Constraint Handling Methods in Genetic Algorithms

Spyros A. Kazarlis

Dept. of Informatics & Communications, T.E.I. of Serres, Greece

Abstract

Real-world optimisation problems are often subject to constraints that must be satisfied by the optimal solution. Genetic Algorithms, although being powerful optimisation tools, they do not incorporate constraint-handling features. This is the reason why a large number of methods have been proposed in the literature to rectify this inefficiency. In this paper we present the most significant of these techniques, divided into categories. Simulation results are given for a number of constraint handling GA methods on a common test set of 5 constrained optimisation problems.

Keywords: Genetic Algorithms, Constrained Optimization

(This work is co-funded by the European Social Fund and National Resources - (EPEAEK-II) ARXIMHDHS)

1. Introduction

Optimisation tasks in real-world problems are often subject to a number of problem constraints. A general constrained optimisation problem can be formulated as follows:

Optimise the objective function $F(\vec{x})$, $\vec{x} \in \mathrm{S}$, where $\vec{x}$ is a generalized vector of problem parameters, and S is the solution search space, subject to the following inequality constraints:

$$IC_i(\vec{x}) \le 0, \quad \text{i=1..p,} \tag{1}$$

and the following equality constraints:

$$EC_j(\vec{x}) = 0, \quad \text{j=1..q,} \tag{2}$$

In the special case where $\vec{x} = (x_1, x_2, ..., x_n) \in \Re^n$ ($S \subseteq \Re^n$), we have the general nonlinear programming problem (NP).

The general constrained optimisation problem is intractable [Michalewicz et.al.(1996)], i.e. no global algorithm (other than exhaustive search) has been found yet to provide the optimum solution for every conceivable type of the objective function $F(\vec{x})$ and the constraint functions $IC_i(\vec{x})$ and $EC_j(\vec{x})$. This leaves the door open for innovation and application of a great variety of optimisation methods, that attempt to solve the above general problem formulation.

Genetic Algorithms (GAs) define a class of powerful global optimisation algorithms with wide applicability and a large number of successful applications. However, they lack the ability to handle problem constraints. For this reason, a number of methods have been proposed in the literature to cover this inefficiency. The proposed methods vary significantly in the way they try to attack the problem constraints and they are classified in a number of categories.

This work presents the most significant methods of the literature for handling problem constraints in GAs [Carlos A. Coello Coello (2002)]. One such technique is the Varying Fitness Function Technique (VFF) [Kazarlis et.al.(1996)], [Kazarlis et.al.(1998)], [Petridis et.al.(1994)], [Petridis et.al.(1998)], that has been proposed by the author. In this work we also present simulation results of a total of 13 different GA-based constraint handling methods on a test set of 5 constrained optimisation problems.

Section 2 presents the most significant constraint handling methods for GAs. The 13 constraint handling methods (and their implementations) for which results are presented are discussed in Section 3. Section 4 describes the five test problems. The results are reported in Section 5 and finally conclusions are presented in Section 6.

2. Constraint Handling Methods

The proposed constraint handling methods for GAs can be divided into the following categories:

a) Rejection of unfeasible solutions.

b) Approximation of unfeasible solutions with feasible ones (or repair),

c) Penalty function methods

d) Methods that use special phenotype-to-genotype representations (decoders)

e) Methods that use special recombination and permutation operators

f) Methods based on parent selection strategies.

g) Methods that use two-phase strategies for satisfying the constraints and optimising the objective function.

h) Multi objective optimisation methods.

i) Co-evolutionary methods that maintain more than one populations.

j) Complex hybrid methods

k) Methods based on novel approaches.

2.1 Category (a): Rejection of unfeasible solutions

During the reproduction phase of the GA, any unfeasible offspring that is produced is immediately rejected and the reproduction is repeated until it delivers a feasible solution. This method is quite simple but, in highly constrained spaces, a lot of CPU time is consumed in the effort of finding feasible solutions [Michalewicz (1992)].

2.2 Category (b): Approximation of unfeasible solutions with feasible ones

Methods of this category either approximate an invalid solution by its nearest valid one, or repair it to become a valid one [Michalewicz et.al.(1995)].

In the work of D.Orvosh & L.Davis [Orvosh et.al.(1994)] the authors deal with three constrained combinatorial optimisation problems, and they use repairing mechanisms to transform illegal solutions to legal ones before evaluating them. Additionally, they examine the application of the "forcing" technique, i.e. the replacement of illegal genotypes with their repaired counterparts.

