research, a small tolerance δ has also been applied to the equality constraints to aid the finding of a feasible solution. Second, the solution's infeasibility ($\imath(\mathbf{X})$), is taken as the sum of the normalized constraint violation values [(5)]

$$c_j(\mathbf{X}) = \begin{cases} \max\left(0, g_j(\mathbf{X})\right), & \text{if } 1 \leq j \leq q \\ \max\left(0, \left(|h_j(\mathbf{X})| - \delta\right)\right), & \text{if } q+1 \leq j \leq m \end{cases} \tag{4}$$

$$\imath(\mathbf{X}) = \frac{\sum_{j=1}^{m} \frac{c_j(\mathbf{X})}{c_{\max,j}}}{m}. \tag{5}$$

The constraint violation values are normalized since large differences in the magnitude of the constraint values can lead to dominance of the infeasibility by constraints having the highest values. The scaling factor for each constraint $c_{\max,j}$, is taken as the maximum value of the constraint violation in the current population. Resetting the scaling factor for each population provides a further dynamic element to the infeasibility calculation. This has been found to give better algorithm performance than for a static scaling factor, (for instance, by basing $c_{\max,j}$ on the constraint violations in the first population).

B. Identification of the Bounding Solutions

The penalty functions are applied in relation to three bounding solutions:

- $\breve{\mathbf{X}}$ "best" individual;
- $\hat{\mathbf{X}}$ "worst" of the infeasible solutions;
- $\breve{\breve{\mathbf{X}}}$ solution with the highest objective function value in the current population.

For a population containing at least one or more feasible solutions, the "best" individual $\breve{\mathbf{X}}$, is the feasible solution having the lowest objective function value. However, if all individuals are infeasible then the best solution is taken as the solution having the lowest infeasibility value (regardless of the objective function value of the individuals).

The "worst" of the infeasible solutions $\hat{\mathbf{X}}$, is selected by comparing all infeasible individuals against the best individual ($\breve{\mathbf{X}}$). Two potential population distributions exist in relation to this comparison.

- The first population distribution occurs when **one or more** of the infeasible solutions have an objective function value that is **lower** than the "best" solution. In this case, the "worst" of the infeasible solutions is taken as the infeasible solution having the highest infeasibility value and an objective function value that is lower than the "best" solution's. If more than one individual exists with the same highest infeasibility values, then the $\hat{\mathbf{X}}$ is taken as the solution with maximum infeasibility value and the lower of the objective function values.
- The second population distribution occurs when **all** of the infeasible solutions have an objective function value that is **greater** than the "best" solution. Here, the "worst" of the infeasible solutions is identified as being the solution with the highest infeasibility value. If more than one individual exists with the same highest infeasibility value, then $\hat{\mathbf{X}}$ is taken as the solution with the maximum infeasibility value and the higher of the objective function values.

C. Chromosome Fitness

Since we are concerned with the minimization of the objective function, the infeasible solutions are penalized prior to the conversion of the objective function values to fitness form. The conversion to fitness ($F(\mathbf{X})$) is by the simple subtraction of the penalized objective function values ($\ddot{f}(\mathbf{X})$), from the maximum penalized value in the current population. The objective function values of the infeasible solutions are penalized according to the solution's infeasibility in relation to that of the "worst" of the infeasible solutions ($\imath(\hat{\mathbf{X}})$), the "best" solution ($\imath(\breve{\mathbf{X}})$) and the highest objective function value in the current population ($f(\breve{\breve{\mathbf{X}}})$). Note that if a feasible solution exists, then the "best" solution is feasible and will have a zero infeasibility ($\imath(\breve{\mathbf{X}}) = 0.0$).

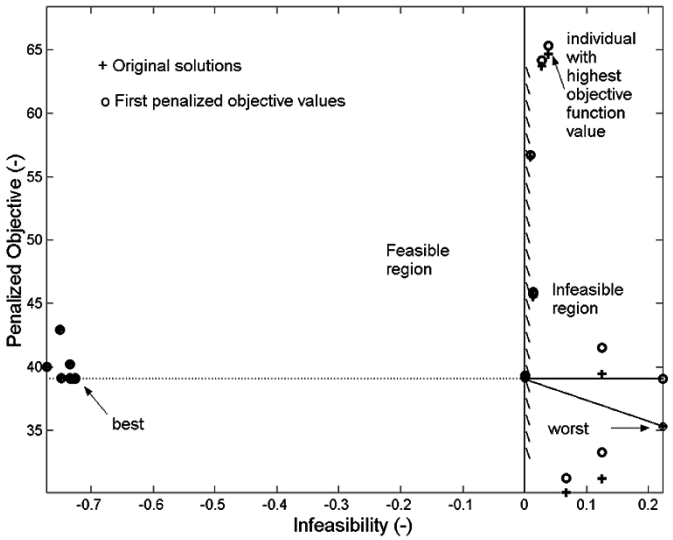


Fig. 1. Application of the first penalty.

