An Adaptive Algorithm for constrained optimization problems

S. Ben Hamida and **Marc Schoenauer**

CMAP, URA CNRS 756,
Ecole Polytechnique,
Palaiseau 91128, France;
sana@cmapx.polytechnique.fr
Marc.Schoenauer@polytechnique.fr

Abstract. Adaptivity has become a key issue in Evolutionary Algorithms, since early works in Evolution Strategies. The idea of letting the algorithm adjust its own parameters for free is indeed appealing. This paper proposes to use adaptive mechanisms at the population level for constrained optimization problems in three important steps of the evolutionary algorithm: First, an adaptive penalty function takes care of the penalty coefficients according to the proportion of feasible individuals in the current population; Second, a Seduction/Selection strategy is used to mate feasible individuals with infeasible ones and thus explore the region around the boundary of the feasible domain; Last, selection is tuned to favor a given number of feasible individuals. A detailed discussion of the behavior of the algorithm on two small constrained problems enlights adaptivity at work. Finally, experimental results on eleven test cases from the literature demonstrate the power of this approach.

1 Introduction

The general nonlinear parameter optimization problem is defined as:

$$\text{optimize } f(\boldsymbol{x}), \boldsymbol{x} = (x_1, \ldots, x_n) \in \mathcal{F} \subseteq \mathcal{S} \subseteq \boldsymbol{R}^n,$$

such that:

$$\begin{cases} g_i(\boldsymbol{x}) \leq 0, \text{ for } i = 1, \ldots, q & \text{inequality constraints} \\ h_j(\boldsymbol{x}) = 0, \text{ for } j = q+1, \ldots, m & \text{equality constraints} \end{cases}$$

where f, g_i and h_j are real-valued functions on the search space $\mathcal{S}$. The satisfaction of the set of constraints (g_i, h_j) defines the feasible region $\mathcal{F}$.

Though many specific methods to handle such constrained problems within Evolutionary Algorithms have been proposed (see e.g. [10] for a survey), this paper will concentrate on the penalty function method.

The idea of penalty methods is to penalize infeasible solutions by adding to the objective function f a positive quantity (when the goal is to minimize f) in order to decrease the quality of such infeasible individuals:

$$eval(\boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}), & \text{if } \boldsymbol{x} \in \mathcal{F} \\ f(\boldsymbol{x}) + penal(\boldsymbol{x}), & \text{otherwise} \end{cases}$$

where $penal(\boldsymbol{x})$ is positive for minimizing and negative for maximizing.

The design of the penalty function $penal$ is the main source of difficulty of penalty methods. The most popular approaches use measures of the constraint violations, such as:

$$penal(\boldsymbol{x}) = \sum_{j=1}^{q} \alpha_j g_j^+(\boldsymbol{x}) + \sum_{j=q+1}^{m} \alpha_j |h_j|(\boldsymbol{x}), \tag{1}$$

where x^+ denotes the positive part of x. The real numbers $\alpha_j, j = 1 \dots m$ are called the *penalty coefficients*, and the main difficulty is to determine appropriate values for each one of them.

This paper presents ASCHEA, the Adaptive Segregational Constraint Handling Evolutionary Algorithm, based on a population level adaptive way [3] to adjust the values of the penalty coefficients, an adaptive selection/seduction mechanism for mate choice and a specific feasibility-oriented selection operator. The idea is to maintain in the population both feasible and infeasible individuals: the search can either concentrate around the boundary of the feasible region, as it is well known that the optimum lies on that boundary in many real problems, or normally explore the feasible region otherwise. This is achieved by increasing the penalty coefficients if there are not enough feasible individuals in the population, keeping the feasible individuals in the population using segregated selection, and mating feasible and infeasible individuals together to explore the surroundings of the boundary.

The paper is organized as follows. Section 2 briefly surveys previous constraint handling methods based on penalty functions. Section 3 introduces the components of the algorithm and a priori discusses their advantages. Section 4 is devoted to a detailed study of the behavior of the algorithm on two selected test cases. Finally, an extensive experimental study is presented in section 5, giving some results obtained by this technique on some reference test problems taken from [10, 6].

