# Chapter 1
# A Critical Review of Adaptive Penalty Techniques in Evolutionary Computation

**Helio J.C. Barbosa, Afonso C.C. Lemonge and Heder S. Bernardino**

**Abstract** Constrained optimization problems are common in the sciences, engineering, and economics. Due to the growing complexity of the problems tackled, nature-inspired metaheuristics in general, and evolutionary algorithms in particular, are becoming increasingly popular. As move operators (recombination and mutation) are usually blind to the constraints, most metaheuristics must be equipped with a constraint handling technique. Although conceptually simple, penalty techniques usually require user-defined problem-dependent parameters, which often significantly impact the performance of a metaheuristic. A penalty technique is said to be adaptive when it automatically sets the values of all parameters involved using feedback from the search process without user intervention. This chapter presents a survey of the most relevant adaptive penalty techniques from the literature, identifies the main concepts used in the adaptation process, as well as observed shortcomings, and suggests further work in order to increase the understanding of such techniques.

**Keywords** Adaptive techniques · Penalty techniques · Evolutionary computation

## 1.1 Introduction

Constrained optimization problems are common in the sciences, engineering, and economics. Due to the growing complexity of the problems tackled, nature-inspired metaheuristics in general, and evolutionary algorithms in particular, are becoming

H.J.C. Barbosa (✉)
National Laboratory for Scientific Computing—LNCC, Petropolis, Rio de Janeiro, RJ, Brazil
e-mail: hcbm@lncc.br

A.C.C. Lemonge
Department of Applied and Computational Mechanics, Federal University of Juiz de Fora, Juiz de Fora, MG, Brazil
e-mail: afonso.lemonge@ufjf.edu.br

H.S. Bernardino · H.J.C. Barbosa
Department of Computer Science, Federal University of Juiz de Fora, Juiz de Fora, MG, Brazil
e-mail: heder@ice.ufjf.br

increasingly popular. That is due to the fact that, in contrast to classical mathematical programming techniques, they can be readily applied to situations where the objective function(s) and/or constraints are not known as explicit functions of the decision variables. This happens when potentially expensive computer models (generated by means of the finite element method (Hughes 1987), for example) must be run in order to compute the objective function and/or check the constraints every time a candidate solution needs to be evaluated. For instance, in the design of truss structures, one possible definition of the problem is to find the cross-section areas of the bars that minimize the structure's weight subject to limitations in the nodal displacements and in the stress of each bar (Krempser et al. 2012). Notice that although the structure's weight can be easily calculated from the design variables, the values of the nodal displacements and of the stress in each bar are determined by solving the equilibrium equations defined by the finite element model.

As move operators (recombination and mutation) are usually blind to the constraints (i.e., when operating upon feasible individual(s) they do not necessarily generate feasible offspring) most metaheuristics must be equipped with a constraint handling technique. In simpler situations, repair techniques (Salcedo-Sanz 2009), special move operators (Schoenauer and Michalewicz 1996), or special decoders (Koziel and Michalewicz 1998) can be designed to ensure that all candidate solutions are feasible.

We do not attempt to survey the current literature on constraint handling in this chapter, and the reader is referred to survey papers of, e.g., Michalewicz (1995), Michalewicz and Schoenauer (1996), Coello (2002), and Mezura-Montes and Coello (2011) as well as to the other chapters in this book. Instead we consider the oldest, and perhaps most general class of constraint handling methods: the penalty techniques, where infeasible candidate solutions have their fitness value reduced and are allowed to coexist and evolve with the feasible ones.

Although conceptually simple, penalty techniques usually require user-defined problem-dependent parameters, which often significantly impact the performance of a metaheuristic.

The main focus of this chapter is on adaptive penalty techniques, which automatically set the values of all parameters involved using feedback from the search process without user intervention. This chapter presents a survey of the most relevant adaptive penalty techniques from the literature as well as a critical assessment of their assumptions, rationale for the design choices made, and reported performance on test-problems.

The chapter is structured as follows. Section 1.2 summarizes the penalty method, Sect. 1.3 introduces the main taxonomy for strategy parameter control, and Sect. 1.4 reviews some representative proposals for adapting penalty parameters. Section 1.5 presents a discussion of the main findings and the chapter ends with some conclusions, including suggestions for further work in order to increase the understanding of such adaptive techniques.

## 1.2 The Penalty Method

We consider in this chapter the constrained optimization problem consisting in the minimization of a given objective function $f(x)$, where $x \in \mathbb{R}^n$ is the vector of decision/design variables, which are subject to inequality constraints $g_p(x) \geq 0$, $p = 1, 2, \ldots, \bar{p}$ as well as equality constraints $h_q(x) = 0$, $q = 1, 2, \ldots, \bar{q}$. In many applications the variables are also subject to bounds $x_i^L \leq x_i \leq x_i^U$. However, this type of constraint is usually trivially enforced in an EA and are not considered here. The set of all feasible solutions are denoted by $\mathscr{F}$, while $d(x, \mathscr{F})$ is a distance measure of the element $x$ to the set $\mathscr{F}$. The definition of $d(x, \mathscr{F})$ depends on the particular constraint-handling strategy adopted and is specified for each strategy independently.

The penalty method, which transforms a constrained optimization problem into an unconstrained one, can be traced back at least to the paper by Courant (1943) in the 1940s, and its adoption by the evolutionary computation community happened very soon.

In this chapter, penalty techniques used within evolutionary computation methods are classified as *multiplicative* or *additive*. In the multiplicative case, a positive penalty factor $p(v(x), T)$ is introduced where $v(x)$ denotes a measure of how constraints are violated by the candidate solution $x$ and $T$ denotes a "temperature". The idea is to amplify the value of the fitness function of an infeasible individual (in a minimization problem):

$$F(x) = p(v(x), T) \times f(x).$$

One would have $p(v(x), T) = 1$, for any feasible candidate solution $x$, and $p(v(x), T) > 1$ otherwise. Also, $p(v(x), T)$ increases with the "temperature" $T$ and with the magnitude of the constraint violation $v(x)$. An initial value for the temperature is required together with the definition of a schedule for $T$ such that $T$ grows as the evolution advances. This type of penalty has however received much less attention in the EC community than the additive type. The most recent work seems to be by Puzzi and Carpinteri (2008), where the technique introduced by Yokota et al. (1995) and later modified in Gen and Cheng (1996), is also presented. Harrell and Ranjithan (1999) compare additive and multiplicative penalty techniques for an instance of the watershed management problem.