2.3 Category (c): Penalty Function Methods

Methods of this category add penalty terms to the fitness function that are related to the violation of constraints. According to this principle, the fitness function has the following general form:

$$Q(\vec{x}) = \begin{cases} F(\vec{x}), \text{ if } \vec{x} \text{ is a feasible solution} \\ F(\vec{x}) + P(\vec{x}), \text{ if } \vec{x} \text{ is unfeasible} \end{cases} \quad \textbf{(3)}$$

where $P(\vec{x})$ is a penalty function depending on the degree of constraint violation.

This method is probably the most commonly used method for handling constraints and is implemented in many variations that can be divided in two sub-categories: (1) static penalty methods and (2) varying/adaptive penalty methods, where penalties are time-variant.

Sub-category (c-1): Static penalty methods.

An example of this category is the work of A. Homaifar, S. Lai and X. Qi [Homaifar et.al.(1994)]. This method first creates several (ℓ) levels of violation for every problem constraint j (j=1..n). For each level of violation and for each constraint, a penalty coefficient is created R_{ij} (i=1..ℓ, j=1..n), following the principle that higher levels of violation require larger values of the penalty coefficient. Then the fitness function is given by: $Q(\vec{x}) = F(\vec{x}) + \sum_{j=1}^{p+q} R_{ij} \cdot d_j^2(\vec{x})$, where $d_j(\vec{x})$ is a measure of violation of the j^{th} constraint.

Another method is the static penalties of D. Powell and M. Skolnick [Powell et.al.(1993)] that makes a clear distinction between feasible and unfeasible solutions, based on their fitness values, by adopting the rule that for any feasible solution $\vec{x}_f$ and any unfeasible solution $\vec{x}_u$, it must be $Q(\vec{x}_f) < Q(\vec{x}_u)$ (minimization problem). This means that all unfeasible solutions are ranked worse than the worst feasible one. This is ensured by the following fitness function: $Q(\vec{x}) = F(\vec{x}) + r \cdot \sum_{i=1}^{p+q} d_i(\vec{x}) + \lambda(g, \vec{x})$, where r is a constant and $\lambda(g, \vec{x})$ is an iteration dependent function that influences the evaluation of unfeasible solutions so as to maintain the separation of feasible/unfeasible fitness values.

Sub-category (c-2): Varying/Adaptive penalty methods.

In order to overcome the problem of determining the suitable level of penalties for the specific application, many researchers have proposed non-stationary (varying / adaptive) penalty function methods:

The work of J.A. Joines and C.R. Houck (1994) [Joines et.al.(1994)]. In this work the authors propose a fitness function of the form: $Q(\vec{x}, g) = F(\vec{x}) + (C \cdot g)^a \cdot \sum_{i=1}^{n} d_i^{\beta}(\vec{x})$, where C is a constant, g is the generation

index, α and β are "severity" exponent factors, n is the number of problem constraints and $d_i(\vec{x})$, i=1..n, are the measures of violation of the constraints.

The work of V.Petridis and S.Kazarlis [Petridis et.al.(1994)], where the authors deal with a Cutting Stock problem where the constraint is that no shape overlapping is allowed. The authors use a varying fitness function of the form: $Q(\vec{x},g) = F(\vec{x}) + \frac{g}{G} \cdot (A \cdot \Phi(d(\vec{x})) + B) \cdot \delta_{\vec{x}}$, where $F(\vec{x})$ is the objective function, g is the current generation index, G is the generation limit of the GA, A is a "severity" penalty factor, Φ(.) is a nonnegative penalty function, $d(\vec{x})$ is a measure of the constraint violation (shape overlapping), B is a penalty threshold and $\delta_{\vec{x}}$ is a binary coefficient ($\delta_{\vec{x}}$=0 for feasible and $\delta_{\vec{x}}$=1 for unfeasible solutions). This is a linearly varying fitness function (VFF) that keeps the penalty level low at the first stages of search and increases it linearly during the run, reaching appropriately large values at the end of the GA evolution.

Also in the work of S. Kazarlis, A. Bakirtzis and V. Petridis [Kazarlis et.al.(1996)] the authors deal with the problem of Unit Commitment (power systems) and use a similar varying fitness function of the form: $Q(\vec{x},g) = F(\vec{x}) + \frac{g}{G} \cdot (A \cdot \sum_{i=1}^{n} d_i(\vec{x}) + B) \cdot \delta_{\vec{x}}$, where n is the number of problem constraints, $di(\vec{x})$ is a measure of the constraint violation of the i^{th} constraint, and the other coefficients are the same as in the previous paragraph.