2 Evolutionary penalty methods

First note that the general penalty approach to handle constraints is not specific to Evolutionary Computation, as any optimization method can be applied to the penalized objective function. For instance, the static methods reviewed here are very general. The dynamic methods could probably be adapted to any iterative optimization algorithm. But the other methods are very specific to Evolutionary Computation. Anyway, all methods will be presented in the framework of Evolutionary Computation.

2.1 Static methods

The penalty function is fixed once and for all at the beginning of the algorithm.

The "death penalty" method simply rejects infeasible solutions from the population. Though not exactly a penalty method (the rejection takes place in the selection step), it can be viewed as a penalty method with infinite penalty

[1]. This method might have great difficulties to find even a first feasible point.

The **static penalty** methods use user-defined values for the penalty coefficients. Some improvement was brought in [4] by increasing these values with the level of violation – but the main difficulty results from the lack of any hint about the value these penalty coefficient should have.

2.2 Dynamic methods

In these methods, the penalty coefficients are modified along evolution according to a user-defined schedule – usually increased in order to ensure feasible individuals in the end. These methods suffer the same defect than static penalty, i.e. it is not easy to tune the values of their parameters.

In the **dynamic penalty** method [5], the penalty coefficients increase like $C \times t^{\beta}$, for user-defined C and β (t is the current generation).

Genocop II [8] uses the method of **annealing penalties** : the penalty coefficients are computed by $\frac{1}{2\tau(t)}$ with a "freezing temperature" (τ) which decreases every generation. Note that Genocop II also uses for linear constraints a set of special operators maintaining feasibility of solutions [9].

2.3 Adaptive methods

In these methods, information is gleaned from the population to update the value of the penalty functions.

A first example of adaptive method is the method based on the **superiority of feasible points** [11]: Some positive term, depending on the current infeasible individuals, is added to the constraint violations in the penalty function, ensuring any feasible individual will be better than any infeasible individual.

Different **adaptive penalty** methods have been proposed. In [2], the penalty coefficient is increased (resp. decreased) if the best individual is infeasible (resp. feasible) during a given number of generations. In [15], the penalty is proportional to the difference between the best feasible objective function ever seen and the best overall objective value ever seen - the first goal is to find at least one feasible solution. However, both method rely only on the best individuals in the past populations to adjust the penalty, and this clearly might not reflect precisely what is going on at the population level.

2.4 Yet another penalty method

The segregated GA [7] tries to make a balance between heavy and moderate penalties. It uses a specific selection strategy whose effect is to somehow maintain two cooperating subpopulations: A first subpopulation is evaluated using large values of penalty parameters, thus containing mostly feasible individuals; A second subpopulation uses small values for penalty coefficients, and hence is likely to contain mostly infeasible individuals.

3 The adaptive segregational algorithm

This section introduces the original Adaptive Segregational Constraint Handling Evolutionary Algorithm (**ASCHEA**). The main idea in ASCHEA is to maintain both feasible and infeasible individuals in the population, at least when it seems necessary. In order to achieve this goal, three main ingredients will be used: An original adaptive penalty method that uses global information of the population to adjust the penalty coefficients; a constraint-driven recombination, where in some cases feasible individuals can only mate with infeasible individuals; a segregational selection that distinguishes between feasible and infeasible individuals.

In the rest of the paper, τ_t will denote the proportion of feasible individuals in the population at generation t, and τ_{target} will be a user-defined proportion. The idea of the proposed method is to maintain τ_t as close as possible from τ_{target} along evolution.

3.1 Population-based adaptive penalty

The penalty function used here is that of equation (1). For the sake of simplicity, only the case of a single coefficient will be considered – whether there is a single constraint, or all penalty coefficients are set to the same value $\alpha(t)$ at generation t.