In the additive case, a penalty functional is added to the objective function in order to define the fitness value of an infeasible element. They can be further divided into: (a) *interior* techniques, when a barrier functional $B(x)$—which grows rapidly as $x \in \mathscr{F}$ approaches the boundary of the feasible domain—is added to the objective function

$$F_k(x) = f(x) + \frac{1}{k}B(x)$$

and (b) *exterior* techniques, where a penalty functional is introduced

$$F_k(x) = f(x) + kP(d(x, \mathscr{F})) \tag{1.1}$$

such that $P(d(x, \mathscr{F})) = 0$ if $x$ is feasible ($x \in \mathscr{F}$) and $P(.) > 0$ otherwise.

In both cases (a) and (b), under reasonable conditions, as $k \to \infty$ any limit point of the sequence $\{x_k\}$ of solutions of the unconstrained problem of minimizing $F_k(x)$ is a solution of the original constrained problem (Luenberger and Ye 2008).

In order to define $d(x, \mathscr{F})$ it is useful to define a measure of the violation of the $j$th constraint by a given candidate solution $x \in \mathbb{R}^n$. One possibility is to take

$$v_j(x) = \begin{cases} |h_j(x)|, & \text{for an equality constraint,} \\ \max\{0, -g_j(x)\} & \text{otherwise} \end{cases} \tag{1.2}$$

However, the equality constraints $h_j(x) = 0$ are often replaced by the inequalities $|h_j(x)| - \varepsilon \leq 0$, for some small positive $\varepsilon$, and one would have

$$v_j(x) = \begin{cases} \max\{0, |h_j(x)| - \varepsilon\}, & \text{for an equality constraint,} \\ \max\{0, -g_j(x)\} & \text{otherwise} \end{cases} \tag{1.3}$$

For computational efficiency the violations $v_j(x)$ are used to compute a substitute for $d(x, \mathscr{F})$ in the design of penalty functions that grow with the vector of violations $v(x) \in \mathbb{R}^m$, where $m = \bar{p} + \bar{q}$ is the number of constraints to be penalized. At this point it is easy to see that *interior* penalty techniques, in contrast to *exterior* ones, require feasible solutions (which are often hard to find) thus explaining the high popularity of the later.

The most popular penalty function is perhaps (Luenberger and Ye 2008)

$$P(x) = \sum_{j=1}^{m} (v_j(x))^2 \tag{1.4}$$

where $P(d(x, \mathscr{F}))$ is equal to the square of the Euclidean norm of $v(x)$.

Although it is conceptually easy to obtain the unconstrained problem, the definition of good penalty parameter(s) is usually a time-consuming, problem dependent, trial-and-error process.

One must also note that even if both the objective function $f(x)$ and distance to the feasible set $\mathscr{F}$ (usually based on the constraint violations $v_j(x)$) are defined for all $x$, it is not possible to know in general which of the two given infeasible solutions is closer to the optimum or should be operated upon or kept in the population. One can have $f(x_1) > f(x_2)$ and $\|v(x_1)\| = \|v(x_2)\|$ or $f(x_1) = f(x_2)$ and $\|v(x_1)\| > \|v(x_2)\|$ and still have $x_1$ closer to the optimum. Figure 1.1 illustrates these situations.

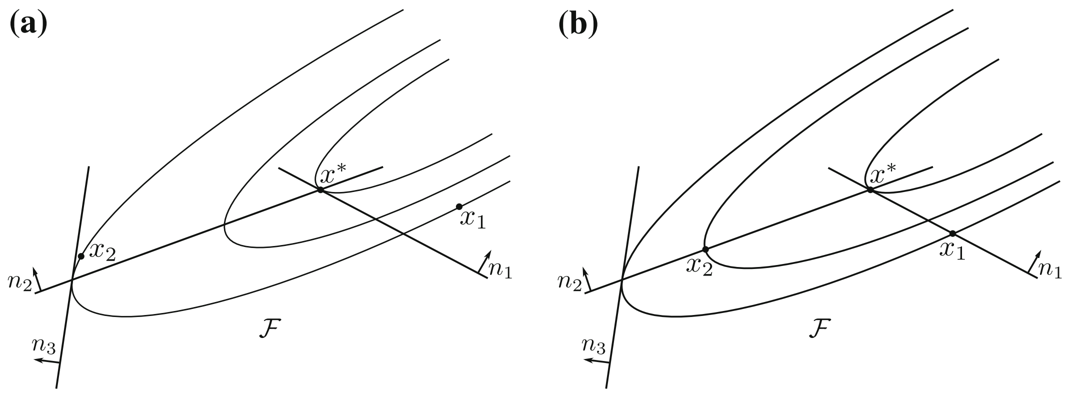

**Fig. 1.1** Illustration of situations in which $x_1$ is closer to the optimum ($x^*$) than $x_2$ even when: **a** $f(x_1) = f(x_2)$ and $\|v(x_1)\| > \|v(x_2)\|$; or **b** $f(x_1) > f(x_2)$ and $\|v(x_1)\| = \|v(x_2)\|$

## 1.3 A Taxonomy

In order to organize the large amount of penalty techniques available in the literature Coello (2002) proposed the following taxonomy: (a) static penalty, (b) dynamic penalty, (c) annealing penalty, (d) adaptive penalty, (e) co-evolutionary penalty, and (f) death penalty. We think however that the general definitions proposed by Eiben and Smith (2003) with respect to the way strategy parameters are set within metaheuristics in general and evolutionary algorithms in particular can be naturally adopted here.

Beyond the simplest *static* case where strategy parameters are defined by the user and remain fixed during the run, *dynamic* schemes have been also used where an exogenous schedule is proposed in order to define the strategy parameters at any given point in the search process. It is easy to see that if setting fixed parameters is not trivial, defining the way they should vary during the run seems to be even harder. It is also felt that such strategy parameters should not be defined before the run but rather vary according to what is actually happening in the search process. This gives rise to the so-called *adaptive* techniques, where feedback from the search process is used to define the current strategy parameters.

