Genetic Algorithms, Numerical Optimization, and Constraints

Zbigniew Michalewicz*
Department of Computer Science
University of North Carolina
Charlotte, NC 28223

Abstract

During the last two years several methods have been proposed for handling constraints by genetic algorithms for numerical optimization problems. In this paper we review these methods, test them on five selected problems, and discuss their strengths and weaknesses. We provide also some suggestions for further research.

1 INTRODUCTION

It is a common knowledge that a successful implementation of an evolutionary technique for a particular real-world problem requires some additional heuristics. These heuristic rules may apply to genetic representation of solutions, to 'genetic' operators that alter their composition, to values of various parameters, to methods for creating initial populations. Also, the process of evaluating an individual in a population may be quite complex (especially in the presence of feasible and unfeasible solutions of a constrained problem) and may lead to different constraint-handling methods. Recently, eleven approaches (including the use of specialized operators, repair algorithms, penalty functions, cultural algorithms, etc.) were surveyed [10] in the context of mathematical programming problems.

Evolutionary computation techniques have received a lot of attention regarding their potential as optimization techniques for complex numerical functions. Many difficult functions were examined; very often they served as test-beds for selection methods, various operators, different representations, etc. However, evolutionary computation methods did not make a significant breakthrough in the area of nonlinear programming due to the fact that they did not address the issue of constraints in a systematic way. Evolution strategies [1] or evolutionary programming techniques (modified to handle numerical optimization problems [5]) just reject unfeasible individuals. Genetic algorithms, on the other hand, penalize unfeasible individuals, however, there are no guidelines on designing penalty functions. A few hypotheses were formulated in [15], but they were not appropriate for generalizations in continuous domains or/and required a huge computational overhead. This paper addresses the issue of constrained numerical optimization: it surveys a few methods proposed recently and examines their merits.

1.1 THE PROBLEM

The general nonlinear programming problem[1] is to find $\overline{X}$ so as to

$$\text{optimize } f(\overline{X}),\ \overline{X} = (x_1, \ldots, x_n) \in R^n,$$

where $\overline{X} \in \mathcal{S} \cap \mathcal{F}$. The set $\mathcal{S} \subseteq R^n$ defines the search space and the set $\mathcal{F} \subseteq R^n$ defines a *feasible* search space. Usually, the search space $\mathcal{S}$ is defined as a n-dimensional rectangle in R^n (domains of variables defined by their lower and upper bounds):

$$l(i) \leq x_i \leq u(i), \quad 1 \leq i \leq n,$$

whereas the feasible set $\mathcal{F}$ is defined by a set of additional $m \geq 0$ constraints:[2]

*Institute of Computer Science, Polish Academy of Sciences, ul. Ordona 21, 01-237 Warsaw, Poland.

[1] We consider here only continuous variables.

[2] The above definition of the search space $\mathcal{S}$ as an n-dimensional rectangle applies only for the case of genetic algorithms. For other methods, such as evolutionary programming and evolution strategies, there is no distinction between the search space and feasible search space, since only feasible solutions are considered.

$g_j(\overline{X}) \leq 0$, for $j = 1, \ldots, q$, and
$h_j(\overline{X}) = 0$, for $j = q+1, \ldots, m$.

Most research on applications of evolutionary computation techniques to nonlinear programming problems was concerned with complex objective functions with $\mathcal{S} \cap \mathcal{F} = \mathcal{S}$. Several test functions used by various researchers during the last 20 years considered only domains of n variables; this was the case with five test functions F1–F5 proposed by De Jong [2], as well as with many other test cases proposed since then [3, 5, 19]. Only recently several approaches were reported to handle general nonlinear programming problems. However, their description is supported by experimental evidence based on different test cases; some of them provide results for test cases with very few variables [11, 16, 7]; some of them did not use functions which have a closed form [14]. Also, they differ in details (representation, selection method, operators and their frequencies, etc.), so it is quite hard to make any comparisons.

This paper surveys these methods (next Section) and provides five test cases (Section 3) which are used here to test the methods listed, and can be used in the future for testing additional methods. These test cases were selected carefully using several criteria. Section 4 presents experimental results and concludes the paper.