The test results reported in the above mentioned works (Category c-2) generally show a superiority of the varying/adaptive penalty schemes over the corresponding static penalty schemes that they have been compared with.

2.4 Category (d): Methods that use special phenotype-to-genotype representations (decoders) to encode only the feasible part of the search space.

Methods of this category use special phenotype-to-genotype representation schemes (decoders), that minimize or eliminate the possibility of producing unfeasible solutions through the standard genetic operators, Crossover and Mutation [Michalewicz (1992)]. In the work of S. Koziel and Z. Michalewicz [Koziel et.al.(1999)] a "homomorphous mapping" method is proposed that maps the feasible area F of the search space into a hyper-cube $[-1,1]^n$. This method is easily applied when the feasible area is convex, but in nonconvex or disjoint feasible spaces it uses an extended mapping method that maps the feasible parts of the space into disjoint intervals. This method is unable to handle equality constraints of the form $EC(\vec{x})$=0.

Thus it replaces them with two inequalities of the form $IC_1(\vec{x})<=\delta$ and $IC_2(\vec{x})>=-\delta$, where δ is a small positive value.

2.5 Category (e): Methods using special recombination and permutation operators that preserve feasibility

In this category special problem-specific recombination and permutation operators are designed, similar to traditional crossover and mutation, that produce only feasible solutions [Michalewicz et.al.(1994)], [Michalewicz et.al.(1995)]. We should mention the Genocop method of Z. Michalewicz [Michalewicz (1992)] where the feasibility of solutions regarding the linear constraints is preserved by the use of special "closed" operators. The special mutation operator first calculates the feasible domain around the specific solution and then performs the mutation within that domain, while the crossover operator is an arithmetic crossover of the form: $\vec{o} = a\vec{p}_1 + (1-a)\vec{p}_2$, where $\vec{o}$ is the offspring and $\vec{p}_1$, $\vec{p}_2$ the two parents and $0 \leq \alpha \leq 1$. The above crossover always produces a feasible solution.

2.6 Category (f): Methods that are based on parent selection strategies

This category includes methods that implement sophisticated parent selection strategies in order to promote the optimization of the objective function as well as the satisfaction of constraints. These methods treat the objective quality and the feasibility of solutions as two different metrics of their overall fitness and use parent selection methods that select individuals according to either their objective quality or their feasibility.

In the work of F. Jimenez and J. L. Verdegay [Jimenez et.al.(1999)], the "optimality" and the "unfeasibility" of solutions are treated as two different measures and a tournament selection method is used that selects parents according to the following three rules: (1) two feasible individuals are compared based on their objective fitness and the best is selected, (2) a feasible solution is always preferred over an unfeasible one, and (3) when two unfeasible solutions are compared, the maximum constraint violation is calculated for both solutions and the one with the lower constraint violation wins.

2.7 Category (g): Methods that use two-phase strategies for satisfying the constraints and optimising the objective function

Methods of this category [Hinterding et.al.(1998)], [Schoenauer et.al.(1993)] first concentrate on finding feasible solutions, and then they try to optimise the objective function preserving the solution feasibility. This category includes the Behavioral Memory Model method proposed by M. Schoenauer and S. Xanthakis [Schoenauer

et.al.(1993)] that, before optimising the objective function, it evolves a population of solutions trying to find the feasible area of the search space. Another method of this category is the CONGA method of R. Hinterding and Z. Michalewicz [Hinterding et.al.(1998)] that uses sophisticated parent matching techniques to select parent genotypes in order to promote the generation of offspring that first satisfy the constraints and later optimise the objective function in a two-phase strategy.