From the reasoning above, the death penalty can be included as a particular case of *static* penalty, and the annealing penalty can be seen as a *dynamic* penalty scheme. Co-evolutionary penalty techniques are considered in Sect. 1.5.2.

It should be noted here that the design of the adaptive mechanisms mentioned above often involve meta-parameters or, at least, implicit design choices. The rationale here is that such meta-parameters should be easier to set appropriately; preferably fixed by the designer, with no posterior user intervention required. However, the parameter setting in some adaptive techniques can be as hard as in the case of the static ones (Coello 2002), contradicting the main objective of the adaptive penalty methods.

Finally, an even more ambitious proposal can be found in the literature: the *self-adaptive* schemes. In this case, strategy parameters are coded together with the candidate solution, and conditions are created so that the evolutionary algorithm not only evolves increasingly better solutions but also better adapted strategy parameters. With this increasing sophistication in the design of the algorithms one not only seeks to improve performance but also to relieve the user from the task of strategy parameter setting and control.

However, as will be shown in the next section, another possibility, which has not been contemplated in the taxonomy considered above, can be found in the literature for the task of automatically setting strategy parameters. The idea is to maintain an additional population with the task of *co-evolving* such strategy parameters (here penalty coefficients) along with the standard population evolving the solutions to the constrained optimization problem at hand.

## 1.4 Some Adaptive Techniques

In this section some selected papers from the literature are reviewed in order to provide an overview of the diversity of techniques proposed for automatically setting parameters involved in the various penalty schemes for constrained optimization. Such techniques not only intend to relieve the user from the task of parameter setting for each new application but also to improve the final performance in the case at hand by adapting the values of those parameters along the search process in a principled way. Table 1.1 presents a summary of the adaptive penalty techniques cited in this section. Some references are not included in the table as their work extends a previous one but do not require any additional information.

The main lines of reasoning have been identified and a few representative proposals of each line have been grouped together in the following subsections.

### *1.4.1 The Early Years*

A procedure where the penalty parameters change according to information gathered during the evolution process was proposed by Bean and Alouane (1992). The fitness function is again given by (1.1) but with the penalty parameter $k = \lambda(t)$ adapted at each generation by the following rules:

$$\lambda(t+1) = \begin{cases} \beta_1^{-1}\lambda(t), & \text{if } b^i \in \mathscr{F} \quad \text{for all } t-g+1 \le i \le t \\ \beta_2\lambda(t), & \text{if } b^i \notin \mathscr{F} \quad \text{for all } t-g+1 \le i \le t \\ \lambda(t) & \text{otherwise} \end{cases}$$

where $b^i$ is the best element at generation $i$, $\mathscr{F}$ is the feasible region, $\beta_1 \neq \beta_2$ and $\beta_1, \beta_2 > 1$. In this method the penalty parameter of the next generation $\lambda(t+1)$ decreases when all best elements in the last $g$ generations were feasible, increases if all best elements were infeasible and otherwise remains without change.

Table 1.1 Summary of the adaptive penalty techniques described here

| Reference | Used information |
|---|---|
| Bean and Alouane (1992) | Feasibility of the best |
| Coit et al. (1996) | Degree of infeasibility |
| | Difference between the fitnesses of the best and best feasible individuals |
| Hamida and Schoenauer (2000) | Percentage of feasible individuals |
| | Ratio between the sum of the objective function values and constraint violations |
| Nanakorn and Meesomklin (2001) | Mean of the objective function values of the feasible solutions |
| Beaser et al. (2011) | Average of the objective function values |
| | Degree of infeasibility |
| Barbosa and Lemonge (2003b) | Average of the objective function values |
| Lemonge and Barbosa (2004) | Average of the violation values of each constraint |
| Rocha and Fernandes (2009) | |
| Farmani and Wright (2003) | Normalized violation values |
| | Objective function value of the worst solution |
| Lin and Wu (2004) | Percentage of feasible solutions with respect to each constraint |
| | Rate between the objective function value and a given constraint violation |
| | Fitness of the best solution |
| | Number of objective function evaluations |
| | Difference between the medians of the objective function values of feasible and inFeasible solutions |
| | Ratio of the previous value and the median of the Constraint violations |
| Tessema and Yen (2006, 2009) | Percentage of feasible solutions |
| | Average of the normalized constraint violation values |
| | Normalized objective function value |
| Wang et al. (2009) | Degree of infeasibility |
| | Percentage of feasible solutions |
| Gan et al. (2010) | Percentage of Feasible solutions |
| Costa et al. (2013) | Degree of infeasibility |
| | Objective function value of the worst solution |
| | Constraint violation of the equality constraints for the best solution |
| Vincenti et al. (2010) | Objective function value of the best feasible solution |
| Montemurro et al. (2013) | Objective function value of the best infeasible solution |
| | Difference between the two previous values |
| | Ratio between the previous difference and the violation value of each constraint |

The method proposed by Coit et al. (1996), uses the fitness function $F(x)$ written as

$$F(x) = f(x) + (F_{\text{feas}} - F_{\text{all}}) \sum_{j=1}^{m} \left( \frac{d_j(x, \mathscr{F})}{NFT_j} \right)^{K_j}$$

where $f(x)$ is the unpenalized objective function for the solution $x$, $F_{\text{all}}$ corresponds to the best solution already found, $F_{\text{feas}}$ corresponds to the best feasible solution already found, and $d_j(x, \mathscr{F})$ returns the distance between $x$ and the feasible region (dependent of the problem). $K_j$ and $NFT_j$, the near-feasible threshold of the $j$th constraint, are user-defined parameters.

Rasheed (1998) proposed an adaptive penalty approach for handling constraints within a GA. The strategy required the user to set a relatively small penalty parameter and then it would increase or decrease it on demand as the optimization progresses. The method was tested in a realistic continuous-variable conceptual design of a supersonic transport aircraft, and the design of supersonic missile inlets, as well as in benchmark engineering problems. The fitness of each individual was based on the sum of an adequate measure of merit computed by a simulator (such as the take-off mass of an aircraft). If the fitness value is between $V$ and $10 \times V$, where $V$ is a power of 10, the penalty coefficient starts with the value $\frac{V}{100}$. The proposed algorithm featured two points: (i) the individual that has the least sum of constraint violations and (ii) the individual that has the best fitness value. The penalty coefficient is considered adequate if both individuals are the same and otherwise the penalty coefficient is increased to make the two solutions have equal fitness values. The author concluded that the idea of starting with a relatively small initial penalty coefficient and increasing it or decreasing it on demand proved to be very good in the computational experiments conducted.