The penalty is applied in two stages. The first stage only applies if one or more infeasible solutions have a lower and, therefore potentially better objective function value than the "best" solution ($\exists X | (f(\mathbf{X}) < f(\breve{\mathbf{X}})) \wedge (\imath(\mathbf{X}) > 0.0)$). If this relationship holds, then the penalty is applied to all of the infeasible solutions. If the relationship does not hold, then the first penalty is not applied to any solution. The goal of the first penalty is to increase the objective function value of the infeasible solutions such that the "worst" of the infeasible solutions has an objective function value that is equal to that of the "best" solution. This has been implemented using a simple linear relationship between the objective function values and the infeasibility of the "best" and "worst" infeasible solutions [(6) and (7)]

$$\tilde{\imath}(\mathbf{X}) = \frac{\imath(\mathbf{X}) - \imath(\breve{\mathbf{X}})}{\imath(\hat{\mathbf{X}}) - \imath(\breve{\mathbf{X}})} \tag{6}$$

$$\dot{f}(\mathbf{X}) = f(\mathbf{X}) + \tilde{\imath}(\mathbf{X})\,(f(\breve{\mathbf{X}}) - f(\hat{\mathbf{X}})). \tag{7}$$

Note that if the penalty is not applied then $\dot{f}(\mathbf{X}) = f(\mathbf{X})$. The application of the first penalty is illustrated by Fig. 1 in which the original solutions are indicated by a "+" and the penalized solutions by a "o." The "best" and "worst" of the infeasible solutions are connected by a line (with the infeasibility of the "best" solution being reset to zero). Note that in this and subsequent figures, a negative infeasibility indicates a feasible solution (the negative infeasibilities only being assigned for the purposes of illustrating the distribution of solutions in the current population).

The second penalty increases the objective function values such that the penalized objective function value of the "worst" infeasible individual is equal to that of the "worst" objective individual [(8) and (9)]. Making the method more dynamic and self-adaptive has been found to give good performance in comparison with the previous fitness formulation [30] that used a fixed weight for the second penalty.

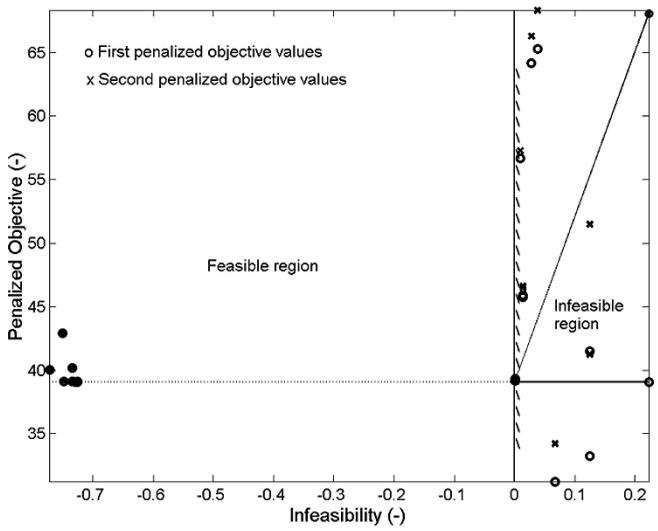


Fig. 2. Application of the second penalty.

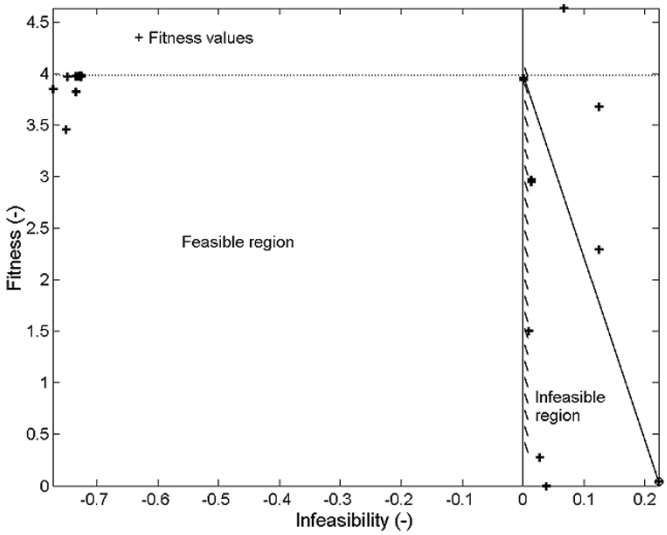


Fig. 3. Solution fitness.