Increasing the value of the penalty coefficient α in equation (1) clearly favors feasible individuals in subsequent selections, while decreasing it favors infeasible individuals. Hence, in order to try to maintain a given proportion τ_{target} of feasible individuals (and hence $1-\tau_{target}$ of infeasible individuals!) the following strategy is used for coefficient α:

$$\begin{array}{ll} \text{if } (\tau_t > \tau_{target}) & \alpha(t+1)(\boldsymbol{x}) = \alpha(t)/fact \\ \text{otherwise} & \alpha(t+1)(\boldsymbol{x}) = \alpha(t) * fact \end{array}$$

where $fact > 1$ is a user-defined parameter.

The initial penalisation $\alpha(0)$ is computed using the first population, to try to balance between objective function and constraint violation:

$$\alpha(0) = \left| \frac{\sum_{i=1}^{n} f(\boldsymbol{x_i})}{\sum_{i=1}^{n} v(\boldsymbol{x_i})} \right| * 1000,$$

where $v(\boldsymbol{x_i})$ is the sum of the constraint violations of individual x_i.

3.2 Constraint-driven mate selection

In many real-world problems, it is known that the constrained optimum lies close to the boundary of the feasible domain (e.g. when minimizing some cost with technological constraints). On the other hand, restricting the search to that boundary, though very powerful when applicable [13], is highly problem-dependent. Moreover, it might prove too restrictive, too, as the solution sometimes lies very close to, but outside from, that boundary.

In order to both achieve better exploration of the boundary region and attract infeasible individuals more rapidly toward feasible ones, it is proposed to use a selection/seduction mechanism [12], choosing the mate of feasible individuals to be infeasible. However, to also allow exploration of the feasible region, this mechanism is only applied when too few (with respect to τ_{target}) feasible individuals are present in the population: otherwise, i.e. if $\tau_t > \tau_{target}$, it is assumed that exploration should proceed normally – and the mate is chosen in the whole population. More precisely, to select the mate x_2 for a first parent x_1:

if (0 < $\tau_t < \tau_{target}$) **and** (x_1) is feasible
  select (x_2) among infeasible individuals only
else select (x_2) according to fitness only

Such strategy can of course be used together with any recombination operator.

3.3 Segregational selection

To further enhance the chances of survival of feasible individuals, a specific selection operator is used in the algorithm. This selection, called segregational selection, can be viewed as intermediate between the method based on the superiority of feasible points (see section 2.3 and [11]) and the replacement method used in the Segregated GA [7] (see section 2.4).

The segregational selection is a deterministic replacement mechanism used in an ES-like scheme [14]: from a population of μ parents, λ offspring are generated (all parents giving birth to $\frac{\lambda}{\mu}$ offspring on average). Among those λ offspring (for the "," strategy) or among the μ parents plus the λ offspring (for the "+" strategy), μ individuals are selected to become the new parents the following way:

First, let τ_{select} be a user-defined proportion: the segregational selection starts by selecting without replacement among feasible individuals, based on their fitness, until $\tau_{select} * \mu$ have been selected, or no more feasible individual is available. The population is then filled using standard deterministic selection on the remaining individuals, based on the current penalized fitness.

So only a proportion τ_{select} of feasible individuals is considered superior to all infeasible points. Moreover, the above segregational selection can be viewed as a Segregated GA [7] where the large penalty coefficient would be set to $+\infty$.

3.4 Discussion

A first consequence of the use of the segregated selection is some sort of feasible-elitism: as soon as a feasible individual appears, it can only disappear from the population by being replaced by a better feasible guy, even if the penalty coefficient later reaches very small values and greatly favors infeasible individuals.

This in turn ensures that the constraint-driven mate selection can play its role, which is to accelerate the movement toward the feasible region of infeasible individuals, and to explore in detail the region close to the boundary of the feasible domain.

But all the above mechanisms are activated based on the proportion of feasible in the population. This enables the proposed algorithm to adaptively handle both the case where the optimum lies on the boundary of the feasible region, and the case where the optimum lies inside the feasible region. Indeed, in the first case, the constraint-driven mate selection allows to explore both sides of the boundary. And in the second case, it is likely that the whole population will rapidly tend to enter the feasible region, and standard "blind" mate selection will be applied while the segregational selection will not be different from pure deterministic selection for the feasible fraction of the population.