Hamida and Schoenauer (2000) proposed an adaptive scheme named as Adaptive Segregational Constraint Handling Evolutionary Algorithm (ASCHEA) employing: (i) a function of the proportion of feasible individuals in the population; (ii) a seduction/selection strategy to mate feasible and infeasible individuals applying a specific feasibility-oriented selection operator, and (iii) a selection scheme to give advantage for a given number of feasible individuals. The ASCHEA algorithm was improved (Hamida and Schoenauer 2002) by considering a niching technique with adaptive radius to handle multimodal functions and also (i) a segregational selection that distinguishes between feasible and infeasible individuals, (ii) a constraint-driven recombination, where in some cases feasible individuals can only mate with infeasible ones, and (iii) a population-based adaptive penalty method that uses global information on the population to adjust the penalty coefficients. Hamida and Schoenauer (2002) proposed the following penalty function:

$$P(x) = \sum_{j=1}^{m} \alpha_j v_j(x) \tag{1.5}$$

where $\alpha_j$ is adapted as

$$\begin{cases} \alpha_j(t+1) = \alpha_j(t)/\text{fact} & \text{if } (\tau_t(j) > \tau_{\text{target}}) \\ \alpha_j(t+1) = \alpha_j(t) \times \text{fact} & \text{otherwise} \end{cases} \tag{1.6}$$

where fact $> 1$ and $\tau_{\text{target}}$ are to be defined by the user (although the authors suggest $\tau_{\text{target}} = 0.5$), and $\tau_t(j)$ is the proportion of individuals which do not violate the $j$th constraint. The idea is to have feasible and infeasible individuals on both sides of the corresponding boundary. The adapted parameter $\alpha_j$, with initial value $\alpha_j(0)$, are computed using the first population, trying to balance objective function and constraint violations:

$$\begin{cases} \alpha_j(0) = 1, & \text{if } \sum_i^n v_j(x_i) = 0 \\ \alpha_j(0) = \frac{\sum_i^n |f(x_i)|}{\sum_i^n |v_j(x_i)|} \times 100, & \text{otherwise} \end{cases} \tag{1.7}$$

The early proposals reviewed here were not able in general to adequately deal with the problem, suggesting that more information from the search process, at the price of added complexity, was required.

### 1.4.2 Using More Feedback

Nanakorn and Meesomklin (2001) proposed an adaptive penalty function for a GA that is able to adjust itself during the evolutionary process. According to that method, the penalty is such that

$$F(x) \leq \phi(t) f_{\text{avg}} \quad \text{for } \forall x \notin \mathscr{F} \tag{1.8}$$

where $f_{\text{avg}}$ represents the average fitness value of all feasible individuals in the current generation and $\phi(t)$ depends on $f_{\text{avg}}$. Thus, the fitness function is defined as

$$F(x) = f(x) - \lambda(t) E(x) \tag{1.9}$$

where

$$E(x) = \sum_{i=1}^{m} v_i(x) \tag{1.10}$$

The adaptive parameter $\lambda(t)$ is written as

$$\lambda(t) = \max\left\{0, \max_{\forall x \notin \mathscr{F}} \left[\frac{f(x) - \phi(t) f_{\text{avg}}}{E(x)}\right]\right\}. \tag{1.11}$$

The function $\phi(t)$ is defined according to the user defined parameter $\varphi$. If $\varphi \geq 1$ then

$$\phi(t) = \frac{Cf_{\text{avg}} + F_{\max}(\varphi - 1) - \varphi f_{\text{avg}}}{(C-1) f_{\text{avg}}} \tag{1.12}$$

where $C$ is a user defined parameter which is the maximum scaled fitness value assigned to the best feasible member. The scaled fitness values are used only in the selection procedure and will not be described here.

Otherwise (if $\varphi < 1$), then $\phi(t)$ is defined by an iterative process which is initialized with $\phi(t) = 1$ and is repeated until the value of $\phi(t)$ becomes unchanging. The steps of the procedure are

(i) to calculate $\lambda$ by means of Eq. (1.11)
(ii) to evaluate the candidate solutions according to Eq. (1.9)
(iii) to obtain $x_{\min}$ and $x_\lambda$, where $F_{\min} = F(x_{\min})$ is the minimum value of $F$ and $x_\lambda$ is the candidate solution that leads to

$$\lambda(t) = \frac{F(x_\lambda) - \phi(t) f_{\text{avg}}}{E(x_\lambda)} \tag{1.13}$$

(iv) $\phi(t)$ is updated by

$$\phi(t) = \frac{(\varphi - 1) E(x_{\min}) F(x_\lambda) + E(x_\lambda)\left[F(x_{\min}) + \varphi f_{\text{avg}} - \varphi F(x_{\min})\right]}{f_{\text{avg}}\left[E(x_\lambda) + (\varphi - 1) E(x_{\min})\right]} \tag{1.14}$$

Beaser et al. (2011) updates the adaptive penalty function theory proposed by Nanakorn and Meesomklin (2001), expanding its validity beyond maximization problems to minimization as well. The expanded technique, using a hybrid genetic algorithm, was applied to a problem in chemistry.