2 CONSTRAINT-HANDLING METHODS

During the last two years several methods were proposed for handling constraints by genetic algorithms for numerical optimization problems. Most of them are based on the concept of penalty functions, which penalize unfeasible solutions, i.e.,[3]

$$eval(\overline{X}) = \begin{cases} f(\overline{X}), & if\ \overline{X} \in \mathcal{F} \cap \mathcal{S} \\ f(\overline{X}) + penalty(\overline{X}), & otherwise, \end{cases}$$

where $penalty(\overline{X})$ is zero, if no violation occurs, and is positive, otherwise. In most methods a set of functions f_j $(1 \leq j \leq m)$ is used to construct the penalty; the function f_j measures the violation of the j-th constraint in the following way:

$$f_j(\overline{X}) = \begin{cases} \max\{0, g_j(\overline{X})\}, & if\ 1 \leq j \leq q \\ |h_j(\overline{X})|, & if\ q+1 \leq j \leq m. \end{cases}$$

However, these methods differ in many important details, how the penalty function is designed and applied to unfeasible solutions. In the following subsections we discuss them in turn; the methods are sorted in decreasing order of parameters they require.

[3] In the rest of the paper we assume minimization problems.

2.1 METHOD #1

This method was proposed by Homaifar, Lai, and Qi [7]. The method assumes that for every constraint we establish a family of intervals which determine appropriate penalty coefficient. It works as follows:

- for each constraint, create several (ℓ) levels of violation,
- for each level of violation and for each constraint, create a penalty coefficient R_{ij} $(i = 1, 2, \ldots, \ell,$ $j = 1, 2, \ldots, m)$; higher levels of violation require larger values of this coefficient.
- start with a random population of individuals (feasible or unfeasible),
- evolve the population; evaluate individuals using the following formula

$$eval(\overline{X}) = f(\overline{X}) + \sum_{j=1}^{m} R_{ij} f_j^2(\overline{X}).$$

The weakness of the method is in the number of parameters: for m constraints the method requires: m parameters to establish number of intervals for each constraint (in [7], these parameters are the same for all constraints equal to $\ell = 4$), ℓ parameters for each constraint (i.e., $\ell \times m$ parameters in total); these parameters represent boundaries of intervals (levels of violation), ℓ parameters for each constraint ($\ell \times m$ parameters in total); these parameters represent the penalty coefficients R_{ij}. So the method requires $m(2\ell+1)$ parameters in total to handle m constraints. In particular, for $m = 5$ constraints and $\ell = 4$ levels of violation, we need to set 45 parameters! Clearly, the results are parameter dependent. It is quite likely that for a given problem there exist the optimal set of parameters for which the system returns feasible near-optimum solution, however, it might be quite hard to find it.

2.2 METHOD #2

The second method was proposed by Joines and Houck [8]. As opposed to the previous method, the authors assumed dynamic penalties. Individuals are evaluated (at the iteration t) by the following formula:

$$eval(\overline{X}) = f(\overline{X}) + (C \times t)^{\alpha} \sum_{j=1}^{m} f_j^{\beta}(\overline{X}),$$

where C, α and β are constants. A reasonable choice for these parameters (reported in [8]) is $C = 0.5$,

$\alpha = \beta = 2$. The method requires much smaller number (independent of the number of constraints) of parameters than the first method. Also, instead of defining several levels of violation, the pressure on unfeasible solutions is increased due to the $(C \times t)^\alpha$ component of the penalty term: towards the end of the process (for high values of the generation number t), this component assumes large values.

2.3 METHOD #3

The third method was proposed by Schoenauer and Xanthakis [16]; it works as follows:

- start with a random population of individuals (feasible or unfeasible),
- set $j = 1$ (j is a constraint counter),
- evolve this population with $eval(\overline{X}) = f_j(\overline{X})$, until a given percentage of the population (so-called flip threshold ϕ) is feasible for this constraint,[4]
- set $j = j + 1$,
- the current population is the starting point for the next phase of the evolution, where $eval(\overline{X}) = f_j(\overline{X})$. During this phase, points that do not satisfy one of the 1st, 2nd, ... ,or $(j-1)$-th constraint are eliminated from the population. The stop criterion is again the satisfaction of the j-th constraint by the flip threshold percentage ϕ of the population.
- if $j < m$, repeat the last two steps, otherwise ($j = m$) optimize the objective function, i.e., $eval(\overline{X}) = f(\overline{X})$, rejecting unfeasible individuals.