4 Experimental behavior

This section studies the application of ASCHEA to two test-cases in detail, selected from [6]. The first function (G6) has 2 nonlinear inequality constraints and its optimum lies on the boundary of the feasible region (Fig. 1, left). The second one (G8) has also two nonlinear inequality constraints and 2 quasi-optima inside the feasible region (Fig. 1,right).

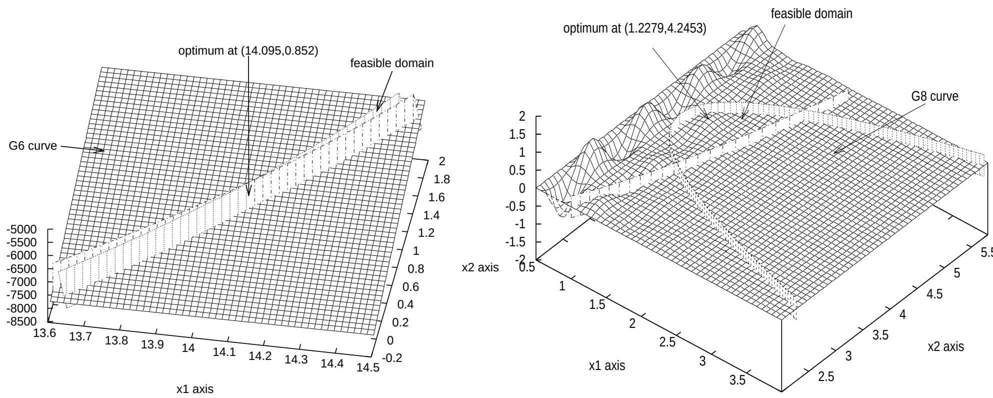

Fig. 1. *G6 (left) and G8 (right) landscapes. The constraint boundaries have been artificially made visible. Note that function G8 has huge peaks toward the x_2 axis that are not shown here.*

4.1 Experimental conditions

ASCHEA uses here a (100+300)-ES segregational selection strategy (section 3.3). Mutation is the standard Gaussian mutation and self-adaptive standard deviations [14], with initial values of 0.03, global learning rate of 0.1 and local learning rate of 1. Recombination is the standard arithmetical crossover – plus of course constraint-driven mate selection when triggered (section 3.2). For all experiments, τ_{select} is set to 0.3, τ_{target} to 0.6 and $fact$ to 1.1, the crossover and mutation rates are set to 0.9. At least 31 independent runs of 5000 generations are performed for all tests.

4.2 The solution lies on the boundary

For each trial, the optimum of G6 is found pretty quickly (Fig. 2 and 3). In the first generations, as soon as feasible individuals are found, the segregational selection keeps them in the population. The constraint-driven mate selection is then applied, which rapidly increases the proportion of feasible points in the population and aggregates some individuals close to the boundary of the feasible region. Fig. 2 shows how the algorithm helps the individuals to enter rapidly (5 generations) into the feasible region.

