Mathematical and Computational Applications, Vol. 10, No. 1, pp. 45-56, 2005.
© Association for Scientific Research

PENALTY FUNCTION METHODS FOR CONSTRAINED OPTIMIZATION WITH GENETIC ALGORITHMS

Özgür Yeniay
Hacettepe University, Faculty of Science
Department of Statistics, 06532,
Beytepe, Ankara
yeniay@hacettepe.edu.tr

Abstract- Genetic Algorithms are most directly suited to unconstrained optimization. Application of Genetic Algorithms to constrained optimization problems is often a challenging effort. Several methods have been proposed for handling constraints. The most common method in Genetic Algorithms to handle constraints is to use penalty functions. In this paper, we present these penalty-based methods and discuss their strengths and weaknesses.

Key Words- Genetic algorithms, Optimization, Constraint handling, Penalty function

1. INTRODUCTION

Genetic Algorithms (GAs) are stochastic optimization methods based on concepts of natural selection and genetics [11,14]. They work with a population of individuals, each representing a possible solution to a given problem. GAs typically work by iteratively generating and evaluating individuals using an evaluation function. The simplest of GAs work according to the scheme shown in Figure 1.

```
1. Initialize population of individuals (t)
2. Evaluate each individual using evaluation function (t)
3. Repeat until a stopping criterion is satisfied
   - Select parents from population (t)
   - Perform crossover on parents creating population (t+1)
   - Perform mutation on population (t+1)
   - Evaluate each individual of population (t+1)
```

Figure 1. The Steps of a Typical Genetic Algorithm

They have been applied to a wide range of problems in diverse fields such as engineering, mathematics, operations research etc. (for a variety of applications, see [11,29]). Most of the problems in these fields are stated as constrained optimization problems. Since GAs are directly applicable only to unconstrained optimization, it is necessary to use some additional methods that will keep solutions in the feasible region. During the past few years, several methods were proposed for handling constraints by GAs [6,7,23,25,36]. Most of these methods have serious drawbacks. While some of them may give infeasible solution or require many additional parameters, others are problem-dependent (i.e. specific algorithm has to be designed for each particular problem). The most popular approach in GA community to handle constraints is to use penalty functions that penalize infeasible solutions by reducing their fitness values in proportion to their degrees of constraint violation [30,36]. In this paper, we analyze these penalty-based methods.

The rest of this paper is organized as follows. In the second section, a definition of a constrained optimization problem is given. The third section gives constraint handling methods and a classification of them. The fourth section provides an introduction to penalty functions and explains penalty-based methods in more detail. The last section concludes the paper.

2. CONSTRAINED OPTIMIZATION

A constrained optimization problem is usually written as a nonlinear optimization problem of the following form:

$$\begin{array}{ll} \text{Min } f(\overline{x}) & \overline{x} = (x_1, \ldots, x_n)^t \in F \subseteq S \subseteq R^n \\ \text{subject to} & \\ g_i(\overline{x}) \leq 0 & i = 1, \ldots, q \\ h_j(\overline{x}) = 0 & j = q+1, \ldots, m \end{array} \tag{1}$$

where $\overline{x} = (x_1, \ldots, x_n)^t$ is the vector of solutions, F is the feasible region and S is the whole search space. There are q inequality and m-q equality constraints. $f(\overline{x})$ is usually called the objective function or criterion function. Objective function and constraints could be linear or nonlinear in the problem. Vector $\overline{x}$ that satisfies all the constraints is a feasible solution of the problem. All of the feasible solutions constitute the feasible region. Inequality constraints that satisfy $g_i(\overline{x}) = 0$ are called active at $\overline{x}$. Using these definitions, nonlinear programming problem is to find a point $\overline{x}^* \in F$ such that $f(\overline{x}^*) \leq f(\overline{x})$ for all $\overline{x} \in F$ [3].