The method requires a linear order of all constraints which are processed in turn. It is unclear what is the influence of the order of constraints on the results of the algorithm; our experiments indicated that different orders provide different results (different in the sense of the total running time and precision).

In total, the method requires 3 parameters: the sharing factor σ, the flip threshold ϕ, and a particular order of constraints. The method is very different to the previous two methods, and, in general, is different than other penalty approaches, since it considers only one constraint at the time. Also, in the last step of the algorithm the method optimizes the objective function f itself without any penalty component.

[4] The method suggests the use of a sharing scheme (to maintain diversity of the population).

2.4 METHOD #4

The fourth method was described by Michalewicz and Attia [11]; its modified version works as follows:

- divide all constraints into four subsets: linear equations LE, linear inequalities LI, nonlinear equations NE, and nonlinear inequalities NI,
- select a random single point as a starting point (the initial population consists of copies of this single individual). This initial point satisfies linear constraints (LE and LI),
- set the initial temperature $\tau = \tau_0$,
- evolve the population using the following formula:
$$eval(\overline{X}, \tau) = f(\overline{X}) + \frac{1}{2\tau} \sum_{j=1}^{m} f_j^2(\overline{X}),$$
- if $\tau < \tau_f$, stop, otherwise
  - decrease temperature τ,
  - the best solution serves as a starting point of the next iteration,
  - repeat the previous step of the algorithm.

This is the only method described here which distinguishes between linear and nonlinear constraints. The algorithm maintains feasibility of all linear constraints using a set of closed operators, which convert a feasible solution (feasible in terms of linear constraints only) into another feasible solution. At every iteration the algorithm considers active constraints only, the pressure on unfeasible solutions is increased due to the decreasing values of temperature τ.

The method has an additional unique feature: it starts from a single point.[5] Consequently, it is relatively easy to compare this method with other classical optimization methods whose performance are tested (for a given problem) from some starting point.

The method requires a starting and 'freezing' temperatures, τ_0 and τ_f, respectively, and the cooling scheme to decrease temperature τ. Standard values (reported in [11]) are $\tau_0 = 1$, $\tau_{i+1} = 0.1 \cdot \tau_i$, with $\tau_f = 0.000001$.

2.5 METHOD #5

The fifth method was developed by Powell and Skolnick [14]. The method is a classical penalty method with one notable exception. Each individual is evaluated by the formula:

[5] This feature, however, is not essential. The only important requirement is that the next population contains the best individual from the previous population.

$$eval(\overline{X}) = f(\overline{X}) + r\sum_{j=1}^{m} f_j(\overline{X}) + \lambda(t, \overline{X}),$$

where r is a constant; however, there is also a component $\lambda(t, \overline{X})$. This is an additional iteration dependent function which influences the evaluations of unfeasible solutions. The point is that the method distinguishes between feasible and unfeasible individuals by adopting an additional heuristic rule (suggested earlier in [15]): for any feasible individual $\overline{X}$ and any unfeasible individual $\overline{Y}$: $eval(\overline{X}) < eval(\overline{Y})$, i.e., any feasible solution is better than any unfeasible one. This can be achieved in many ways; one possibility is to set

$$\lambda(t, \overline{X}) = \begin{cases} 0, & if\ \overline{X} \in \mathcal{F} \cap \mathcal{S} \\ \max\{0, \max_{\overline{X} \in \mathcal{F} \cap \mathcal{S}}\{f(\overline{X})\} - \\ \quad \min_{\overline{X} \in \mathcal{S} - \mathcal{F}}\{f(\overline{X}) + r\sum_{j=1}^{m} f_j(\overline{X})\}\}, \\ \quad otherwise \end{cases}$$

In other words, unfeasible individuals have increased penalties: their values cannot be better than the value of the worst feasible individual (i.e., $\max_{\overline{X} \in \mathcal{F} \cap \mathcal{S}}\{f(\overline{X})\}$).[6]

2.6 METHOD #6

The final method rejects unfeasible individuals (death penalty); the method has been used by evolution strategies [1], evolutionary programming adopted for numerical optimization [5], and simulated annealing.

3 FIVE TEST CASES

In the selection process of the following five test cases we took into account (1) the type of the objective function, (2) the number of variables, (3) the number of constraints, (4) the types of constraints, (5) the number of active constraints at the optimum, and (6) the ratio ρ between the sizes of the feasible search space and the whole search space $|\mathcal{F} \cap \mathcal{S}|/|\mathcal{S}|$. We do not make any claims on the completeness of the proposed set of these test cases G1–G5, however, it may constitute a handy collection for preliminary tests for other constraint handling methods.