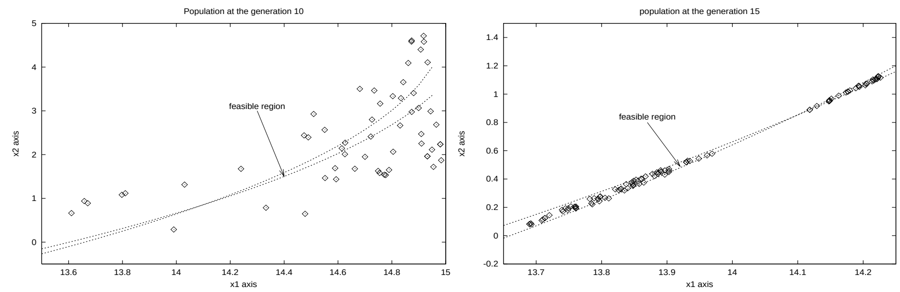

Fig. 2. *Population of G6 at the generation 10 and 15*

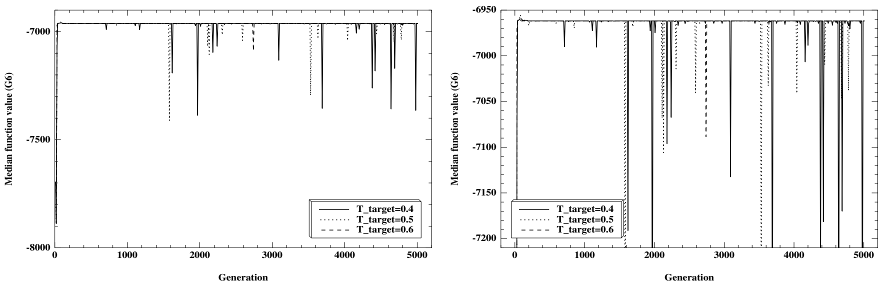

Fig. 3. *Curve of the median (out of 31 runs) of the best objective function values (***NOT fitness***!) vs the number of generations for G6 with different values of* τ_{target}. *The second plot is a zoom near the optimum. The effect of adaptive penalty are clearly visible in the downward peaks – corresponding to infeasible best individuals.*

The deterministic ES-like selection increases the exploitation capability of the algorithm. But, as can be seen on Fig. 3, exploration continues around the boundary, and even in the infeasible region, thanks to the adaptive penalisation strategy. The peaks in the figure demonstrate that the optimum of the fitness can change (low objective values correspond to infeasible individuals). However, the best feasible individual is never lost.

4.3 The solution lies inside the feasible domain

As for the first test, 31 independent runs are performed on function G8. As can be seen on Figure 4-left, whereas all individuals of generation 3 are still widely spread over both feasible and infeasible domains, the whole population has become feasible at generation 5. Later, at generation 10, the population is already concentrated around the optimum. The evolution then proceeds normally with standard deterministic selection, while the penalty coefficient decreases rapidly. Hence, when a first mutant hits an outside peak (the one that can be seen on Fig. 1), it quickly attracts part of the population (Fig. 4-right). However, thanks to the segregated selection, a fraction τ_{select} of the population remains feasible, and the whole population has converged in less than 30 generations (not shown). Note however that if evolution proceeds, this phenomenon will happen again.

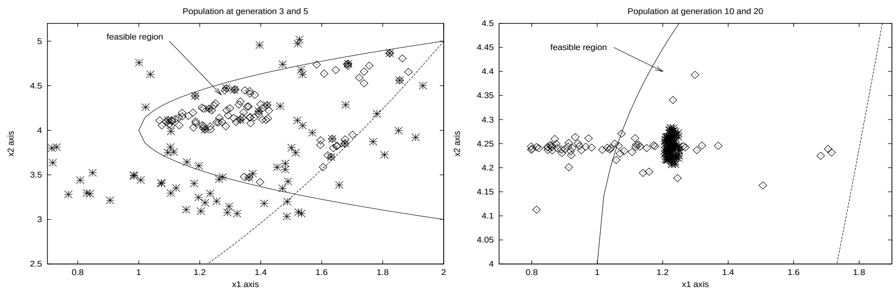

Fig. 4. *Evolution of population during G8 optimization: at generation 3 and 5 (resp. stars and squares on left) and at generation 10 and 20 (resp. stars and squares on right). While the population had started to converge at generation 10, a lucky mutation toward the closest peak outside the feasible domain has drawn many individuals toward infeasibility. Thanks to the segregated selection and the mate selection, the population will be back around the optimum rapidly.*

5 Extensive experimental results

In this section, in order to demonstrate the robustness of our approach, experiments are made for the eleven benchmark functions proposed in [10] and [6] (from G1 to G11). For each one, 31 independent runs were performed using the parameters described in section 4.1 – except for function G10 where meaningful results could only be found using a mutation rate of 0.3.

The results of all experiments are summarized in Table 1. All runs could find a feasible solutions. The exact optimal solutions were found for 7 problems out of 11.