2.8 Category (h): Multi objective optimisation methods

The optimization of an objective function $F(\vec{x})$ that is subject to n=p+q constraints can be seen as a multi objective optimization problem (MOOP) where the objective function $F(\vec{x})$ and the n measures of constraint violation $d_i(\vec{x})$, i=1..n, constitute a n+1 dimensioned vector $\vec{U}$ = ($F(\vec{x})$, $d_1(\vec{x})$, $d_2(\vec{x})$, ... , $d_n(\vec{x})$), each member of which must be optimised (minimized for $d_i(\vec{x})$, i=1..n). This formulation allows the application of any multi objective optimization method [Fonseca et.al.(1995)] to the area of constrained optimization problems. Some classical multi objective optimisation methods create an aggregating optimisation function $Q_{agg}(\vec{x})$ that usually is a combination (linear or not) of the elements of vector $\vec{U}$ (e.g. $Q_{agg}(\vec{x}) = w_o \cdot F(\vec{x}) + \sum_{i=1}^{n} w_i \cdot d_i(\vec{x})$, where w_i, i=0..n are weight coefficients). This approach resembles the various penalty approaches mentioned in category (c). Other multi objective optimization techniques focus on creating sophisticated parent selection strategies to promote the simultaneous optimization of each of the objectives within the evolving population. Some of these selection strategies aim in producing certain fractions of the offspring population by selecting parents according to each one of the objectives, separately. Other methods use Pareto-based [Fonseca et.al.(1995)] fitness assignment mechanisms that are based on Pareto-optimality. Such methods assign ranks to individuals in a population according to whether they are dominated by other individuals (i.e. if other individuals exist that have better objective values concerning all of the objectives) or not.

In the work of D. Schaffer [Schaffer (1985)] a Vector Evaluated GA (VEGA) is proposed that selects 1/(n+1) of the parents based on each of the objectives.

In the work of P. D. Surry et.al. [Surry et.al.(1995)] all members of the population are ranked on the basis of constraint violation with a penalty value $r(\vec{x})$. The penalty $r(\vec{x})$, together with the value of the objective function $F(\vec{x})$, leads to the two-objective optimisation problem.

2.9 Category (i): Co-evolutionary methods that maintain more than one populations.

In this category we should report the Genocop III method introduced by Z. Michalewicz and G. Nazhiyath [Michalewicz et.al.(1995)]. This method maintains and evolves two populations: one with search points that are produced genetically and may be feasible or unfeasible, and another with reference feasible points selected from the previous population. The method repairs the unfeasible solutions of the first population using reference points from the second population.

2.10 Category (j): Complex hybrid methods

Methods of this class combine one or more of the above methods together with calculus-based or other methods. A representative example is that of the method called Genocop II, introduced by Z. Michalewicz and N. Attia [Michalewicz et.al.(1994)]. This method combines traditional calculus-based optimisation methods together with GAs and a meta-level Simulated Annealing scheme, for the solution of optimisation problems with linear and nonlinear constraints.

2.11 Category (k): Methods based on novel approaches

One of the methods of this category is Cultural Algorithms [Reynolds (1994)]. Cultural Algorithms are based on the notion of culture, being another form of inheritance system. They maintain a "belief space" that is used to constrain the combination of characteristics that the population members can have. The belief space is associated with the "acceptable" individuals' characteristics (or behavior) and is used as a constraint to guide the search for characteristics with good qualities.

Also belonging in this category is the method of P. Hajela and J. Lee [Hajela et.al.(1996)] who use an Artificial Immune System model. In their implementation the authors separate the feasible genotypes of the population (antigens) from the unfeasible ones (antibodies). Then, they separately evolve the sub-population of unfeasible genotypes in order to maximize the similarity between them and the feasible ones. Then the two populations are mixed, and, since all genotypes are now feasible, they are evolved using a standard genetic algorithm.

Another novel approach is the method proposed by G. Bilchev and I. Parmee [Bilchev et.al.(1996)] who use an Ant Colony search model. This model is applicable to continuous search spaces by representing a finite number of search directions from a base point (the nest) as vectors that evolve in time according to the ants' fitness. The model works on three levels: the individual search (e.g. stochastic hill climbing), the cooperation between agents (ants) and the meta-cooperation level between search paths. In order to handle constraints, the authors add the feature of "acceptability" to the "food sources" (highly fit solutions) that guide the evolution of the ants. The acceptability of a "food source" decreases as the amount of constraint violation increases.

3. Constraint Handling Methods Used for the Comparison Test

For presenting comparative results of various constraint handling techniques we consulted the work of Z. Michalewicz and G. Nazhiyath [Michalewicz et.al.(1995)] and the work of R. Hinterding and Z. Michalewicz [Hinterding et.al.(1998)]. The nine methods compared in these works are:

1. The static penalty method of A. Homaifar, S. Lai and X. Qi, (cat. c-1)
2. The method of varying penalties of J.A.Joines and C.R.Houck (cat. c-2).
3. The method of M. Schoenauer and S. Xanthakis (cat. g).
4. The Genocop II method by Z. Michalewicz and N. Attia (cat. j).
5. The method of static penalties of D. Powell and M. Skolnick (cat. c-1).