3. CONSTRAINT HANDLING in GAs

There are several approaches proposed in GAs to handle constrained optimization problems. These approaches can be grouped in four major categories [28]:

Category 1: Methods based on penalty functions
- Death Penalty [2]
- Static Penalties [15,20]
- Dynamic Penalties [16,17]
- Annealing Penalties [5,24]
- Adaptive Penalties [10,12,35,37]
- Segregated GA [21]
- Co-evolutionary Penalties [8]

Category 2: Methods based on a search of feasible solutions
- Repairing unfeasible individuals [27]
- Superiority of feasible points [9,32]
- Behavioral memory [34]

Category 3: Methods based on preserving feasibility of solutions

- The GENOCOP system [26]
- Searching the boundary of feasible region [33]
- Homomorphous mapping [19]

Category 4: Hybrid methods [1,4,13,18,22]

Note that there are other classification schemes of constraint handling methods in GAs. As seen above even though many methods to handle constrained optimization problems within GAs have been proposed this paper will concentrate on penalty function methods.

3.1. Penalty Functions

Penalty method transforms constrained problem to unconstrained one in two ways. The first way is to use additive form as follows:

$$\mathrm{eval}(\bar{x}) = \begin{cases} f(\bar{x}), & \text{if } \bar{x} \in F \\ f(\bar{x}) + p(\bar{x}), & \text{otherwise} \end{cases} \tag{2}$$

where $p(\bar{x})$ presents a penalty term. If no violation occurs, $p(\bar{x})$ will be zero and positive otherwise. Under this conversion, the overall objective function now is $\mathrm{eval}(\bar{x})$ which serves as an evaluation function in GAs.

Second way is to use multiplicative form,

$$\mathrm{eval}(\bar{x}) = \begin{cases} f(\bar{x}), & \text{if } \bar{x} \in F \\ f(\bar{x})\,p(\bar{x}), & \text{otherwise} \end{cases} \tag{3}$$

For minimization problems, if no violation occurs $p(\bar{x})$ is one and bigger than one, otherwise.

The additive penalty type has received much more attention than the multiplicative type in the GA community.

In classical optimization, two types of penalty function are commonly used: interior and exterior penalty functions. In GAs exterior penalty functions are used more than interior penalty functions. The main reason of this, there is no need to start with a feasible solution in exterior penalty functions. Because finding a feasible solution in many GAs problems is a NP- hard itself. The general formulation of an exterior penalty function is [7],

$$\phi(\bar{x}) = f(\bar{x}) + \left[\sum_{i=1}^{q} r_i G_i + \sum_{j=q+1}^{m} c_j L_j \right] \tag{4}$$

where $\phi(\bar{x})$ indicates the new objective function to be optimized. G_i and L_j are the functions of $g_i(\bar{x})$ and $h_j(\bar{x})$ constraints respectively, and r_i ve c_j are penalty parameters. General formulas of G_i and L_j are,

$$G_i = \max[0,\ g_i(\bar{x})]^{\beta} \tag{5}$$

$$L_j = \left|h_j(\bar{x})\right|^{\gamma} \tag{6}$$

where β and γ are commonly 1 or 2. If the inequality is hold, $g_i(\bar{x}) \le 0$ and $\max[0, g_i(\bar{x})]$ will be zero. Therefore the constraint does not effect $\phi(\bar{x})$. If the constraint is violated that means $g_i(\bar{x}) > 0$ or $h_j \neq 0$, a big term will be added to $\phi(\bar{x})$ function such that the solution is pushed back towards to the feasible region. The severeness of the penalty depends on the penalty parameters r_i and c_j. If either the penalty is too large or too small, the problem could be very hard for GAs. A big penalty prevents to search unfeasible region. In this case GA will converge to a feasible solution very quickly even if it is far from the optimal. A pretty small penalty will cause to spend so much time in searching an unfeasible region; thus GA would converge an unfeasible solution [25].

Death Penalty

This simple and popular method just rejects unfeasible solutions from the population: $P(\bar{x}) = +\infty,\ \bar{x} \in S - F$. In this case, there will never be any unfeasible solutions in the population. If a feasible search space is convex or a reasonable part of the whole search space, it can be expected that this method can work well [30]. However, when the problem is highly constrained, the algorithm will spend a lot of time to find too few feasible solutions. Also, considering only the points that are in feasible region of the search space prevents to find better solutions.