The first conclusion of these results is that ASCHEA gave satisfactory results for all test cases, except for the test case G5, where the replacement of both equality constraints by inequalities did not prove efficient. The problem seems to come from the use of a single penalty coefficient. Further work will use one penalty coefficient for each constraint.

Moreover, it is possible to further enhance the performances of the algorithm by a more precise tuning of its parameters. For example, for test case G3*, the number of successful runs increases when the algorithm is run for longer time, as expected! Table 2 shows the influence of the number of generations on the quality of the results.

Another parametric study should (and will) be devoted to parameter τ_{target}: it can have a great influence on the quality of the results, as for instance for test case G2: higher values of τ_{target} and $fact$ gave better results than those of Table 1; With $\tau_{target} = 0.7$ and $fact = 1.2$, the algorithm gave as best result 0.803603 with average 0.606.

In the opposite, on test case G4, ASCHEA gave better results with lower values of τ_{target}. But this is probably due to the fact that the optimum lies on the boundary of the feasible region, a lower τ_{target} value emphasize exploration of the feasible domain boundary, as confirmed by the results of Table 3.

Function	Opt. value	Opt. found			After 5000 gen		
		Hits	best (nb gen)	Avg. (nb gen)	best	median	Avg.
G1	-15	24	249	261	**-15**	-15	-14.6819
G2	0.803553	–	–	–	0.800781	0.506	0.5462
G3*	1.0	1	4205	4205	**1**	0.999979	0.999844
G4	-30665.5	7	109	159	**-30665.5**	-30658.9	-30650.745
G5*	5126.4981	–	–	–	4707.52	4283.41	4323.73
G6	-6961.81	31	42	85	**-6961.81**	-6961.81	-6961.81
G7	24.3062	–	–	–	24.36	24.505	24.6109
G8	0.095825	31	12	32	**0.095825**	0.095825	0.095825
G9	680.630	1	4267	4267	**680.630**	680.637	680.648
G10	7049.33	–	–	–	7095.15	7840.67	8858.64
G11*	0.75	31	27	61	**0.75**	0.75	0.75

Table 1. Summary of results on 11 test cases, giving the best, the median and the average best (feasible) objective value out of 31 runs after 5000 generations (the 3 last columns). When the optimal solution could be found at least once, the first 3 columns give the number of hits, the best and the average time needed to find the optimum (for the successive runs). The test cases G2 and G3 were run with respectively 20 and 10 variables. The stars denote that each equality constraint has been turned into one inequality constraint – the solution staying the same.

After 5000 gen		After 10000 gen		After 20000 gen	
Avg.	Hits	Avg.	Hits	Avg.	Hits
0.99984	1	0.99990	3	0.99999	7

Table 2. Results of the algorithm on test case G3* for different generation number.

$\tau_{target} = 0.6$		$\tau_{target} = 0.55$		$\tau_{target} = 0.5$		$\tau_{target} = 0.45$		$\tau_{target} = 0.4$		$\tau_{target} = 0.35$	
Avg.	Hits	Avg.	Hits	Avg.	Hits	Avg.	Hits	Avg.	Hits	Avg.	Hits
-30650.745	7	-30656.56	10	-30653.15	15	-30658.93	20	-30664.53	26	-30664.6	25

Table 3. Results of the algorithm on test case G4 for different τ_{target} values.

Otherwise, for the test case G7 and G9, even if the algorithm didn't found the optimum, the best solutions were very close to the optimum value: all results are between 24.36 and 25.75 for G7, and between 680.63 and 680.7 for G9 – and could probably be improved by a careful tuning of the parameters.

6 Conclusion and further work

This paper has introduced ASCHEA, an original evolutionary algorithm for constrained optimization. ASCHEA tries to maintain a given proportion of feasible individuals in the population using an adaptive penalty mechanism coupled with a segregational selection that favors a given number of feasible individuals and a constraint-based choice of mate that allows the detailed exploration of both side of the boundary of the feasible region. However, ASCHEA is also able to explore the interior of the feasible domain as all these features are adaptive, and only triggered when few feasible individuals exist in the population.