6a. The method of rejecting the unfeasible solutions (death penalty) (cat. a)

6b. The same method but starting with a feasible population (cat. a).

7. The method Genocop III of Z. Michalewicz and G. Nazhiyath (cat. i).
8. The CONGA method of R. Hinterding and Z. Michalewicz (cat. g).

The population was 70 genotypes in all cases and the generation limit was 5000 generations.

To test the performance of the VFF technique we have built implementations for two more methods:

9. A GA using a static penalty function of the form : $Q(\vec{x}) = F(\vec{x}) + P(\vec{x}) = F(\vec{x}) + \left(A \cdot \sum_{i=1}^{p+q} \delta_i \cdot w_i \cdot \Phi(d_i(\vec{x})) + B \right) \cdot \delta_{\vec{x}}$, where A is a "severity" factor, δ_i is a binary factor (δ_i=1 if constraint i is violated and δ_i=0 otherwise), w_i is a "weight" factor for constraint i, $d_i(\vec{x})$ is a measure of the degree of violation of constraint i, $\Phi_i(.)$ is a function of this measure, B is a penalty threshold and $\delta\vec{x}$ is a binary factor ($\delta\vec{x}$=1 if $\vec{x}$ is unfeasible and $\delta\vec{x}$=0 otherwise).Two different implementations are built for this method. The first one (9a) is a simple Genetic Algorithm (sGA). The second (9b) is a GA with a Hill Climbing operator (GA-HC). This implementation incorporates all of the features of sGA plus an additional hill climbing operator called Phenotype Mutation [Petridis et.al.(1994)], [Petridis et.al.(1998)].

10. A GA with the VFF technique. Two implementations are also built for this method. The first one (10a) is a simple GA with the VFF technique (GA-VFF). The second (10b) is a GA with Hill Climbing and the VFF technique (GA-HC-VFF). This

implementation incorporates all of the features of the GA-HC plus the application of the Varying Fitness Function technique.

4. The five Problems of the Comparison Test

In order to perform the comparison test, we have chosen five test problems. They were specifically chosen because they comprise a maximum common set of problems for which simulation results have already been published regarding methods 1 through 8 of the previous Section. The five test cases were the following:

Test Case 1. Minimize $G1(X) = 5x_1 + 5x_2 + 5x_3 + 5x_4 - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$ **(4)**

with variable ranges: $0 \le x_i \le 1$, i=1…9, $0 \le x_i \le 100$, i=10,11,12, $0 \le x_i \le 1$, i=13

and 9 constraints:

$2x_1+2x_2+x_{10}+x_{11} \le 10$

$2x_1+2x_3+x_{10}+x_{12} \le 10$

$2x_2+2x_3+x_{11}+x_{12} \le 10$

$-8x_1+x_{10} \le 0$

$-8x_2+x_{11} \le 0$

$-8x_3+x_{12} \le 0$

$-2x_4-x_5+x_{10} \le 0$

$-2x_6-x_7+x_{11} \le 0$

$-2x_8-x_9+x_{12} \le 0$

Test Case 2. Minimize $G2(X) = x_1 + x_2 + x_3$ **(5)**

with variable ranges: $10^2 \le x_1 \le 10^4$, $10^3 \le x_i \le 10^4$, i=2,3, $10 \le x_i \le 10^3$, i=4..8 ,

and 6 constraints:

$1-0.0025(x_4+x_6) \ge 0$

$1-0.0025(x_5+x_7-x_4) \ge 0$

$1-0.01(x_8-x_5) \ge 0$

$x_1x_6-833.33252x_4-100x_1+83333.333 \ge 0$

$x_2x_7-1250x_5-x_2x_4+1250x_4 \ge 0$

x_3x_8-1250000-x_3x_5+2500$x_5 \geq 0$

Test Case 3. Minimize the function:

$G3(X)=(x_1-10)^2+5(x_2-12)^2+x_3^4+3(x_4-11)^2+10x_5^6+7x_6^2+x_7^4-4x_6x_7-10x_6-8x_7$ **(6)**

with variable ranges: $-10 \leq x_i \leq 10$, i=1..7 ,

and 4 constraints:

$127 - 2x_1^2 - 3x_2^4 - x_3 - 4x_4^2 - 5x_5 \geq 0$

$282 - 7x_1 - 3x_2 - 10x_3^2 - x_4 + x_5 \geq 0$

$196 - 23x_1 - x_2^2 - 6x_6^2 + 8x_7 \geq 0$

$-4x_1^2 - x_2^2 + 3x_1x_2 - 2x_3^2 - 5x_6 + 11x_7 \geq 0$

Test Case 4. Minimize $G4(X) = e^{x_1x_2x_3x_4x_5}$ **(7)**

with variable ranges: $-2.3 \leq x_i \leq 2.3$, i=1,2 , $-3.2 \leq x_i \leq 3.2$, i=3,4,5

and 3 constraints:

$x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 = 10$

$x_2x_3 - 5x_4x_5 = 0$

$x_1^3 + x_2^3 = -1$

Test Case 5. Minimize the function:

$G5(X) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$ **(8)**

with variable ranges: $-10 \leq x_i \leq 10$, i=1..10 ,

and 8 constraints:

$105 - 4x_1 - 5x_2 + 3x_7 - 9x_8 \geq 0$

$-10x_1 + 8x_2 + 17x_7 - 2x_8 \geq 0$

$8x_1 - 2x_2 - 5x_9 + 2x_{10} + 12 \geq 0$

$-3(x_1 - 2)^2 - 4(x_2 - 3)^2 - 2x_3^2 + 7x_4 + 120 \geq 0$

$-5x_1^2 - 8x_2 - (x_3 - 6)^2 + 2x_4 + 40 \geq 0$

$-x_1^2 - 2(x_2 - 2)^2 + 2x_1x_2 - 14x_5 + 6x_6 \geq 0$

$-0.5(x_1 - 8)^2 - 2(x_2 - 4)^2 - 3x_5^2 + x_6 + 30 \geq 0$

$3x_1 - 6x_2 - 12(x_9 - 8)^2 + 7x_{10} \geq 0$

5. Simulation Results and Comparisons

The test results of methods 1 through 6b of Section 3 on the five problems of Section 4 are shown in Table 1. Also the test results of methods 7 through 10b of Section 3 are reported in Table 2.

For methods 1 through 7, 10 experiments were performed for each problem. For method 8, 20 experiments were performed, and for methods 9a – 10b 50 experiments were performed for each problem.

In both tables for every problem the best (b), median (m) and worst (w) solution found is shown, together with the number (c) of violated constraints at the median solution. The three numbers displayed in the (c) rows of Table II are the number of constraints violated by a quantity between 1.0 and 10, 0.1 and 1.0, and 0.001 and 0.1 respectively. For methods 9a through 10b the percentage of feasible solutions is shown instead. The symbol '*' means that the method was not applied on the specific test case and the symbol '—' means that the solution produced was not meaningful (the constraints were violated by a quantity more than 10).

Among methods 1 through 8 the ones that exhibit the best overall performance are the CONGA method (R. Hinterding and Z. Michalewicz, method 8), Genocop II method (Z. Michalewicz and N. Attia, method 4), the Genocop III method (Z. Michalewicz and G. Nazhiyath, method 7) and the GA-HC-VFF method (S.Kazarlis and V. Petridis, method 10b).

***Table 1**. Results of 7 methods on 5 problems. Method numbering corresponds to that of Section 3*

test case	Exact optima		Method 1	Method 2	Method 3	Method 4	Method 5	Method 6a	Method 6b
1	-15.00	b	-15.002	-15.000	-15.000	-15.000	-15.000		-15.000
		m	-15.002	-15.000	-15.000	-15.000	-15.000	—	-14.999
		w	-15.001	-14.999	-14.998	-15.000	-14.999		-13.616
		c	0, 0, 4	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0		0, 0, 0
2	7049.33	b	2282.72	3117.24	7485.66	7377.97	2101.36		7872.95
		m	2449.79	4213.49	8271.29	8206.15	2101.41	—	8559.42
		w	2756.68	6056.21	8752.41	9652.90	2101.55		8668.65
		c	0, 3, 0	0, 3, 0	0, 0, 0	0, 0, 0	1, 2, 0		0, 0, 0
3	680.63	b	680.771	680.787	680.836	680.642	680.805	680.934	680.847
		m	681.262	681.111	681.175	680.718	682.682	681.771	681.826
		w	689.660	682.798	685.640	680.955	685.738	689.442	689.417
		c	0, 0, 1	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
4	0.05395	b	0.084	0.059		0.054	0.067		
		m	0.955	0.812	*	0.064	0.091	*	*
		w	1.000	2.542		0.557	0.512		
		c	0, 0, 0	0, 0, 0		0, 0, 0	0, 0, 0		
5	24.3062	b	24.690	25.486		18.917	17.338		25.653
		m	29.258	26.905	—	24.418	22.932	—	27.116
		w	36.060	42.358		44.302	48.866		32.477
		c	0, 1, 1	0, 0, 0		0, 1, 0	1, 0, 0		0, 0, 0