3.1 TEST CASE #1

The problem [4] is to minimize a function:

$$G1(\overline{X}) = 5x_1 + 5x_2 + 5x_3 + 5x_4 - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i,$$

subject to

$$\begin{array}{l} 2x_1 + 2x_2 + x_{10} + x_{11} \leq 10, \\ 2x_1 + 2x_3 + x_{10} + x_{12} \leq 10, \\ 2x_2 + 2x_3 + x_{11} + x_{12} \leq 10, \\ -8x_1 + x_{10} \leq 0, \quad -8x_2 + x_{11} \leq 0, \\ -8x_3 + x_{12} \leq 0, \quad -2x_4 - x_5 + x_{10} \leq 0, \\ -2x_6 - x_7 + x_{11} \leq 0, \quad -2x_8 - x_9 + x_{12} \leq 0, \\ 0 \leq x_i \leq 1,\ i = 1, \ldots, 9, \quad 0 \leq x_i \leq 100, \\ i = 10, 11, 12, \quad 0 \leq x_{13} \leq 1. \end{array}$$

The problem has 9 linear constraints; the function $G1$ is quadratic with its global minimum at

$$\overline{X}^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1),$$

where $G1(\overline{X}^*) = -15$. Six (out of nine) constraints are active at the global optimum (all except the following three: $-8x_1 + x_{10} \leq 0$, $-8x_2 + x_{11} \leq 0$, $-8x_3 + x_{12} \leq 0$).

3.2 TEST CASE #2

The problem [6] is to minimize a function:

$$G2(\overline{X}) = x_1 + x_2 + x_3,$$

where

$$\begin{array}{l} 1 - 0.0025(x_4 + x_6) \geq 0, \\ 1 - 0.0025(x_5 + x_7 - x_4) \geq 0, \\ 1 - 0.01(x_8 - x_5) \geq 0, \\ x_1x_6 - 833.33252x_4 - 100x_1 + 83333.333 \geq 0, \\ x_2x_7 - 1250x_5 - x_2x_4 + 1250x_4 \geq 0, \\ x_3x_8 - 1250000 - x_3x_5 + 2500x_5 \geq 0, \\ 100 \leq x_1 \leq 10000, \quad 1000 \leq x_i \leq 10000, \\ i = 2, 3, \quad 10 \leq x_i \leq 1000, \quad i = 4, \ldots, 8. \end{array}$$

The problem has 3 linear and 3 nonlinear constraints; the function $G2$ is linear and has its global minimum at

$$\begin{array}{l} \overline{X}^* = (579.3167, 1359.943, 5110.071, 182.0174, \\ \qquad 295.5985, 217.9799, 286.4162, 395.5979), \end{array}$$

where $G2(\overline{X}^*) = 7049.330923$. All six constraints are active at the global optimum.

3.3 TEST CASE #3

The problem [6] is to minimize a function:

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

[6] Powell and Skolnick achieved the same result by mapping evaluations of feasible solutions into the interval $(-\infty, 1)$ and unfeasible solutions into the interval $(1, \infty)$. For ranking and tournament selections this implementational difference is not important.

where

$$
\begin{aligned}
&127 - 2x_1^2 - 3x_2^4 - x_3 - 4x_4^2 - 5x_5 \geq 0,\\
&282 - 7x_1 - 3x_2 - 10x_3^2 - x_4 + x_5 \geq 0,\\
&196 - 23x_1 - x_2^2 - 6x_6^2 + 8x_7 \geq 0,\\
&-4x_1^2 - x_2^2 + 3x_1x_2 - 2x_3^2 - 5x_6 + 11x_7 \geq 0\\
&-10.0 \leq x_i \leq 10.0,\ i = 1, \ldots, 7.
\end{aligned}
$$

The problem has 4 nonlinear constraints; the function $G3$ is nonlinear and has its global minimum at

$$
\begin{aligned}
\overline{X}^* = (&2.330499, 1.951372, -0.4775414,\\
&4.365726, -0.6244870, 1.038131, 1.594227),
\end{aligned}
$$

where $G3(\overline{X}^*) = 680.6300573$. Two (out of four) constraints are active at the global optimum (the first and the last one).