Static Penalties

In the methods of this group, penalty parameters don't depend on the current generation number and a constant penalty is applied to unfeasible solutions.
Homaifar et al. [15] proposed a static penalty approach in which users describe some levels of violation. Steps of the method are as follows:

- Generate *l* levels of violation for each constraint,
- Generate penalty coefficient R_{ij} (i=1,...,*l* ; j=1,…,m) for each level of violation and each constraint. The bigger coefficients are given to the bigger violation levels,
- Generate a random population using both feasible and unfeasible individuals,
- Evaluate individuals using

$$\text{eval}(\bar{x}) = f(\bar{x}) + \sum_{j=1}^{m} R_{ij} \max[0, g_j(\bar{x})]^2 \tag{7}$$

where R_{ij} indicates the penalty coefficient corresponding to jth constraint and ith violation level. m is the number of constraints. Homaifar et al. [15] transformed equality constraints to inequality constraints by $\left|h_j(\bar{x})\right| - \varepsilon \le 0$ (where ε is a small positive number).

The disadvantage of this method is the large number of parameters that must be set. In this method, for m constraints it is needed to set m(2*l*+1) parameters in total. For instance, for m=5 constraints and *l*=4 levels of violation the total of 45 parameters should be set. Michalewicz [31] shows that the quality of solutions is very sensitive to the values of these parameters.

Kuri Morales and Quezada [20] suggested an another static penalty approach. In this method, individuals are evaluated using the following formula:

$$\text{eval}(\bar{x}) = \begin{cases} f(\bar{x}), & \text{if } \bar{x} \in F \\ K - \sum_{i=1}^{s} \left(\frac{K}{m} \right) & \text{otherwise} \end{cases} \tag{8}$$

where s is the number of non-violated constraints and m is the total number of constraints. Constraints can be equality or inequality. K is a large positive constant. Kuri Morales and Quezada [20] used this constant as $1\text{x}10^9$. There is only one restriction on K. K should guarantee that an unfeasible individual must be graded worse than a feasible individual. This approach uses information about the number of violated constraints, not the amount of constraint violation.

Dynamic Penalties

In the methods of this group, penalty parameters are usually dependent on the current generation number. Jones and Houck [16] suggested following dynamic function to evaluate individuals at each generation t:

$\text{eval}(\bar{x}) = f(\bar{x}) + (C\,t)^{\alpha}\ \text{SVC}(\beta, \bar{x})$

where C, α and β are constants determined by users. Joines and Houck [16] used C=0.5, α=1 or 2 and β=1 or 2. $\text{SVC}(\beta, \bar{x})$ is described as

$$\text{SVC}(\beta, \bar{x}) = \sum_{i=1}^{q} D_i^{\beta}(\bar{x}) + \sum_{j=q+1}^{m} D_j(\bar{x}) \tag{9}$$

where

$$D_i(\bar{x}) = \begin{cases} 0, & g_i(\bar{x}) \le 0,\ \ 1 \le i \le q \\ \left|g_i(\bar{x})\right|, & \text{otherwise} \end{cases} \tag{10}$$

$$D_j(\bar{x}) = \begin{cases} 0, & -\varepsilon \le h_j(\bar{x}) \le \varepsilon,\ \ q+1 \le j \le m \\ \left|h_j(\bar{x})\right|, & \text{otherwise} \end{cases} \tag{11}$$

This dynamic method increases the penalty as generation grows. The quality of a possible solution is very sensitive to changes of α and β values. There is no explanation about the sensitivity of the method for different values of C. Even Joines and Houck [16] stated that they used good selection of C=0.5 and α=β=2, Michalewicz [31] gave some examples to state that these parameter values cause premature convergence. He also showed that the method converges an unfeasible solution or a solution that is so far from an optimal solution.
Kazarlis and Petridis [17] used the following dynamic penalty function,

$$\mathrm{eval}(\overline{x}) = f(\overline{x}) + V(g)\left[A\sum_{i=1}^{m}(\delta_i W_i \phi(d_i(S))) + B \right]\delta_s \tag{12}$$

where
A=severity factor,
m=total number of constraints,

$$\delta_i = \begin{cases} 1, & \text{if constraint i is violated} \\ 0, & \text{otherwise} \end{cases}$$

W_i= weight factor for constraint i,
$d_i(S)$= measure of the degree of violation of constraint i introduced by solution S,
$\Phi_i(S)$ = function of this measure,
β= penalty threshold factor,

$$\delta_s = \begin{cases} 1, & \text{if S is feasible} \\ 0, & \text{otherwise} \end{cases}$$

V(g)= an increasing function of the current generation g in the range (0,..,1).