It is important to note that the exponential weighting parameter of 2.0 in (8), is a constant and does not require tuning. The exponential function gives a slight reduction in the rate of penalty applied to solutions of low infeasibility, thus, helping to maintain the fitness of the slightly violated solutions. A study of the effect of the weighting parameter indicated that the algorithm was insensitive to the parameter provided that the parameter values had the effect of only slightly reducing the penalty weight. However, it was concluded that the slight reduction in penalty weight resulting from a penalty value of 2.0, gave good algorithm performance over a range of test problems

$$\ddot{f}(\mathbf{X}) = \dot{f}(\mathbf{X}) + \gamma \left|\dot{f}(\mathbf{X})\right| \left(\frac{\exp(2.0\,\tilde{i}(\mathbf{X})) - 1.0}{\exp(2.0) - 1.0}\right) \quad (8)$$

$$\gamma = \begin{cases} \frac{f(\check{\mathbf{X}}) - f(\hat{\mathbf{X}})}{f(\hat{\mathbf{X}})}, & \text{if } (f(\hat{\mathbf{X}}) \leq f(\check{\mathbf{X}})) \\ 0.0, & \text{if } (f(\hat{\mathbf{X}}) = f(\check{\mathbf{X}})) \\ \frac{f(\check{\mathbf{X}}) - f(\hat{\mathbf{X}})}{f(\hat{\mathbf{X}})}, & \text{if } (f(\hat{\mathbf{X}}) > f(\check{\mathbf{X}})) \end{cases} . \quad (9)$$

The scaling factor γ, simply ensures that the penalized value of "worst" infeasible solution is equivalent to the highest objective function value in the current population. The second case in (9) ($\gamma = 0.0$), applies when the "worst" infeasible individual has an objective function value equal to the highest in the population. Here, no penalty is applied since the infeasible solutions would naturally have a low fitness and should not be penalized further. The use of absolute values of the objective function in (8) is necessary to allow the minimization of objective functions having negative values.

The application of the second penalty is illustrated in Fig. 2 (the first penalized objective values being indicated by "o" and the second penalized objective values by "$\times$").

Fig. 3, shows the penalized objective function values converted to fitness form. Note that the fittest individuals lie in both the infeasible and feasible regions. This allows the slightly infeasible, low objective function value solutions to be selected for reproduction. Note also that in this case, the fittest individual (with the lowest penalized objective function value), is infeasible.

It is evident that the approach is dynamic in the allocation of the penalty in that the absolute value of the penalty depends on the objective values of the "best," the "worst" infeasible, and the highest objective individuals. The penalty also accounts for the range of infeasibility in the current population and the distribution of the infeasible solutions in relation to the "best" individual in the population.

Fig. 4 illustrates the general procedure for self-adaptive fitness formulation method.

IV. Test Cases

The performance of the proposed constraint handling method has been evaluated using a set of 11 test cases ([13], [16]). These test cases include various forms of objective function (linear, quadratic, cubic, polynomial, nonlinear), and each test case has a different number of variables (n). The test problems also pose a range of constraint types and number of constraints [linear inequalities (LI); nonlinear equalities (NE); and nonlinear inequalities (NI)]. The general form of each test case is given in Table I, which also indicates the number of constraints active at the optimum solution (a).

The self-adaptive fitness formulation described here has been implemented and evaluated using simple genetic algorithm with Gray encoding of the variables (25 bits used to represent each variable). The implementation uses proportional ("roulette wheel") selection strategy, single point crossover, random bit mutation, and finally an elitist replacement strategy.

The performance of any evolutionary algorithm for constrained optimization is determined by the constraint handling technique used as well as the evolutionary search algorithm (including parameters) [23]. The performance of the algorithm described here has been compared with the results reported for the homomorphous mapping method [13], and therefore, where