For test case 1 all methods find the same optimal solution. For test case 2 the best solution is found by method 8 (CONGA) while the second best is found by GA-HC-VFF. For test case 3, method 7 (Genocop III) finds the overall best solution, followed by method 8 (CONGA). For test case 4, the best solution is found by GA-HC-VFF, followed by method 8 (CONGA) and method 4 (Genocop II). And finally for test case 5, the best solution is also found by the GA-HC-VFF method while method 8 (CONGA) comes second best.

From these results it is evident that some methods seem to clearly outperform others, exhibiting better performance on diverse problems. It is also evident that among the best methods one cannot clearly distinguish a winner, as some of them exhibit better performance on some problems and worse on others.

TableII. *Results of 6 more methods on the 5 problems. Method numbering corresponds to that of Section 3*

test case	Exact optima		Method 7	Method 8	Method 9a sGA	Method 9b GA-HC	Method 10a GA-VFF	Method 10b GA-HC-VFF
1	-15.00	b	-15.000	-15.000	-15.00	-15.00	-15.0	-15.00
		m	-15.000	-15.00	-15.00	-15.00	-15.0	-15.00
		w	-15.000		-15.00	-15.00	-15.0	-15.00
		c	0, 0, 0		100%	100%	100%	100%
2	7049.33	b	7286.65	7083.21	7529.0478	7270.5369	7248.3587	7115.9982
		m		7804.33	10080.1641	8936.8215	9040.7122	9002.5598
		w			17085.2636	12983.4040	12561.9301	11944.3052
		c			78%	90%	72%	90%
3	680.63	b	680.640	680.65	680.805677	680.72180	680.73259	680.676424
		m		680.72	683.813684	682.74501	682.47911	681.4013675
		w	680.889		689.413076	688.59298	688.36369	682.4630019
		c			100%	100%	100%	100%
4	0.05395	b		0.054	0.0616218	0.0541294	0.0540426	0.0539762
		m	*	0.054	0.617415	0.2450273	0.0805291	0.0756711
		w		0.054	1.000000	0.7632619	0.2201789	0.2263098
		c			90%	92%	98%	98%
5	24.3062	b	25.883	24.44	26.699057	24.966805	24.618510	24.403088
		m		25.61	41.180946	35.489303	27.335638	27.460453
		w			74.121369	65.504347	30.802398	30.020087
		c			100%	100%	100%	100%

6. Conclusions

In this paper, a large number of methods, reported in the literature, have been presented for handling constraints, when applying GAs on constrained optimization problems. Also test results have been presented for a total of 13 different methods on a test suite of 5 problems. From the test results it is shown that some methods like CONGA and GA-HC-VFF stand out from the crowd, exhibiting robust performance on all problems. This work could be enhanced by implementing more methods and testing them on a wider range of constrained problems.

References

Bilchev G. and Parmee I.C.(1996), *Constrained and Multi-Modal Optimisation with an Ant Colony Search Model* in Proceedings of the 2nd International Conference on Adaptive Computing in Engineering Design and Control (ACEDC'96), I.C. Parmee and M.J. Denham (eds.), University of Plymouth, Plymouth, UK, pp. 145-151.

Carlos A. Coello Coello (2002), *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, Vol. 191, No. 11-12, pp. 1245-1287, January 2002

Fonseca C.M. and Flemming P.J.(1995), *An Overview of Evolutionary Algorithms in Multiobjective Optimization* Evolutionary Computation, Vol. 3, No. 1, pp. 1-16.

Hajela P. and Lee J.(1996), *Constrained Genetic Search via Schema Adaptation. An Immune Network Solution* Structural Optimization, vol 12, no 1, pp. 11-15.

Hinterding R. and Michalewicz Z.(1998), *Your Brains and My Beauty: Parent Matching for Constrained Optimisation* in Proceedings of the 5th IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, May 4-9, pp. 810-815.

Homaifar A., Lai S., and Qi X.(1994), *Constrained Optimization via Genetic Algorithms* Simulation, vol. 62, no. 4, pp. 242-254.