Kazarlis and Petridis [17] tried linear, quadratic and cubic forms of V(g) and received the best performance by

$$V(g) = \left(\frac{g}{G}\right)^2 \tag{13}$$

where G is the total number of generations. This method needs some parameters depending on the problem. It is very hard to determine these parameters. Kazarlis and Petridis [17] did not state why they took A=1000 and B=0 in their work.

Annealing Penalties

There are some methods based on annealing algorithm in this group.
Michalewicz and Attia [24] developed a method (GENECOP II) based on the idea of simulated annealing. Steps of this method are as follows:

- Separate all constraints into four subsets: linear equations, linear inequalities, nonlinear equations and nonlinear inequalities,

- Find a random single point that satisfies linear constraints and take it as a starting point. The initial population consists of copies of this single point,
- Create a set of active constraints A which consists of nonlinear equations and violated nonlinear inequalities,
- Set $\tau=\tau_0$,
- Evolve the population using,

$$\text{eval}(\overline{x},\tau) = f(\overline{x}) + \frac{1}{2\tau}\sum_{j\in A} f_j^2(\overline{x}) \tag{14}$$

- If $\tau<\tau_f$ (freezing point) stop the procedure, otherwise,
    - Diminish τ,
    - Use the best solution as a starting point for next generation,
    - Update A,
    - Repeat the previous step of the main part.

GENOCOP II distinguishes between linear and nonlinear constraints. In the algorithm only active constraints are under consideration at every iteration. While the temperature τ decrease, selective pressure on unfeasible solutions increases. As an interesting point of the method, there is no diversity of the initial generation that consists of multiple copies of a solution satisfied all linear constraints. At each generation the temperature τ decreases and the best solution found at the previous iteration is used as a starting point of next iteration. The algorithm is terminated at a previously determined freezing point τ_f.
The method is very sensitive to values of the parameters. There is no specific way how to decide these parameters for any particular problem. Michalewicz and Attia [24] used $\tau_0=1$, $\tau_{i+1}=0.1\tau_i$ and $\tau_f=0.000001$ in their experiments.
Carlson Skalak et al. [5] recommended to evaluate individuals using

$$\text{eval}(\overline{x}) = \alpha(M,T)\,f(\overline{x}) \tag{15}$$

In the formula, $\alpha(M,T)$ depends on the parameters M, the measurement of constraint violation, and temperature T, a function of running time of the algorithm. As execution proceeds, T approaches to zero. Carlson Skalak et al. [5] used the following function for $\alpha(M,T)$:

$$\alpha(M,T) = e^{-M/T} \tag{16}$$

Initial value of penalty parameter is small but increases over time so that unfeasible solutions found from last generations can be eliminated. For T,

$$T = \frac{1}{\sqrt{t}} \tag{17}$$

is used. Here, t indicates the last temperature used at the previous iteration.

Adaptive Penalties

In the methods of this group, penalty parameters are updated for every generation according to information gathered from the population.
Hadj-Alouane and Bean [12] evaluated an individual by following formula:

$$\text{eval}(\bar{x}) = f(\bar{x}) + \lambda(t)\left[\sum_{i=1}^{q} g_i^2(\bar{x}) + \sum_{j=q+1}^{m} \left|h_j(\bar{x})\right|\right] \tag{18}$$

Here penalty parameter λ(t) is updated at every generation k:

$$\lambda(t+1) = \begin{cases} \left(\dfrac{1}{\beta_1}\right)\lambda(t), & \text{if Case\#1} \\ \beta_2\lambda(t), & \text{if Case\#2} \\ \lambda(t), & \text{otherwise} \end{cases} \tag{19}$$

where Case # 1 denotes all of the best individuals in the last k generations are feasible, and Case # 2 denotes they are not feasible. In other words, if all best individuals of last k generations are feasible, penalty term λ(t+1) for generation (t+1) decreases. If they are unfeasible penalty term is increased. Otherwise, i.e. if the best individuals in the last k generations consist of feasible and unfeasible solutions, penalty term does not change. The main problem of this method is how to choose k, β_1 and β_2.
Smith and Tate [35] suggested to evaluate each individual by following formula:

$$\text{eval}(\bar{x}) = f(\bar{x}) + \left(B_{\text{feas}} - B_{\text{all}}\right)\sum_{i=1}^{q}\left(\frac{g_i(\bar{x})}{\text{NFT}(t)}\right)^k \tag{20}$$

where B_{all} is the value of the optimal objective function regardless of constraints, B_{feas} is the value of the optimal objective function satisfying constraints, $g_i(\bar{x})$ is the value of violation for constraint i, k is the severity parameter and NFT (called Near Feasibility Threshold) is the threshold distance from the feasible region F. Unfeasible solutions at this distance are considered to be reasonable since they are very close to the feasible region. Disadvantage of this method is how to choose NFT that is problem dependent
Yokota et al. [37] took the multiplication form as the evaluation function. Individuals are evaluated by the following formula:

$$\text{eval}(\bar{x}) = f(\bar{x})\left[1 - \frac{1}{q}\sum_{i=1}^{q}\left(\frac{\Delta b_i(\bar{x})}{b_i}\right)^k\right] \tag{21}$$

where $\Delta b_i(\bar{x})$ is the value of violation for constraint i . It is defined as follows,

$$\Delta b_i(\bar{x}) = \max[0, g_i(\bar{x}) - b_i] \tag{22}$$

For a feasible solution $g_i(\bar{x}) \le b_i$ is satisfied so for NFT=b_i this approach is a special case of the approach of Smith and Tate [35].
Gen and Cheng [10] improved the approach presented above by giving high penalties for unfeasible solutions. Evaluation function is as follows:

$$\mathrm{eval}(\bar{x}) = f(\bar{x})\left[1 - \frac{1}{q}\sum_{i=1}^{q}\left(\frac{\Delta b_i(\bar{x})}{\Delta b_i^{\max}}\right)^k\right] \tag{23}$$

In the formula above, $\Delta b_i(\bar{x})$ is the value by which the constraint i is violated in the individual $\bar{x}$. $\Delta b_i^{\max}$ is the maximum violation for constraint i among current population and ε is a small positive number. $\Delta b_i(\bar{x})$ and $\Delta b_i^{\max}$ are defined as follows:

$$\Delta b_i(\bar{x}) = \max[0, g_i(\bar{x}) - b_i] \tag{24}$$

$$\Delta b_i^{\max} = \max[\varepsilon, \Delta b_i(\bar{x});\ \bar{x} \in P(t)] \tag{25}$$

The purpose of this method is to protect the diversity of the population and to protect the unfeasible solutions, which form the big part of the population in the early generations, from high penalties. Gen and Cheng [10] claims that their approach is independent from the problem. However, this approach is only used in an optimization problem so their claim can be appreciated as suspicious.

Segregated GA

Le Riche et al. [21] developed a segregated GA (sometimes it is called as yet another method) that uses two penalty parameters (say, p_1 and p_2) in two different populations. The aim is to overcome the problem of too high and too low penalties (see section 3.1). If a small value is selected for p_1 and a large value for p_2, a simultaneous convergence from feasible and unfeasible sides can be achieved. The method works as follows:

- Generate 2 x pop individuals at random (pop is the population size),
- Design two different evaluation functions:
  $f_1(\bar{x}) = f(\bar{x}) + p_1(\bar{x})$ and $f_2(\bar{x}) = f(\bar{x}) + p_2(\bar{x})$
- Evaluate each individual according to two evaluation functions,
- Create two ranked lists according to $f_1(\bar{x})$ and $f_2(\bar{x})$,
- Combine two ranked lists in a single one ranked population with size pop,
- Apply genetic operators,
- Evaluate the new population using both penalty parameters,
- From the old and new populations generate two populations of the size pop each,
- Repeat the last four steps of the process.

Le Riche et al. [21] applied their method to laminated design problem and obtained excellent results. The way of choosing penalties for each of two populations is again the main problem in the method.