3.4 TEST CASE #4

The problem [6] is to minimize a function:

$$G4(\overline{X}) = e^{x_1x_2x_3x_4x_5},$$

subject to

$$
\begin{aligned}
&x_1^2+x_2^2+x_3^2+x_4^2+x_5^2 = 10, \quad x_2x_3-5x_4x_5 = 0,\\
&x_1^3 + x_2^3 = -1, \quad -2.3 \leq x_i \leq 2.3,\ i = 1, 2,\\
&-3.2 \leq x_i \leq 3.2, \quad i = 3, 4, 5.
\end{aligned}
$$

The problem has 3 nonlinear equations; nonlinear function $G4$ has its global minimum at

$$
\begin{aligned}
\overline{X}^* = (&-1.717143, 1.595709, 1.827247,\\
&-0.7636413, -0.7636450),
\end{aligned}
$$

where $G4(\overline{X}^*) = 0.0539498478$.

3.5 TEST CASE #5

The problem [6] is to minimize a function:

$$
\begin{aligned}
G5(\overline{X}) = &x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2\\
&+4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 +\\
&7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45,
\end{aligned}
$$

where

$$
\begin{aligned}
&105 - 4x_1 - 5x_2 + 3x_7 - 9x_8 \geq 0,\\
&-10x_1 + 8x_2 + 17x_7 - 2x_8 \geq 0,\\
&8x_1 - 2x_2 - 5x_9 + 2x_{10} + 12 \geq 0,\\
&-3(x_1-2)^2-4(x_2-3)^2-2x_3^2+7x_4+120 \geq 0,\\
&-5x_1^2 - 8x_2 - (x_3 - 6)^2 + 2x_4 + 40 \geq 0,\\
&-x_1^2 - 2(x_2 - 2)^2 + 2x_1x_2 - 14x_5 + 6x_6 \geq 0,\\
&-0.5(x_1-8)^2-2(x_2-4)^2-3x_5^2+x_6+30 \geq 0,\\
&3x_1 - 6x_2 - 12(x_9 - 8)^2 + 7x_{10} \geq 0,\\
&-10.0 \leq x_i \leq 10.0, \quad i = 1, \ldots, 10.
\end{aligned}
$$

The problem has 3 linear and 5 nonlinear constraints; the function $G5$ is quadratic and has its global minimum at

$$
\begin{aligned}
\overline{X}^* = (&2.171996, 2.363683, 8.773926, 5.095984,\\
&0.9906548, 1.430574, 1.321644, 9.828726,\\
&8.280092, 8.375927),
\end{aligned}
$$

where $G5(\overline{X}^*) = 24.3062091$. Six (out of eight) constraints are active at the global optimum (all except the last two).

3.6 SUMMARY

All test cases are summarized in Table 1; for each test case (TC) we list number n of variables, type of the function f, the ratio $\rho = |\mathcal{F} \cap \mathcal{S}|/|\mathcal{S}|$, the number of constraints of each category (linear inequalities LI, nonlinear equations NE and inequalities NI), and the number a of active constraints at the optimum.

TC	n	Type of f	ρ	LI	NE	NI	a
#1	13	quadratic	0.0111%	9	0	0	6
#2	8	linear	0.0010%	3	0	3	6
#3	7	polynomial	0.5121%	0	0	4	2
#4	5	nonlinear	0.0000%	0	3	0	3
#5	10	quadratic	0.0003%	3	0	5	6

Table 1: Summary of five test cases. The ratio $\rho = |\mathcal{F} \cap \mathcal{S}|/|\mathcal{S}|$ was determined experimentally by generating 1,000,000 random points from $\mathcal{S}$ and checking whether they belong to $\mathcal{F}$. LI, NE, and NI represent the number of linear inequalities, and nonlinear equations and inequalities, respectively.

4 EXPERIMENTS, RESULTS, AND CONCLUSIONS

In all experiments we assumed floating point representation, nonlinear ranking selection, Gaussian mutation, arithmetical and heuristic crossovers; the probabilities of all operators were set at 0.08, and the population size was 70. For all methods the system was run for 5,000 generations.[7]

[7] Modified versions of Genocop system [9] were used for all experiments; the code was updated accordingly for each method. Genocop is available from anonymous, ftp.uncc.edu, directory coe/evol, file genocop3.0.tar.Z.