The first modification was introduced in the Eq. (1.8):

$$F(x) \geq \phi(t) f_{\text{avg}} \quad \text{for } \forall x \notin \mathscr{F} \tag{1.15}$$

Then, the modified value for the parameter $\lambda(t)$ is defined as

$$\lambda(t) = \min\left\{0, \min_{\forall x \notin \mathscr{F}}\left[\frac{f(x) - \phi(t) f_{\text{avg}}}{E(x)}\right]\right\} \tag{1.16}$$

An adaptive decision maker (ADM) proposed by Gan et al. (2010) is designed in the form of an adaptive penalty function. The method decides which individual is maintained in a Pareto optimal set and decides which individuals are going to be replaced. The fitness function in this strategy is written as usual:

$$F(x) = f(x) + C \cdot G(x). \tag{1.17}$$

A parameter $r_f$ is introduced denoting the proportion of feasible solutions in the population and $C$ is designed as a function of $r_f$, i.e., $C(r_f)$, and two basic rules need to be satisfied: (1) It should be a decreasing function, because the coefficient $C$ decreases as $r_f$ increases and, (2) When $r_f$ varies from 0 to 1, $C$ decreases sharply from a large number at the early stage, and decreases slowly to a small number at the late stage. The reason is that, with $r_f$ increasing (it means that there are more and more feasible solutions in the population), the search emphasis should shift from low constraint violations to good objective function values quickly. The proposed function that satisfies these two rules is expressed as $C(r_f) = 10^{\alpha(1-r_f)}$, where $\alpha$ is a positive constant coefficient to be adjusted, and the fitness function is rewritten as

$$F(x) = f(x) + 10^{\alpha(1-r_f)} \cdot G(x) \tag{1.18}$$

Besides, two properties are established: (1) the fitness assignment maps the two-dimensional vector into the real number space: in this way, it is possible to compare the solutions in the Pareto optimal set, selecting which one is preferable and (2) the penalty coefficient $C$ varies with the feasibility proportion of the current population and, if there are no feasible solutions in the population, this parameter will receive a relatively large value in order to guide the population in the direction of the feasible space.

The common need for user-defined parameters together with the difficulty of finding adequate parameter values for each new application pointed the way to the challenge of designing penalty techniques which do not require such parameters.

### 1.4.3 Parameterless Techniques

A parameterless adaptive penalty scheme for GAs was proposed by Barbosa and Lemonge (2003b), which does not require the knowledge of the explicit form of the constraints as a function of the design variables and is free of parameters to be set by the user. In contrast with other approaches where a single penalty parameter is used for all constraints, an adaptive scheme automatically sizes the penalty parameter corresponding to *each* constraint along the evolutionary process. The fitness function proposed is written as

$$F(x) = \begin{cases} f(x), & \text{if } x \text{ is feasible,} \\ f(x) + \displaystyle\sum_{j=1}^{m} k_j v_j(x), & \text{otherwise} \end{cases} \tag{1.19}$$

The penalty parameter is defined at each generation by

$$k_j = |\langle f(x) \rangle| \frac{\langle v_j(x) \rangle}{\sum_{l=1}^{m} [\langle v_l(x) \rangle]^2} \tag{1.20}$$

where $\langle f(x) \rangle$ is the average of the objective function values in the current population and $\langle v_l(x) \rangle$ is the violation of the $l$th constraint averaged over the current population.

The idea is that the values of the penalty coefficients should be distributed in a way that those constraints that are more difficult to be satisfied should have a relatively higher penalty coefficient.

With the proposed definition one can prove the following property: an individual whose $j$th violation equals the average of the $j$th violation in the current population for all $j$, has a penalty equal to the absolute value of the average fitness function of the population.

The performance of the APM was examined using test problems from the evolutionary computation literature as well as structural engineering constrained optimization problems but the algorithm presented difficulties in solving some benchmark problems, for example, the functions $G_2$, $G_6$, $G_7$ and $G_{10}$ proposed by Michalewicz and Schoenauer (1996). That was improved in the conference paper (Barbosa and Lemonge 2002), where $f(x)$ in the definition of the objective function of the infeasible individuals in Eq. (1.19) was changed to

$$\bar{f}(x) = \begin{cases} f(x), & \text{if } f(x) > \langle f(x) \rangle, \\ \langle f(x) \rangle & \text{otherwise} \end{cases} \tag{1.21}$$

and $\langle f(x) \rangle$ is the average of the objective function values in the current population. The new version was tested (Lemonge and Barbosa 2004) in benchmark engineering optimization problems and in the $G$–Suite (Michalewicz and Schoenauer 1996) with a more robust performance.

The procedure proposed by Barbosa and Lemonge (2002), originally conceived for a generational GA, was extended to the case of a steady-state GA (Barbosa and Lemonge 2003a), where, in each "generation", usually only one or two new individuals are introduced in the population. Substantial modifications were necessary to obtain good results in a standard test-problem suite (Barbosa and Lemonge 2003a). The fitness function for an infeasible individual is now computed according to the equation:

$$F(x) = H + \sum_{j=1}^{m} k_j v_j(x) \tag{1.22}$$

where $H$ is defined as

$$H = \begin{cases} f(x_{\text{worst}}) & \text{if there is no feasible element in the population,} \\ f(x_{\text{bestFeasible}}) & \text{otherwise} \end{cases} \tag{1.23}$$

and the penalty coefficients are redefined as

$$k_j = H \frac{\langle v_j(x) \rangle}{\sum_{l=1}^{m} [\langle v_l(x) \rangle]^2} \tag{1.24}$$

Also, every time a better feasible element is found (or the number of new elements inserted into the population reaches a certain level), $H$ is redefined and all fitness values are recomputed. The updating of each penalty coefficient is performed in such a way that no reduction in its value is allowed. The fitness function value is then computed using Eqs. (1.22)–(1.24). It is clear from the definition of $H$ in (1.23) that if no feasible element is present in the population, one is actually minimizing a measure of the distance of the individuals to the feasible set, since the actual value of the objective function is not taken into account. However, when a feasible element is found then it immediately enters the population as, after updating all fitness values using (1.19), (1.23), and (1.24), it becomes the element with the best fitness value.

Later, APM variants were introduced with respect to the definition of the penalty parameter $k_j$ (Barbosa and Lemonge 2008). The APM, as originally proposed, computes the constraint violations in the initial population, and updates all penalty coefficients, for each constraint, after a given number of offspring is inserted in the population. A second variant, called sporadic APM with constraint violation accumulation, accumulates the constraint violations during a given number of insertions of new offspring in the population, updates the penalty coefficients, and keeps the penalty coefficients for the next generations. The APM with monotonic penalty coefficients is the third variant, where the penalty coefficients are calculated as in the original method, but no penalty coefficient is allowed to have its value reduced along the evolutionary process. Finally, the penalty coefficients are defined by using a weighted average between the previous value of a coefficient and the new value predicted by the method. This variant is called the APM with damping. Besides that, these variants of the APM were extended to the steady-state GA and presented in Lemonge et al. (2012).