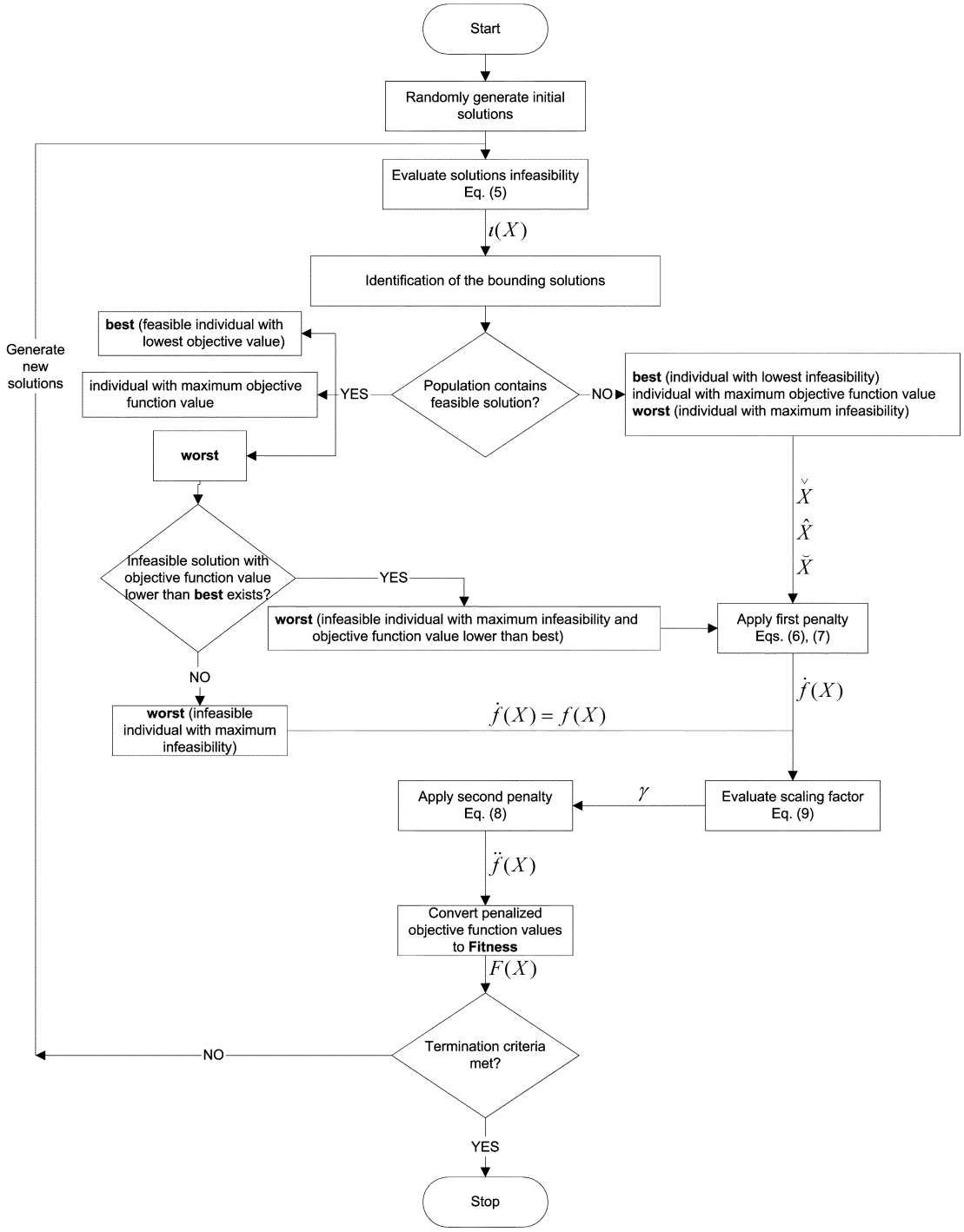


Fig. 4. The optimization procedure.

possible, the same genetic algorithm (GA) parameter values have been adopted (a population size of 70; 90% probability of crossover; and a probability of mutation between 0.3%–0.5%). The small tolerance of $\delta = 0.0001$ is applied to the equality constraints. A comparison of the algorithm performance is also made to the results obtained by Runarsson and Yao [23] and Ben Hamida and Schoenauer [1] for the same test problems although it is understood that both methods used an evolution strategy algorithm and performed different experiments to those given in this paper. Also, a comparison of the algorithm to other methods ([6], [9], [21], [30]) that reported results for some of these test functions is given. For each test case, the two types of experiment have been performed as described by Koziel and Michalewicz [13].

- Experiment 1: 20 runs each starting from a different randomly generated population; the maximum number of generations was set to 5000.
- Experiment 2: the same as Experiment 1, except that the maximum number of generations was increased to 20 000.

The result of Experiments 1 and 2 are given in Table II.

TABLE I
SUMMARY OF TEST CASES

Function	n	Form of $f(\mathbf{X})$	LI	NE	NI	a
G1	13	quadratic	9	0	0	6
G2	k	nonlinear	0	0	6	1
G3	k	polynomial	0	1	0	1
G4	5	quadratic	0	0	6	2
G5	4	cubic	2	3	0	3
G6	2	cubic	0	0	2	2
G7	10	quadratic	3	0	5	6
G8	2	nonlinear	0	0	2	0
G9	7	polynomial	0	0	4	2
G10	8	linear	3	0	3	6
G11	2	quadratic	0	1	0	1

A. General Performance of the Algorithm

Table II indicates that the algorithm described here found good, if not optimum solutions for all eleven test cases. This algorithm is a development of that described in [30]. The results for the best and mean solutions for the current and the earlier version of the algorithm are given in Tables V and VI. A comparison of the results for the two versions of the algorithm indicate that adding a further adaptive element to the fitness formulation has resulted in an improvement in best and average results for all test functions.

It should also be noted that the standard deviations for the test results (Table II) are very small. This is an important characteristic for the application of the algorithm to the solution of "real world" problems, where cost and time constraints prohibit repeated runs of the algorithm. The small standard deviations for the algorithm described here indicate that it is robust in finding a near optimum solution without the need for repeated runs of the optimization.

As well as the two experiments conducted here, Koziel and Michalewicz [13] presented a third experiment in which the initial population was seeded with the best solution from experiment 1 (thus, testing the algorithm's sensitivity to a good initial guess of the solution). This experiment was not conducted here as results for the earlier version of this algorithm [30] made clear that its ability to find the optimum solution was not sensitive to an initial guess of the solution.