Co-evolutionary Penalties

Coello [8] developed a method of co-evolutionary penalties that split the penalty into two values, so that the GA has enough information about the number of constraint violations and the amounts of the violation. Each individual is evaluated by the formula:

$$\mathrm{eval}(\overline{x}) = f(x) - (\mathrm{coef}\ w_1 + \mathrm{viol}\ w_2) \qquad (26)$$

where $f(\overline{x})$ is the value of the objective function, w_1 and w_2 are two penalty parameters and coef is the sum of the amounts by which the constraints are violated. coef is defined as follows:

$$\mathrm{coef} = \sum_{i=1}^{q} g_i(\overline{x}), \qquad \forall g_i(\overline{x}) > 0 \qquad (27)$$

where initial value of viol is zero and increases by one for each constraint that is violated. This approach introduces the definition of four parameters. If they are not carefully chosen, a lot of evaluation function assessments may be required. Note that only inequality constraints are considered in this method.

4. CONCLUSIONS

Real-world optimization problems have constraints that must be satisfied by the solution of the problem. A variety of constraint handling methods have been suggested by many researchers Each method has its own advantages and disadvantages. The most popular constraint handling method among users is penalty function methods. In this paper we discussed these methods and gave their weakness and strengths. It is impossible to say one of the methods is the best for every problem. The main problem of the most methods is to set appropriate values of the penalty parameters. Consequently, users have to experiment with different values of penalty parameters.

5. REFERENCES

[1] Adeli, H. and Cheng, N.T. Augmented lagrangian genetic algorithm for structural optimization, *Journal of Aerospace Engineering*, 7, 104-118, 1994.

[2] Bäck, T., Hoffmeister, F. and Schwell, H.P. A Survey of evolution strategies, *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann, 2-9, 1991.

[3] Bazaraa, M.S., Sherali H.D. and Shetty, C.M. *Nonlinear programming: theory and algorithms*, John Wily and Sons, 2nd edition, 1993.

[4] Belur, S.V. CORE: Constrained optimization by random evolution, *Late Breaking Papers at the Genetic Programming Conference*, Stanford University, 280-286, 1997.

[5] Carlson Skalak, S., Shonkwiler, R., Babar, S., and Aral, M. Annealing a genetic algorithm over constraints, *SMC 98 Conference*, available from http://www.math.gatech.edu/~shenk/body.html.

[6] Coello, C.A.C. A Survey of constraint handling techniques used with evolutionary algorithms, Technical Report, *Lania-RI-99-04*, Laboratorio de Inform atica Avanzada, Veracruz, Mexico, 1999, available from http://www.cs.cinvestav.mx/~constraint/.

[7] Coello, C.A.C. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art, *Computer Methods in Applied Mechanics and Engineering*, 191, 1245-1287, 2002.

[8] Coello, C.A.C. Use of a self-adaptive penalty approach for engineering optimization problems, *Computers in Industry*, 41, 113-127, 2000.

[9] Deb, K. An Efficient constraint handling method for genetic algorithms, *Computer Methods in Applied Mechanics and Engineering*, 186, 311-338, 2000.

[10] Gen, M. and Cheng, R. A Survey of penalty techniques in genetic algorithms, *Proceedings of the 1996 International Conference on Evolutionary Computation*, IEEE, 804-809, 1996.

[11] Goldberg, D.E. *Genetic algorithms in search, optimization and machine learning*, Addison-Wesley, Reading, MA, 1989.

[12] Hadj-Alouane, A.B. and Bean, J.C. A Genetic algorithm for the multiple-choice integer program, *Operations Research*, 45, 92-101, 1997.

[13] Hajela, P. and Lee, J. Constrained genetic search via schema adaptation, An immune network solution, *Structural Optimization*, 12, 11-15, 1996.

[14] Holland, J.H. *Adaptation in naturel and artificial systems*, MIT Press, Cambridge, 1975.

[15] Homaifar, A., Lai, S.H.Y. and Qi, X. Constrained optimization via genetic algorithms, *Simulation*, 62, 242-254, 1994.

[16] Joines, J. and Houck, C. On the use of non-stationary penalty functions to solve non-linear constrained optimization problems with GAs, *Proceedings of the First IEEE International Conference on Evolutionary Computation*, IEEE Press, 579-584, 1994.