ASCHEA gave very good results on eleven test cases from the literature. However, the present study is limited to a single penalty coefficient, and further work will extend the adaptive penalty to multiple penalty coefficients – one per constraint. Moreover, more work is needed to tackle equality constraints. The first experiments indicate that, when using the standard way to turn the constraint $h = 0$ into the double inequality constraint $-\varepsilon < h < \varepsilon$, these two inequalities should be handled independently. Further, the value of ε should be adjusted along evolution by making it adaptive.

References

1. T. Bäck, F. Hoffmeister, and H.-P. Schwefel. A survey of evolution strategies. In R. K. Belew and L. B. Booker, editors, *Proc of the* 4^{th} *Int Conf on Genetic Algorithms*, pages 2–9. Morgan Kaufmann, 1991.
2. A. B. Hadj-Alouane and J. C. Bean. A genetic algorithm for the multiple-choice integer program. Technical Report TR 92-50, Department of Industrial and Operations Engineering, The University of Michigan, 1992.
3. R. Hinterding, Z. Michalewicz, and A. E. Eiben. Adaptation in evolutionary computation: A survey. In T. Bäck, Z. Michalewicz, and X. Yao, editors, *Proc of the Fourth IEEE Int Conf on Evolutionary Computation*, pages 65–69. IEEE Press, 1997.
4. A. Homaifar, S. H.-Y. Lai, and X. Qi. Constrained optimization via genetic algorithms. *Simulation*, 62(4):242–254, 1994.
5. J.A. Joines and C.R. Houck. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs. In Z. Michalewicz, J. D. Schaffer, H.-P. Schwefel, D. B. Fogel, and H. Kitano, editors, *Proc of the First IEEE Int Conf on Evolutionary Computation*, pages 579–584. IEEE Press, 1994.
6. S. Koziel and Z. Michalewicz. Evolutionary algorithms, homomorphous mapping and constrained parameter optimization. *Evolutionary Computation*, 7(1):19–44, 1999.
7. R. G. Leriche, C. Knopf-Lenoir, and R. T. Haftka. A segragated genetic algorithm for constrained structural optimization. In L. J. Eshelman, editor, *Proc of the* 6^{th} *Int Conf on Genetic Algorithms*, pages 558–565, 1995.
8. Z. Michalewicz and N. Attia. Evolutionary optimization of constrained problems. In *Proc of the* 3^{rd} *Annual Conf on Evolutionary Programming*, pages 98–108. World Scientific, 1994.
9. Z. Michalewicz and C. Z. Janikow. Handling constraints in genetic algorithms. In R. K. Belew and L. B. Booker, editors, *Proc of the* 4^{th} *Int Conf on Genetic Algorithms*, pages 151–157. Morgan Kaufmann, 1991.
10. Z. Michalewicz and M. Schoenauer. Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation*, 4(1):1–32, 1996.
11. D. Powell and M. M. Skolnick. Using genetic algorithms in engineering design optimization with non-linear constraints. In S. Forrest, editor, *Proc of the* 5^{th} *Int Conf on Genetic Algorithms*, pages 424–430. Morgan Kaufmann, 1993.
12. E. Ronald. When selection meets seduction. In L. J. Eshelman, editor, *Proc of the* 6^{th} *Int Conf on Genetic Algorithms*, pages 167–173. Morgan Kaufmann, 1995.
13. M. Schoenauer and Z. Michalewicz. Boundary operators for constrained parameter optimization problems. In Th. Bäeck, editor, *Proc of the* 7^{th} *Int Conf on Genetic Algorithms*, pages 322–329. Morgan Kaufmann, 1997.
14. H.-P. Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, New-York, 1981. 1995 – 2^{nd} edition.
15. A. Smith and D. Tate. Genetic optimization using a penalty function. In S. Forrest, editor, *Proc of the* 5^{th} *Int Conf on Genetic Algorithms*, pages 499–503. Morgan Kaufmann, 1993.