B. Performance in Finding a Feasible Solution

The ability to find a feasible solution was examined in Experiment 1 (random initialization, and a maximum of 5000 generations). For 9 of the 11 test cases (G1, G2, G3, G4, G6, G7, G8, G9, G11) the algorithm described here found a feasible solution for all 20 independent runs (and in many cases, the search also found the optimum solution). Furthermore, on average the first feasible solution for this group of problems was found within 26 generations (a maximum of 1820 function evaluations). However, problem G10 required a higher number of generations (517 generations) in order to find a feasible solution. Furthermore, feasible solutions for G10 were only found for 17 out of the 20 runs. For problem G10, all six constraints are active at the solution, which makes it particularly difficult to solve. Feasible solutions for G5 were only found for 9 out of the 20 runs, the difficulty in solving this problem being associated with the equality constraints.

C. Comparison to Benchmark Algorithms Performance

The results for the algorithm described here are compared in detail to those for two previously published algorithms. The first is the homomorphous mapping method [13] and the second is the stochastic ranking algorithm [23]. Although the results for the homomorphous mapping have been surpassed by several algorithms, they can still be considered representative of benchmark performance. The results for the stochastic ranking are perhaps the best published to date.

Table III gives the results for the homomorphous mapping [13]. A comparison of these results to those in Table II indicates that the method described here can find more optimal solutions. These solutions are shown in "**bold**" in Table II. Further, the results reported by Koziel and Michalewicz [13] (Table III) require an initial feasible solution, whereas the approach described here can solve the problems starting from a randomly generated and completely infeasible population. The ability to find a feasible solution, as well as the optimum solution represents an improvement in the algorithm's performance (although this characteristic has also recently been reported by Runarsson and Yao [23]).

The algorithm matched most of the solutions reported by Runarsson and Yao [23] (see Tables II and IV). The stochastic ranking method has the advantage that it is simple to implement, but it can be sensitive to the value of its single control parameter. Further, it was less robust in finding a feasible solution for the highly constrained G10 problem (with only 6 feasible solutions from 30, in comparison to 17 from 20 for the algorithm described here). The improved performance of the algorithm described here may be due to the deterministic handling of constraint violations and suggests that it has better performance in solving highly constrained problems.

D. Comparison With Other Published Results

Table V summarizes the best results for the 11 test cases reported by different researchers ([1], [6], [13], [23], [30]) and those obtained by the self-adaptive fitness formulation.

The mean values for 11 test cases reported by different researchers are given in Table VI. Note that the results for Powell and Skolnick [21] and Hadj–Alouane and Bean [9] have been reported by Makinen *et al.* [14]. Deb [6] did not reported any mean

TABLE II
RESULTS FROM EXPERIMENTS 1 AND 2 (SELF ADAPTIVE FITNESS FORMULATION)

Function	Optimum value	Experiment #1				Experiment #2			
		worst	best	average	standard deviation	worst	best	average	standard deviation
G1	-15	**-14.9980**	**-15.0000**	**-14.9993**	4.99E-4	**-15.0000**	**-15.0000**	**-15.0000**	0
G2	0.803553	0.74398	**0.79989**	0.77512	1.53E-2	0.76043	**0.80297**	0.79010	1.2E-2
G3	1.0	**0.99830**	**0.99978**	**0.99930**	3.89E-4	**0.99970**	**1.00000**	**0.99990**	7.5E-5
G4	-30665.5	**-30628.93**	**-30665.45**	**-30659.41**	9.88E+0	**-30663.30**	**-30665.50**	**-30665.20**	4.85E-1
G5	5126.4981		**5828.6181**			**6089.4300**	**5126.9890**	**5432.0800**	3.887E+3
G6	-6961.8	**-6961.699**	**-6961.796**	**-6961.769**	2.35E-2	**-6961.800**	**-6961.800**	**-6961.800**	0
G7	24.306	**32.69**	**24.59**	27.83	2.09E+0	28.40	**24.48**	26.58	1.14E+0
G8	0.095825	**0.029159**	**0.095825**	**0.092539**	1.45E-2	**0.095825**	**0.095825**	**0.095825**	0
G9	680.63	**681.53**	**680.69**	**680.97**	2.5E-1	**680.87**	**680.64**	**680.72**	5.92E-2
G10	7049.33	**8568.81**	**7070.23**	**7760.54**	4.79E+2	**8288.79**	**7061.34**	**7627.89**	3.73E+2
G11	0.75	0.7772	**0.7500**	0.7546	7.27E-3	**0.7500**	**0.7500**	**0.7500**	0

TABLE III
RESULTS FROM EXPERIMENTS 1 AND 2 [13]