TC	Exact opt.		Method #1	Method #2	Method #3
#1	−15.000	b	−15.002	−15.000	−15.000
		m	−15.002	−15.000	−15.000
		w	−15.001	−14.999	−14.998
		c	0, 0, 4	0, 0, 0	0, 0, 0
#2	7049.331	b	2282.723	3117.242	7485.667
		m	2449.798	4213.497	8271.292
		w	2756.679	6056.211	8752.412
		c	0, 3, 0	0, 3, 0	0, 0, 0
#3	680.630	b	680.771	680.787	680.836
		m	681.262	681.111	681.175
		w	689.660	682.798	685.640
		c	0, 0, 1	0, 0, 0	0, 0, 0
#4	0.054	b	0.084	0.059	
		m	0.955	0.812	*
		w	1.000	2.542	
		c	0, 0, 0	0, 0, 0	
#5	24.306	b	24.690	25.486	
		m	29.258	26.905	—
		w	36.060	42.358	
		c	0, 1, 1	0, 0, 0	

Table 2: Experimental results. For each method (#1, #2, and #3) we report the best (b), median (m), and the worst (w) result (out of 10 independent runs) and the number (c) of violated constraints at the median solution: the sequence of three numbers indicate the number of violations by more than 1.0, more than 0.1, and more than 0.001, respectively. The symbols '$*$' and '—' stand for 'the method was not applied to this test case' and 'the results were not meaningful', respectively.

Method #4	Method #5	Method #6	Method #6(f)
−15.000	−15.000		−15.000
−15.000	−15.000	—	−14.999
−15.000	−14.999		−13.616
0, 0, 0	0, 0, 0		0, 0, 0
7377.976	2101.367		7872.948
8206.151	2101.411	—	8559.423
9652.901	2101.551		8668.648
0, 0, 0	1, 2, 0		0, 0, 0
680.642	680.805	680.934	680.847
680.718	682.682	681.771	681.826
680.955	685.738	689.442	689.417
0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
0.054	0.067		
0.064	0.091	*	*
0.557	0.512		
0, 0, 0	0, 0, 0		
18.917	17.388		25.653
24.418	22.932	—	27.116
44.302	48.866		32.477
0, 1, 0	1, 0, 0		0, 0, 0

Table 3: Continuation of the Table 2; experimental results for methods #4, #5, and #6. The results for method #6 report also the results of experiments (method #6(f)) where the initial population was feasible.

The results are summarized in Tables 2 and 3, which report (for each method) the best (row b), median (row m), and the worst (row w) result (out of 10 independent runs) and numbers (row c) of violated constraints at the median solution: the sequence of three numbers indicate the number of violations with violation amount between 1.0 and 10.0, between 0.1 and 1.0, and between 0.001 and 0.1, respectively (a sequence of three zeros indicates a feasible solution). If at least one constraint was violated by more than 10.0 (in terms of functions f_j), the solution was considered as 'not meaningful'. In some cases it was hard to determine "the best" solution due to a relationship between the objective value and the number of violated constraints; the tables report the smallest objective value (for the best solution); consequently, some values are "better" than the value at the global minimum.

It is difficult to provide a complete analysis of all six methods on the basis of five test cases,[8] however, it seems that method #1 can provide good results only if violation levels and penalty coefficients R_{ij} are tuned to the problem (e.g., our arbitrary choice of 4 violation levels with penalties of 100, 200, 500, and 1,000, respectively, worked well for the test cases #1 and #3, whereas it did not work well for other test cases, where some other values for these parameters are required). Also, these violation levels and penalty coefficients did not prevent the system from violating 3 constraints in the test case #2, since the 'reward' was too large to resist (i.e., the optimum value *outside* the feasible region is 2,100 for $\overline{X} = (100.00, 1000.00, 1000.00, 128.33, 447.95, 336.07, 527.85, 578.08)$ with relatively small penalties). In all test cases (except test case #4) the method returned solutions which were unfeasible by a relatively small margin, which is an interesting characteristic of this method.

Method #2 provided better results than the previous method for almost all test cases: for test case #1

[8] Future work would extend the test suite to 25 test cases.