Rocha and Fernandes (2009) proposed alternative expressions for the APM penalty coefficients

$$k_j = \left|\sum_{i=1}^{pop} f(x^i)\right| \frac{\sum_{i=1}^{pop} v_j(x^i)}{\sum_{k=1}^{m}\sum_{i=1}^{pop} v_k(x^i)}$$

and also

$$k_j = \left|\sum_{i=1}^{pop} f(x^i)\right| \left(\exp\left(\frac{\sum_{i=1}^{pop} v_j(x^i)}{\sum_{k=1}^{m}\sum_{i=1}^{pop} v_k(x^i)}\right)^{l} - 1\right)$$

with $l \in \{1, 2\}$.

Farmani and Wright (2003) introduced a parameterless adaptive technique that uses information about the degree of infeasibility of solutions written as

$$u(x) = \frac{1}{m}\sum_{j=1}^{m} \frac{v_j(x)}{v_j^{\max}} \tag{1.25}$$

where $m$ is the total number of inequality and equality constraints, and $v_j^{\max}$ is the maximum value of the $j$th violation in the current population. The $x_{\text{worst}}$ of the infeasible solutions is selected by comparing all infeasible individuals against the best individual $x_{\text{best}}$. Two potential population distributions exist in relation to this: (i) if one or more of the infeasible solutions have an objective function value that is lower than the $f(x_{\text{best}})$, the $f(x_{\text{worst}})$ of the infeasible solutions is taken as the infeasible solution having the highest infeasibility value and an objective function value that is lower than the $f(x_{\text{best}})$ solution. If more than one individual exists with the same highest degree of infeasibility, then $f(x_{\text{worst}})$ is taken as the solution with maximum infeasibility value and the lower of the objective function values, and (ii) when all of the infeasible solutions have an objective function value that is greater than $f(x_{\text{best}})$. Thus, $f(x_{\text{worst}})$ is identified as being the solution with the highest degree of infeasibility value. Having more than one individual in the population with the same highest infeasibility value, then $f(x_{\text{worst}})$ is taken as the solution with the maximum infeasibility value and the higher of the objective function values. The highest objective function value in the current population to penalize the infeasible individuals is defined as $f_{\max}$. The method is applied in two stages where the first stage considers the case where one or more infeasible solutions have a lower and potentially better objective function (minimization problem) than the $x_{\text{best}}$ solution and $\left(\exists x | f(x) < f(x_{\max}) \bigwedge (u(x) > 0.0)\right)$. A linear relationship between the degree of infeasibility of the $x_{\text{best}}$ and $x_{\text{worst}}$ is considered as

$$\tilde{u}(x) = \frac{u(x) - u(x_{\text{worst}})}{u(x_{\text{best}}) - u(x_{\text{worst}})} \tag{1.26}$$

Thus, the fitness function $F_{1st}(x)$, in the first stage, is written as

$$F_{1st}(x) = f(x) + \tilde{u}(x)(f(x_{\max}) - f(x_{\text{worst}})) \tag{1.27}$$

The second stage increases the objective function such that the penalized objective function of the "worst" infeasible individual $F_{2nd}(x)$ is equal to the "worst" objective individual (Eqs. (1.28) and (1.29)).

$$F_{2nd}(x) = F_{1st}(x) + \gamma |\langle F_{1st}(x) \rangle| \left( \frac{\exp(2.0\tilde{u}(x)) - 1}{\exp(2.0) - 1} \right) \tag{1.28}$$

and

$$\gamma = \begin{cases} \frac{f(x_{\max}) - f(x_{\text{best}})}{f(x_{\text{best}})}, & \text{if } f(x_{\text{worst}}) \le f(x_{\text{best}}) \\ 0, & \text{if } f(x_{\text{worst}}) = f(x_{\max}). \\ \frac{f(x_{\max}) - f(x_{\text{worst}})}{f(x_{\text{worst}})}, & \text{if } f(x_{\text{worst}}) > f(x_{\text{best}}) \end{cases} \tag{1.29}$$

The scaling factor $\gamma$, is introduced to ensure that the penalized value of "worst" infeasible solution is equivalent to the highest objective function value in the current population. $\gamma = 0$ (second case in Eq. (1.29)) is used when the "worst" infeasible individual has an objective function value equal to the highest in the population. In this case, no penalty is applied since the infeasible solutions would naturally have a low fitness and should not be penalized further. The use of absolute values of the fitness function in Eq. (1.29) is considered since minimization of objective functions may have negative values. The use of absolute values of the fitness function in Eq. (1.29) is considered since minimization of objective functions may have negative values.

A self-organizing adaptive penalty strategy (SOAPS) is presented in Lin and Wu (2004) featuring the following aspects: (1) The values of penalty parameters are automatically determined according to the population distribution; (2) The penalty parameter for each constraint is independently determined; (3) The objective and constraint functions are automatically normalized; (4) No parameters need to be defined by the user; (5) Solutions are maintained evenly distributed in both feasible and infeasible parts of each constraint. The "pseudo objective function" defined by the proposed algorithm is given as

$$F(x) = f(x) + P(x) \tag{1.30}$$

where the penalty function $P(x)$ is written as

$$P(x) = \frac{100+t}{100} \times \frac{1}{\bar{p}+2\bar{q}} \times \sum_{j=1}^{m} r_j^t \times v_j(x) \tag{1.31}$$

where $t$ is the generation, $r_j^t$ is the penalty parameter for the $j$th constraint at generation $t$, and $\bar{p}$ and $\bar{q}$ are the number of inequality and equality constraints, respectively.

The penalty parameter $r_j^t$ for the $j$th constraint at the $t$th generation is set as

$$r_j^t = r_j^{t-1} \times \left(1 - \frac{\tau_t(j) - 0.5}{5}\right), \qquad t \geq 1 \tag{1.32}$$

where $\tau_t(j)$ is the percentage of feasible solutions with respect to the $j$th constraint at the $t$th generation. This parameter will be adapted during the evolutionary process and its initial value is set as

$$r_j^0 = \frac{QR_{\text{obj}}^1}{QR_{\text{con}j}^1} \tag{1.33}$$

where $QR_{\text{obj}}^1$ and $QR_{\text{con}j}^1$ are the interquartile ranges of the objective function and the $j$th constraint function values, respectively, in the initial population.