Function	Optimum value	Experiment #1			Experiment #2		
		worst	best	average	worst	best	average
G1	-15	-14.0566	-14.7207	-14.4609	-14.6154	-14.7864	-14.7082
G2	0.803553	0.78427	0.79506	0.79176	0.79119	0.79953	0.79671
G3	1.0	0.9917	0.9983	0.9965	0.9978	0.9997	0.9989
G4	-30665.5	-30617.0	-30662.5	-30643.8	-30643.8	-30645.9	-30655.3
G5	5126.4981						
G6	-6961.8	-4236.7	-6901.5	-6191.2	-5473.9	-6952.1	-6342.6
G7	24.306	38.682	25.132	26.619	25.069	24.620	24.826
G8	0.095825	0.0291434	0.095825	0.0871551	0.0291438	0.095825	0.0891568
G9	680.63	682.88	681.43	682.18	683.18	680.91	681.16
G10	7049.33	11894.5	7215.8	9141.7	9659.3	7147.9	8163.6
G11	0.75	0.75	0.75	0.75	0.75	0.75	0.75

values for the test results. Among different constraint handling methods reported here only [1], [13], [23], [30], and self-adaptive fitness formulation methods have reported results for all the 11 test cases. Some of these methods [6], [9], and [21] have not reported results for problems G2 and G5. These two problems pose a challenge for constraint handling methods and are a good measure of testing their ability in handling problems that do not have similar proportion of feasible and infeasible regions. The problem G2 is a highly feasible problem with none or only few infeasible individuals at each population. Contrary to this, problem G5 is a highly infeasible problem having three equality constraints.

In general, the algorithm described here, performs as well as, and in some cases better than the alternative algorithms. This is the case for both the best and average values found; in particular, the average solutions indicate that the approach described here is among the most robust in finding the optimum solution.

TABLE IV
RESULTS OF STOCHASTIC RANKING [23]

Function	Optimum value	Stochastic Ranking			
		worst	best	average	Standard deviation
G1	-15	-15.0	-15.0	-15.0	0.0E+0
G2	0.803553	0.726288	0.803515	0.781975	2.0E-02
G3	1.0	1.0	1.0	1.0	1.9E-04
G4	-30665.5	-30665.539	-30665.539	-30665.539	2.0E-05
G5	5126.4981	5142.472	5126.497	5128.881	3.5E+00
G6	-6961.8	-6350.262	-6961.814	-6875.94	1.6E+02
G7	24.306	24.642	24.307	24.374	6.6E-02
G8	0.095825	0.095825	0.095825	0.095825	2.6E-17
G9	680.63	680.763	680.63	680.656	3.4E-02
G10	7049.33	8835.655	7054.316	7559.192	5.3E+02
G11	0.75	0.75	0.75	0.75	8.0E-05

TABLE V
RESULTS FOR BEST VALUES

Function	Optimum value	Koziel and Michalewicz 1999	Ben Hamida and Schoenauer 2000	Runarsson and Yao 2000	Deb 2000	Wright and Farmani 2001	Self Adaptive Fitness Formulation
G1	-15	-14.7864	**-15.0000**	**-15.0000**	**-15.0000**	-14.9996	**-15.0000**
G2	0.803553	0.799530	0.800781	**0.803515**		0.796640	0.802970
G3	1.0	0.9997	**1.0000**	**1.0000**		0.9994	**1.0000**
G4	-30665.5	-30664.900	**-30665.500**	**-30665.539**	**-30665.537**	-30661.100	**-30665.500**
G5	5126.4981		4707.5200	**5126.4970**		5126.6398	5126.9890
G6	-6961.8	-6952.100	**-6961.810**	**-6961.814**		-6961.370	**-6961.800**
G7	24.306	24.620	24.360	**24.307**	24.373	24.671	24.480
G8	0.095825	**0.095825**	**0.095825**	**0.095825**		**0.095825**	**0.095825**
G9	680.63	680.91	**680.63**	**680.63**	**680.63**	681.20	680.64
G10	7049.33	7147.90	7095.15	**7054.32**	7060.22	7152.83	7061.34
G11	0.75	**0.75**	**0.75**	**0.75**		**0.75**	**0.75**

E. Algorithm Performance in a Discontinuous Search Space

The problem below has been formulated to test the performance of the algorithm in solving problems with a discontinuous search space

Maximize

$$f(X)=\frac{(100-(x_1-5)^2-(x_2-5)^2-(x_3-5)^2)}{100}$$

subject to

$$(x_1-p)^2+(x_2-q)^2+(x_3-r)^2\leq 0.0625$$

where

$$0\leq x_i\leq 10 \quad (i=1,2,3) \quad p,q,r=1,2,\ldots,9.$$

The feasible region of the search space consists of 9^3 disjointed spheres. A point (x_1,x_2,x_3) is feasible if, and only if, there exist p, q, r such that $(x_1-p)^2+(x_2-q)^2+(x_3-r)^2\leq 0.0625$. The optimum value for this function is 1 at point (5,5,5).

Two experiments were conducted for this test case. These experiments were the same as those used in the 11 test cases