Jimenez F. and Verdegay J.L.(1999), *Evolutionary Techniques for Constrained Optimization Problems* in Proceedings of the 7th European Congress on Intelligent Techniques and Soft Computing (EUFIT '99), Aachen, Germany, Springer-Verlag.

Joines J.A. and Houck C.R.(1994), *On the Use of Non-Stationary Penalty Functions to Solve Nonlinear Constrained Optimization Problems with GA's* in Proceedings of the 1st IEEE Conference on Evolutionary Computation. Piscataway, NJ: IEEE Press, pp. 579-584.

Kazarlis S.A., Bakirtzis A.G., and Petridis V.(1996), *A Genetic Algorithm Solution to the Unit Commitment Problem* IEEE Transactions on Power Systems, vol. 11, no. 1, pp. 83-92.

Kazarlis S.A. and Petridis V.(1998), *Varying Fitness Functions in Genetic Algorithms: Studying the Rate of Increase of the Dynamic Penalty Terms* in Proceedings of the 5th International Conference on Parallel Problem Solving from Nature (PPSN-V), Amsterdam, pp. 211-220.

Koziel S. and Michalewicz Z.(1999), *Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization* IEEE Transactions on Evolutionary Computation, vol. 7, No 1, pp 19-44

Michalewicz Z.(1992), *Genetic Algorithms + Data Structures = Evolution Programs,* Second Edition, Springer-Verlag.

Michalewicz Z. and Attia N.(1994), *Evolutionary Optimization of Constrained Problems* in Proceedings of the 3rd Annual Conference on Evolutionary Programming, A.V. Sebald and L.J. Fogel, Eds., River Edge, NJ: World Scientific, pp. 98-108.

Michalewicz Z. and Nazhiyath G.(1995), *Genocop III: A Co-evolutionary Algorithm for Numerical Optimization Problems with Nonlinear Constraints* in Proceedings of the 2nd IEEE Conference on Evolutionary Computation, Perth, Australia (vol. 2). Piscataway, NJ: IEEE Press, pp. 647-651.

Michalewicz Z. and Schoenauer M.(1996), *Evolutionary Algorithms for Constrained Parameter Optimization Problems* IEEE Transactions on Evolutionary Computation, vol. 4, no. 1, pp. 1-32.

Orvosh D. and Davis L.(1994), *Using a Genetic Algorithm to Optimize Problems with Feasibility Constraints* in Proceedings of the 1st IEEE Conference on Evolutionary Computation. Piscataway, NJ: IEEE Press, pp. 548-553.

Petridis V. and Kazarlis S.(1994), *Varying Quality Function in Genetic Algorithms and the Cutting Problem* in Proceedings of the 1st IEEE Conference on Evolutionary Computation (vol. 1). Piscataway, NJ: IEEE Press, pp. 166-169.

Petridis V., Kazarlis S., and Bakirtzis A.(1998), *Varying Fitness Functions in Genetic Algorithm Constrained Optimization: The Cutting Stock and Unit Commitment Problems* IEEE Transactions on Systems, Man, and Cybernetics, vol. 28, part B, no. 5, pp. 629-640.

Powell D. and Skolnick M.(1993), *Using Genetic Algorithms in Engineering Design Optimization with Non-Linear Constraints* in Proceedings of the 5th International Conference on Genetic Algorithms, S. Forrest, Ed. Los Altos, CA: Morgan Kaufmann, pp. 424-430.

Reynolds R.G. (1994), *An Introduction to Cultural Algorithms* in Proceedings of the 3rd Annual Conference on Evolutionary Programming, A. V. Sebald and L. J. Fogel, Eds. River Edge, NJ: World Scientific, pp. 131-139.

Schaffer D.(1985), *Multiple objective optimization with vector evaluated genetic algorithms* in Proceedings of the 1st International Conference on Genetic Algorithms, J.J. Grefenstette (Ed.), Laurence Erlbaum Associates, Hillsdale, NJ, pp. 93-100.

Schoenauer M. and Xanthakis S.(1993), *Constrained GA Optimization* in Proceedings of the 5th International Conference on Genetic Algorithms, S. Forrest, Ed. Los Altos, CA: Morgan Kaufmann, pp. 573-580.

Surry P.D., Radcliffe N.J. and Boyd I.(1995), *A multi-objective approach to constrained optimization of gas supply networks* in Proceedings of the AISB-95 Workshop on Evolutionary Computing, Terence C. Fogarty, Ed., Vol. 993, Springer Verlag, pp. 166-180.