Although the proposed algorithm performed satisfactorily on constrained optimization problems with inequality constraints, it had difficulties in solving problems with equality constraints. The authors presented in the same paper (Wu and Lin 2004) a modification (with added complexity) of the first version of the algorithm. They detected that the initial penalty parameter for a constraint may become undesirably large due to the poor initial population distribution. A sensitivity analysis of the parameter $r_j^0$ was done by the authors and they concluded that enlarged penalties undesirably occur because solutions with these unexpected large constraint violations are not evenly sampled in the initial population. The value for $F(x)$ in the second generation of SOAPS is written as

$$F(x) = \begin{cases} f(x), & \text{if } x \in \mathscr{F} \\ f(x) \times (1 - r_{\text{GEN}}) + F_{\text{BASE}} \times r_{\text{GEN}} + P(x) & \text{otherwise} \end{cases} \tag{1.34}$$

where $F_{\text{BASE}}$ means the minimum value of all feasible solutions or, in the absence of them, the infeasible solutions with the smallest amount of constraint violation. The value of $r_{\text{GEN}}$ is given by the number of function evaluations performed so far divided by the total number of function evaluations. The expression for $P(x)$ is

$$P(x) = \sum_j r_j^q \times v_j(x) \tag{1.35}$$

The modified initial penalty coefficient is rewritten as

$$r_j^0 = \begin{cases} \dfrac{\text{med}^1_{\text{obj,feas}j} - \text{med}^1_{\text{obj,infeas}j}}{\text{med}^1_{\text{con}j}} & \text{if } \text{med}^1_{\text{obj,feas}j} \geq \text{med}^1_{\text{obj,infeas}j} \\ 0.5 \times \dfrac{\text{med}^1_{\text{obj,infeas}j} - \text{med}^1_{\text{obj,feas}j}}{\text{med}^1_{\text{con}j}} & \text{otherwise} \end{cases} \tag{1.36}$$

where $\text{med}^1_{\text{obj,feas}j}$ is the median of the objective function value of the feasible solutions, and $\text{med}^1_{\text{obj,infeas}j}$ is the median of all infeasible solutions with respect to the $j$th constraint, in the initial population. The value $\text{med}^1_{\text{con}j}$ represents the median of all constraint violations of the $j$th constraint in the initial population. The value of $\text{med}_{\text{obj,feas}}$, used in Eq. (1.36), is written as

$$\begin{aligned} \text{med}_{\text{obj,feas}} &= \text{med}_{\Phi,\text{feas}} = \text{med}_{\Phi,\text{infeas}} \\ &= \text{med}_{\text{obj,infeas}} + r \times \text{med}_{\text{con}} \end{aligned} \tag{1.37}$$

where $\text{med}_{\Phi,\text{feas}}$ is the median of the pseudo-objective function values of feasible designs, and $\text{med}_{\Phi,\text{infeas}}$ is the median of the pseudo-objective function values of infeasible designs. The latter, $\text{med}_{\Phi,\text{infeas}}$ consists of $\text{med}_{\text{obj,infeas}}$ the median of objective function values of all infeasible designs and $\text{med}_{\text{con}}$, the median of constraint violations of all infeasible designs. The second generation of SOAPS was tested in two numerical illustrative problems and one engineering problem.

Tessema and Yen (2006) proposed an adaptive penalty function for solving constrained optimization problems using a GA. A new fitness value, called distance value, in the normalized fitness-constraint violation space, and two penalty values are applied to infeasible individuals so that the algorithm would be able to identify the best infeasible individuals in the current population. The performance of the algorithm was tested on the $G_1$ to $G_{13}$ test-problems and the algorithm was considered able to find competitive results when compared with others from the literature.

In (Tessema and Yen 2009) an algorithm that aims to exploit infeasible individuals with low objective value and low constraint violation was proposed. The fraction of feasible individuals in the population is used to guide the search process either toward finding more feasible individuals or searching for the optimum solution. The objective function of all individuals in the current population will be evaluated first, and the smallest and the largest values will be identified as $f_{\min}$ and $f_{\max}$, respectively. The fitness function of each individual is normalized as

$$\tilde{f}(x) = \frac{f(x) - f_{\min}}{f_{\max} - f_{\min}} \tag{1.38}$$

The normalized constraint violation of each infeasible individual is evaluated by Eq. (1.25) and the modified fitness function is then written as

$$F(x) = \begin{cases} \tilde{f}(x), & \text{for a feasible solution} \\ u(x), & \text{if there is no feasible ind.} \\ \sqrt{\tilde{f}(x)^2 + u(x)2} + \left[(1 - r_f)u(x) + r_f\tilde{f}(x)\right], & \text{otherwise} \end{cases}$$

where $r_f \in [0, 1]$ is the fraction of feasible individuals in the population, and $u(x)$ is the average of the normalized violations $(v_j(x))$.

A hybrid evolutionary algorithm and an adaptive constraint-handling technique is presented by Wang et al. (2009). The hybrid evolutionary algorithm simultaneously uses simplex crossover and two mutation operators to generate the offspring population. The proposed method operates upon three types of population: (1) a population that contains only infeasible solutions, (infeasible situation), (2) a population that contains feasible and infeasible solutions, (semi-feasible situation), and (3) a population that contains only feasible solutions, (feasible situation). Denoting $G(x) = \sum_{j=1}^{m} G_j(x)$ as the degree of constraint violation of the individual $x$, one has

1. Infeasible situation: the constrained optimization problem is treated as a constraint satisfaction problem. Thus, finding feasible solutions is the most important objective in this situation. To achieve this, the constraint violations $G(x)$ of the individuals in the population, and the objective function $f(x)$ is disregarded completely. First, the individuals in the parent population are ranked based on their constraint violations in ascending order, and then the individuals with the least constraint violations are selected and form the offspring population.
2. Semi-feasible situation: the population is divided into the feasible group $K_1$ and the infeasible group $K_2$. After that, the best feasible $x_{\text{best}}$ and the worst feasible solutions $x_{\text{worst}}$ are identified from the feasible group $K_1$. Then, the objective function $f(x)$ of a candidate solution is written as

$$f'(x_i) = \begin{cases} f(x_i), & \text{if } x_i \in K_1 \\ \max\{\phi f(x_{\text{best}}) + (1-\phi)f(x_{\text{worst}}), f(x_i)\} & \text{if } x_i \in K_2 \end{cases} \tag{1.39}$$

where $\phi$ is the proportion of feasible solutions in the last population $P(t)$. The normalized objective function is obtained using the Eq. (1.38). Also, the normalized constraints are written as

$$\tilde{G}(x_i) = \begin{cases} 0, & \text{if } x_i \in K_1 \\ \dfrac{G(x_i) - \min\limits_{x \in K_2} G(x)}{\max\limits_{x \in K_2} G(x) - \min\limits_{x \in K_2} G(x)}, & \text{if } x_i \in K_2 \end{cases} \tag{1.40}$$