TABLE VI
RESULTS FOR MEAN VALUES

Function	Optimum value	Koziel Michalewicz 1999	Reported by Makinen et al. 1999		Ben Hamida Schoenauer 2000	Runarsson Yao 2000	Wright and Farmani 2001	Self Adaptive Fitness Formulation
			Powell Skolnick	Hadj-Alouane				
G1	-15	-14.708	-13.534	-13.975	-14.682	-15.000	-14.988	-15.000
G2	0.803553	0.796710			0.546200	0.781975	0.784650	0.790100
G3	1.0	0.9989			0.9998	1.0000	0.9990	0.9999
G4	-30665.5	-30655.3			-30650.8	-30665.5	-30591.8	-30665.2
G5	5126.4981				4323.73	5128.88	5131.04	5432.08
G6	-6961.8	-6342.6	-4125.5	-5436.2	-6961.8	-6875.9	-6657.8	-6961.8
G7	24.306	24.826	33.547	30.884	24.611	24.374	30.927	26.580
G8	0.095825	0.089157			0.095825	0.095825	0.092460	0.095825
G9	680.63	681.16	685.22	685.40	680.65	680.66	684.41	680.72
G10	7049.33	8163.60			8858.64	7559.19	8255.85	7627.89
G11	0.75	0.75			0.75	0.75	0.81	0.75

and with the same parameters. The method performed very well with the best, average, worst, and standard deviation equal to 1, 0.994 375, 0.999 718 75, and 5.34E-04, respectively, for the first experiment and 1, 1, 1, 0 for the second experiment. This shows that the self-adaptive fitness formulation can be applied to problems having a discontinuous search space as well as continuous search space.

V. CONCLUSION

This paper introduces a self-adaptive fitness formulation for constraint optimization. The infeasibility values are represented by the sum of the normalized constraint violation values. The infeasibility measure has the properties that it increases in value with both the number of active constraints and the magnitude of each constraint violation. The infeasibility measure is used to form a two-stage dynamic "penalty" which is applied to the infeasible solutions. The penalty is applied such that slightly infeasible solutions having a low objective function value are allowed to remain fit. It is shown that this approach gives improved or comparable results to those of existing methods. The main advantages of the approach are first, that it is able to find the optimum or near optimum solution starting with a completely infeasible population of solutions. Second, the standard deviation in results over a number of trial runs is very low, which indicates that it is among the most robust in finding an optimum solution in a single run of the algorithm. Third, it is able to solve a range of constrained optimization problems, having both nonlinear equality and inequality constraints. The method is also applicable to test cases with discontinuous and continuous search spaces. Further work is required to investigate the application of the method to other types of constrained problem, such as constrained combinatorial problems.

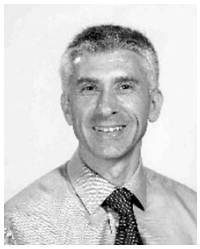
REFERENCES

[1] S. Ben Hamida and M. Schoenauer, "An adaptive algorithm for constrained optimization problems," in *Proc. Parallel Problem Solving from Nature*, vol. VI, 2000, pp. 529–538.
[2] F. Y. Cheng and D. Li, "Multiobjective optimization design with Pareto genetic algorithm," *J. Struct. Eng.*, vol. 123, no. 9, pp. 1252–1261, 1997.
[3] C. A. C. Coello, "Treating constraints as objectives for single-objective evolutionary optimization," *Eng. Opt.*, vol. 32, no. 3, pp. 275–308, 2000.
[4] D. W. Coit and A. E. Smith, "Penalty guided genetic search for reliability design optimization," *Comput. Ind. Eng., Special Issue on Genetic Algorithm*, vol. 30, pp. 895–904, 1996.
[5] D. W. Coit, A. E. Smith, and D. M. Tate, "Adaptive penalty methods for genetic optimization of constrained combinatorial problems," *INFORMS J. Comput.*, vol. 8, pp. 173–182, 1996.
[6] K. Deb, "An efficient constraint handling method for genetic algorithms," *Comput. Meth. Appl. Mech. Eng.*, vol. 186, pp. 311–338, 2000.
[7] K. Deb, A. Pratap, S. Agrawal, and T. Meyarivan, "A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II," Kanpur Genetic Algorithms Laboratory (KanGAL), Tech. Rep. 200 001, 2001.
[8] C. M. Fonseca and P. J. Fleming, "Multiobjective optimization and multiple constraint handling with evolutionary algorithms-Part I: A unified formulation," *IEEE Trans. Syst., Man, Cybern. A*, vol. 28, pp. 26–37, Jan. 1998.
[9] A. B. Hadj-Alouane and J. C. Bean, "A genetic algorithm for multiple-choice integer program," *Oper. Res.*, vol. 45, no. 1, pp. 92–101, 1997.
[10] P. Hajela and J. Yoo, "Constraint handling in genetic search – A comparative study," *AIAA/ASME/ASCE/AHS Structures, Structural Dynamics and Materials Conference—Collection of Technical Papers*, vol. 4, pp. 2176–2186, 1995.
[11] A. Homaifar, C. X. Qi, and S. H. Lai, "Constrained optimization via genetic algorithms," *Simulation*, vol. 62, no. 4, pp. 242–254, 1994.
[12] J. A. Joines and C. R. Houck, "On the use of nonstationary penalty functions to solve nonlinear constrained optimization problems with GA's," in *Proc. IEEE Conf. Evolutionary Computation*, vol. 2, 1994, pp. 579–584.
[13] S. Koziel and Z. Michalewicz, "Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization," *Evol. Comput.*, vol. 7, no. 1, pp. 19–44, 1999.
[14] J. Makinen, K. Miettinen, and M. M. Makela, "Some penalty methods with genetic algorithms," in *Proc. EUROGEN'99, Short Course on Evolutionary Algorithms in Engineering and Computer Science*, vol. A2, K. Miettinen, M. M. Makela, and J. Toivanen, Eds., 1999, pp. 105–112.