(where all returned solutions were feasible), test cases #2 and #4 (where constraint violations were much smaller), and test case #3 (the standard deviation of results was much smaller). On the other hand, method #2 seems to provide too strong penalties: often the factor $(C \times t)^{\alpha}$ grows too fast to be useful. The system has little chances to escape from local optima: in most experiments the best individual was found in early generations. It is also worth mentioning that this method gave very good results for test cases #1 and #5, where the objective functions were quadratic.

Method #3 was not applied to test case #4, and it did not give meaningful results for test case #5, which has a very small ratio ρ (the smallest apart from the test case #4). Clearly, the method is quite sensitive to the size of the feasible part of the search space. Also, some additional experiments indicated that the order of constraints to be considered influenced the results in a significant way. On the other hand, the method performed very well for test cases #1 and #3, and for test case #2 it gave reasonable results.

Method #4 performed very well for the test cases #1, #3 and #4, where it provided the best results. It also gave a reasonable performance in test case #2 (where linear constraints were responsible for the failure of the methods #1 and #2). However, for test case #5 the method gave quite poor results in comparison with methods #1, #2, and #6(f); it seems that linear constraints of this problem prevented the system from moving closer to the optimum. This is an interesting example of the damaging effect of limiting the population to the feasible (with respect to linear constraints) region only. Additional experiments indicated that the method is very sensitive to the cooling scheme. For example, the results of the test case #5 were improved in a significant way for different cooling scheme ($\tau_{i+1} = 0.01 \cdot \tau_i$).

Method #5 had difficulties in locating a feasible solution for test case #2: similarly to methods #1 and #2 the algorithm settled for unfeasible solution. In all other test cases the method gave a stable, reasonable performance, returning feasible solutions (test cases #1, #3, and #5), or slightly unfeasible solutions (test case #4). Additional experiments (not reported in the tables) included runs of the method #5 with a feasible initial population. For test case #2, the results were almost identical to these of method #6(f)). However, for test case #5, the results were excellent (the best of all methods).

The method #6 (apart from test case #3) did not produce any meaningful results. To test this method properly it was necessary to initialize a population by feasible solutions (method #6(f)). This different initialization scheme makes the comparison of these methods even harder. However, an interesting pattern emerged: the method generally gave a quite poor performance. The method was not as stable as other methods on the easiest test case #1 (this was the only method to return a solution of -13.616, far from the optimum), and for the test case #2 only in one run the returned value was below 8000.

No single parameter (number of linear, nonlinear, active constraints, the ratio *rho*, type of the function, number of variables) proved their significance as a measure of difficulty of the problem for the evolutionary techniques. For example, all methods approached the optimum quite closely for the test cases #1 and #5 (with $\rho = 0.0111\%$ and $\rho = 0.0003\%$, respectively), whereas most of the methods experienced difficulties for the test case #2 (with $\rho = 0.0010\%$). Two quadratic functions (test cases #1 and #5) with a similar number of constraints (9 and 8, respectively) and an identical number (6) of active constraints at the optimum, gave a different challenge to most of these methods. It seems that other properties (the characteristic of the objective function together with the topology of the feasible region) constitute quite significant measures of the difficulty of the problem. Also, several methods were quite sensitive to the presence of a feasible solution in the initial population.

All of the methods presented here (except method #3) penalized unfeasible individuals. The penalty functions were based on the amount of violation (functions f_j). It might be worthwhile to experiment also with penalties (as suggested in [15]) which incorporate additional information about the objective function: the estimation of its trend and maximum derivative. Additionally, other constraint handling methods deserve some attention. One of them (currently being implemented) is based on repair algorithms: an unfeasible solution $\overline{X}$ is "forced" into the feasible region and its repaired version is evaluated. The question, whether the repaired version should replace the original vector in the population (and, if yes, with what probability) remains open.[9] An additional possibility would include the use of the values of objective function f and penalties f_j as elements of a vector and applying multi-objective techniques to minimize all components of the vector [18] (however, analysis of Schaffer's VEGA system indicated the equivalence between

[9] For discrete cases, Orvosh and Davis reported so-called 5% rule [12], which states that in many combinatorial optimization problems, an evolutionary computation technique with a repair algorithm provides the best results while 5% of repaired individuals replace their unfeasible originals.

multiobjective optimization and linear combination of f and f_j's [15]). Also, an interesting approach was recently reported by Paredis [13]. The method (described in the context of constraint satisfaction problems) is based on a co-evolutionary model, where a population of potential solutions co-evolves with a population of constraints: fitter solutions satisfy more constraints, whereas fitter constraints are violated by more solutions. It would be interesting to adopt this method for numerical optimization problems and compare this approach with the penalty methods.