[17] Kazarlis S. and V. Petridis, Varying fitness functions in genetic algorithms: studying the rate of increase in the dynamic penalty terms, *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, Berlin, Springer Verlag, 211-220, 1998.

[18] Kim, J.H. and Myung, H. Evolutionary programming techniques for constrained optimization problems, *IEEE Transaction on Evolutionary Computation*, 1, 129-140, 1997.

[19] Koziel, S. and Michalewicz, Z. Evolutionary algorithms, homomorphous mapping and constrained parameter optimization, *Evolutionary Computation*, 7, 19-44, 1999.

[20] Kuri Morales, A. and Quezada, C.C. A Universal eclectic genetic algorithm for constrained optimization, *Proceedings 6th European Congress on Intelligent Techniques & Soft Computing, EUFIT'98*, 518-522, 1998.

[21] Le Riche, R., Knopf-Lenior, C. and Haftka, R.T. A Segregated genetic algorithm for constrained structural optimization, *Proceedings of the Sixth International Conference on Genetic Algorithms*, Morgan Kaufmann, 558-565, 1995.

[22] Le, T.V. Evolutionary approach to constrained optimization problems, *Proceedings of the Second IEEE International Conference on Evolutionary Computation*, IEEE Press, 274-278, 1995.

[23] Michalewicz, Z. A Survey of constraint handling techniques in evolutionary computation methods, *Proceedings of the 4th Annual Conference on Evolutionary Programming*, 135-155, 1995.

[24] Michalewicz, Z. and Attia, N. Evolutionary optimization of constrained problems, *Proceedings of the Third Annual Conference on Evolutionary Programming*, World Scientific, 98-108, 1994.

[25] Michalewicz, Z. and Fogel, D.B. *How to solve it: Modern Heuristics*, Springer Verlag, Berlin, 2000.

[26] Michalewicz, Z. and Janikow, C.Z. Handling constraints in genetic algorithms, *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann, 151-157, 1993.

[27] Michalewicz, Z. and Nazhiyath, G. GENOCOP III: A Co-evolutionary algorithm for numerical optimization problems with nonlinear constraints, *Proceedings of the Second IEEE International Conference on Evolutionary Computation*, IEEE Press, 647-651, 1995.

[28] Michalewicz, Z. and Schouenauer, M. Evolutionary algorithms for constrained parameter optimization problems, *Evolutionary Computation*, 4, 1-32, 1996.

[29] Michalewicz, Z. *Genetic algorithms + Data structures= Evolution programs*, Second Edition, Springer Verlag, Berlin, 1994.

[30] Michalewicz, Z., Dasgupta, D., Le Riche, R. and Schoenauer, M. Evolutionary algorithms for constrained engineering problems, *Computers & Industrial Engineering Journal*, 30, 851-870, 1996.

[31] Michalewicz, Z. Genetic algorithms, numerical optimization, and constraints, *Proceedings of the Sixth International Conference on Genetic Algorithms*, Morgan Kaufmann, 151-158, 1995.

[32] Powell, D. and Skolnick, M.M. Using genetic algorithms in engineering design optimization with non-linear constraints, *Proceedings of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann, 424-430, 1993.

[33] Schoenauer, M. and Michalewicz, Z. Evolutionary computation at the edge of feasibility, *Proceedings of the Fourth International Conference on Parallel Problem Solving from Nature*, Springer Verlag, 22-27, 1996.

[34] Schouenauer, M. and Xanthakis, S. Constrained GA optimization, *Proceedings of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann, 473-580, 1993.

[35] Smith, A. and Tate, D. Genetic optimization using a penalty function, *Proceedings of the fifth International Conference on Genetic Algorithms*, Morgan Kaufmann, 499-503, 1993.

[36] Smith, A.E. and Coit, D.W. Constraint handling techniques- penalty functions, *Handbook of Evolutionary Computation*, chapter C 5.2. Oxford University Press and Institute of Physics Publishing, 1997.

[37] Yokota, T., Gen, M., Ida, K. and Taguchi, T. Optimal design of system reliability by an improved genetic algorithm, *Electron. Commun. Jpn. 3, Fundam. Electron. Sci.*79, 41-51,1996.