[15] Z. Michalewicz and N. F. Attia, "Evolutionary optimization of constrained problems," in *Proc. 3rd Annu. Conf. Evolutionary Programming*, A. V. Sebald and L. J. Fogel, Eds., River Edge, NJ, 1994, pp. 98–108.
[16] Z. Michalewicz and D. B. Fogel, *How to Solve It: Modern Heuristics*. Berlin, Germany: Springer-Verlag, 2000.
[17] Z. Michalewicz and C. Z. Janikow, "Handling constraints in genetic algorithms," in *Proc. Int. Conf. Genetic Algorithms*, vol. 4, 1991, pp. 151–157.
[18] Z. Michalewicz, G. Nazhiyath, and M. Michalewicz, "A note on usefulness of geometrical crossover for numerical optimization problems," in *Proc. 5th Annual Conf. Evolutionary Programming*, San Diego, CA, 1996, pp. 305–312.
[19] H. Myung and J. H. Kim, "Constrained optimization using two-phase evolutionary programming," *Proc. IEEE Int. Conf. Evolutionary Computation*, pp. 262–267, 1996.
[20] I. C. Parmee and G. Purchase, "The development of a directed genetic search technique for heavily constrained design spaces," in *Adaptive Computing in Engineering Design and Control-94*, I. C. Parmee, Ed. Plymouth, U.K.: Univ. Plymouth, 1994, pp. 97–102.
[21] D. Powell and M. M. Skolnick, "Using genetic algorithms in engineering design optimization with nonlinear constraints," in *Proc. 5th Int. Conf. Genetic Algorithms*, vol. 5, S. Forrest, Ed., 1993, pp. 424–431.
[22] J. T. Richardson, M. R. Palmer, G. Liepins, and M. Hilliard, "Some guidelines for genetic algorithms with penalty functions," in *Proc. Int. Conf. Genetic Algorithms*, vol. 3, 1989, pp. 191–197.
[23] T. P. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *IEEE Trans. Evol. Comput.*, vol. 4, pp. 284–294, Sept. 2000.
[24] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proc. 1st Int. Conf. Genetic Algorithms and their Applications*, vol. 1, J. J. Grefenstette, Ed., Hillsdale, NJ, 1985, pp. 93–100.
[25] M. Schoenauer and Z. Michalewicz, "Evolutionary computation at the edge of feasibility," in *Proc. 4th Conf. Parallel Problem Solving from Nature*, vol. 4, Berlin, Germany, Sept. 1996, pp. 245–254.
[26] M. Schoenauer and S. Xanthakis, "Constrained GA optimization," in *Proc. Int. Conf. Genetic Algorithms*, vol. 5, 1993, pp. 573–580.
[27] A. E. Smith and D. W. Coit, "Constraint handling techniques-Penalty functions," in *Handbook of Evolutionary Computation*, T. Baeck, D. Fogel, and Z. Michalewicz, Eds. London, U.K.: Oxford Univ. Press, 1997.
[28] P. D. Surry and N. J. Radcliffe, "The COMOGA method: Constrained optimization by multi-objective genetic algorithms," *Control Cybern.*, vol. 26, no. 3, pp. 391–412, 1997.
[29] J. Wu and S. Azarm, "On a new constraint handling technique for multi-objective algorithms," in *Proc. Genetic and Evolutionary Computation Conf.*, San Francisco, CA, July 7–11, 2001, pp. 741–748.
[30] J. A. Wright and R. Farmani, "Genetic algorithm: A fitness formulation for constrained minimization," in *Proc. Genetic and Evolutionary Computation Conf.*, San Francisco, CA, July 7–11, 2001, pp. 725–732.

Raziyeh Farmani received the Ph.D. degree in optimization of water distribution networks from Bradford University, Bradford, U.K., in 1999.

She is a Senior Research Fellow in the School of Engineering and Computer Science, Exeter University, Devon, U.K. She is a Member of the Centre for Water Systems (CWS), Exeter University. Her research interests are concerned with the application of numerical optimization techniques to the planning and operation of engineering systems. She has been involved in optimal design of different engineering applications including water systems, building services, and renewable energy systems. Her current research focuses on evolutionary based single and multiobjective optimization, hybrid and adaptive techniques, handling constraints in evolutionary algorithms and their engineering applications.



Jonathan A. Wright received the Ph.D. degree in building optimization from Loughborough University, Loughborough, Leicestershire, U.K., in 1986.

He is currently a Senior Lecturer in the Department of Civil and Building Engineering, Loughborough University. His early research focused on the application of direct search methods to the design of building thermal systems. His current research interests are focused on the development and application of evolutionary algorithms for solving highly constrained single and multicriterion mixed-integer problems, with particular focus on the solution of problems relating to the design and control of buildings.