Acknowledgments:

This material is based upon work supported by the National Science Foundation under Grant IRI-9322400.

References

[1] Bäck, T., Hoffmeister, F., and Schwefel, H.-P., *A Survey of Evolution Strategies*, Proceedings of the Fourth ICGA, Morgan Kaufmann Publishers, Los Altos, CA, 1991, pp.2–9.

[2] De Jong, K.A., *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, (Doctoral dissertation, University of Michigan), *Dissertation Abstract International*, 36(10), 5140B. (University Microfilms No 76-9381).

[3] Eshelman, L.J. and Schaffer, J.D, *Real-Coded Genetic Algorithms and Interval Schemata*, Foundations of Genetic Algorithms – 2, Morgan Kaufmann, Los Altos, CA, 1993, pp. 187–202.

[4] Floudas, C.A. and Pardalos, P.M., *A Collection of Test Problems for Constrained Global Optimization Algorithms*, Springer-Verlag, Lecture Notes in Computer Science, Vol.455, 1987.

[5] Fogel, D.B. and Stayton, L.C., *On the Effectiveness of Crossover in Simulated Evolutionary Optimization*, BioSystems, Vol.32, 1994, pp.171–182.

[6] Hock, W. and Schittkowski K., *Test Examples for Nonlinear Programming Codes*, Springer-Verlag, Lecture Notes in Economics and Mathematical Systems, Vol.187, 1981.

[7] Homaifar, A., Lai, S. H.-Y., Qi, X., *Constrained Optimization via Genetic Algorithms*, Simulation, Vol.62, No.4, 1994, pp.242–254.

[8] Joines, J.A. and Houck, C.R., *On the Use of Non-Stationary Penalty Functions to Solve Nonlinear Constrained Optimization Problems With GAs*, Proceedings of the IEEE ICEC 1994, pp.579–584.

[9] Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, New York, 2nd edition, 1994.

[10] Michalewicz, Z., *A Survey of Constraint Handling Techniques in Evolutionary Computation Methods*, Proceedings of the 4th Annual Conference on Evolutionary Programming, MIT Press, Cambridge, MA, 1995.

[11] Michalewicz, Z., and Attia, N., *Evolutionary Optimization of Constrained Problems*, Proceedings of the 3rd Annual Conference on Evolutionary Programming, World Scientific, 1994, pp.98–108.

[12] Orvosh, D. and Davis, L., *Shall We Repair? Genetic Algorithms, Combinatorial Optimization, and Feasibility Constraints*, Proceedings of the Fifth ICGA, Morgan Kaufmann, 1993, p.650.

[13] Paredis, J., *Co-evolutionary Constraint Satisfaction*, Proceedings of the 3rd PPSN Conference, Springer-Verlag, 1994, pp.46–55.

[14] Powell, D. and Skolnick, M.M., *Using Genetic Algorithms in Engineering Design Optimization with Non-linear Constraints*, Proceedings of the Fifth ICGA, Morgan Kaufmann, 1993, pp.424–430.

[15] Richardson, J.T., Palmer, M.R., Liepins, G., and Hilliard, M., *Some Guidelines for Genetic Algorithms with Penalty Functions*, in Proceedings of the Third ICGA, Morgan Kaufmann, 1989, pp.191–197.

[16] Schoenauer, M., and Xanthakis, S., *Constrained GA Optimization*, Proceedings of the Fifth ICGA, Morgan Kaufmann, 1993, pp.573–580.

[17] Schwefel, H.-P., *Numerical Optimization for Computer Models*, Wiley, Chichester, UK, 1981.

[18] Surry, P.D., Radcliffe, N.J., and Boyd, I.D., *A Multi-objective Approach to Constrained Optimization of Gas Supply Networks*, presented at the AISB-95 Workshop on Evolutionary Computing, Sheffield, UK, April 3-4, 1995.

[19] Wright, A.H., *Genetic Algorithms for Real Parameter Optimization*, First Workshop on the Foundations of Genetic Algorithms and Classifier Systems, Morgan Kaufmann Publishers, Los Altos, CA, 1991, pp. 205–218.