If only one infeasible solution appears in the population, the normalized constraint violation $\tilde{G}$ of such individual will always be equal to 0. To avoid it, the normalized constraint violation $\tilde{G}$ of such individual is set to a value uniformly chosen between 0 and 1. The fitness function is defined by adding the normalized objective function values and constraint violations and defined as

$$F(x_i) = \tilde{f}(x_i) + \tilde{G}(x_i) \tag{1.41}$$

3. Feasible situation: in this case, the comparisons of individuals are based only on the objective function $f(x)$.

Costa et al. (2013) proposed an adaptive constraint handling technique where the fitness function of an infeasible individual is defined as

$$F(x) = f_{\max} + \sum_{j=1}^{m} v_j(x) \tag{1.42}$$

and $v_j(x)$ is defined as in Eq. (1.3). An adaptive tolerance was introduced in order to handle equality constraints. An initial tolerance $\varepsilon^0$ is defined and it is adaptively updated along the evolutionary process, with a periodicity of $\gamma$ generations, according to the expression:

$$\varepsilon^{k+1} = \alpha \varepsilon^k + (1 - \alpha)\|C_{\text{best}}\|_2 \tag{1.43}$$

where $\alpha$ is a smoothing factor, $C_{\text{best}}$ is the vector of equality constraints for the best point in the population, and $\|\cdot\|_2$ is the Euclidean norm.

A parameterless adaptive penalty technique used within a GA has been proposed in Vincenti et al. (2010), Montemurro et al. (2013) where the basic idea is that some good infeasible individuals (in the sense of having good objective function values) can be useful to attract the exploration toward the boundary of the feasible domain, as the optimum usually has some active constraints. The penalty coefficients $c_i$ and $q_j$ (for equality and inequality constraints, respectively) are computed at each generation $t$ as

$$c_i(t) = \frac{\left|f^F_{\text{best}} - f^{NF}_{\text{best}}\right|}{(g_i)^{NF}_{\text{best}}} \quad i = 1, \ldots, \bar{q} \quad \text{and} \quad q_j(t) = \frac{\left|f^F_{\text{best}} - f^{NF}_{\text{best}}\right|}{(h_j)^{NF}_{\text{best}}} \quad j = 1, \ldots, \bar{p} \tag{1.44}$$

where the superscripts $F$ and $NF$ stand for feasible and non-feasible, respectively. $f^{F}_{\text{best}}$ and $f^{NF}_{\text{best}}$ are the values of the objective function for the best individuals within the feasible and the infeasible sides of the domain, respectively, while $(g_i)^{NF}_{\text{best}}$ and $\left(h_j\right)^{NF}_{\text{best}}$ represent the violation of inequality and equality constraints, respectively, for the best infeasible solution.

Individuals that are infeasible with respect to the $k$th constraint are grouped and ranked with respect to their objective function values: the objective function of the best individual of such a group is $f^{NF}_{\text{best}}$ while the individuals that are feasible with respect to the $k$th constraint are grouped and ranked with respect to their objective function values: the objective function of the best individual of this group is $f^{F}_{\text{best}}$.

When no feasible individuals are available in the population with respect to the $k$th constraint, the population is then sorted into two groups: individuals having smaller values of the $k$th constraint violation (10 % of the population) are grouped as "virtually feasible" while the rest are grouped as infeasible and ranked in terms of their objective function values: the objective function of the best individual of such a group is $f^{NF}_{\text{best}}$.

It is worth noting that the definition in Eq. (1.44) forces the value of the objective function of the best infeasible individual to be equal to that of the best feasible individual. In the next section, further (perhaps less popular) ways of implementing penalty techniques are briefly described.

## 1.5 Related Techniques

### *1.5.1 Self-adapting the Parameters*

The direct implementation of a standard self-adaptive penalty technique (following Eiben and Smith (2003)) would entail the encoding of one (or more) penalty coefficients in the same chromosome where the candidate solution is encoded. They are then subject to the evolutionary process, undergoing recombination and mutation just as the problem variables in the chromosome. However, evolution would "discover" that the best strategy is to drive down all penalty coefficients of an individual to zero—thus eliminating any reduction in the fitness of the corresponding candidate solution—and actually finding the solution of the unconstrained problem (Eiben and Smith 2003).

Eiben et al. (2000) proposed a scheme to prevent EAs from "cheating" when solving constraint satisfaction problems (CSPs). When solving CSPs by means of EAs, weights are associated with each constraint to add a penalty to the individual if that constraint is not satisfied. Changes in the weights along the run will cause the EA to put more pressure into the satisfaction of the corresponding constraint. Eiben et al. introduced a tournament selection that uses the maximum of each of the weights, across all competitors, as a way to eliminate cheating in the CSP case, without resorting to any feedback mechanism from the search process. Unfortunately, to the best of our knowledge, no strict self-adaptive technique has been applied so far to constrained optimization problems in $\mathbb{R}^n$.

### 1.5.2 Coevolving the Parameters

Coello (2000) introduced a co-evolutionary algorithm to adapt the penalty coefficients of a fitness function in a GA with two populations $P_1$ (size $M_1$) and $P_2$ (size $M_2$). The fitness function is written as

$$F(x) = f(x)k\,(\text{sum\_viol}(x) \times w_1 + \text{num\_viol}(x) \times w_2) \qquad (1.45)$$

where $w_1$ and $w_2$ are two (integer) penalty coefficients, and sum_viol$(x)$ and num_viol$(x)$ are, respectively, the sum of the violations and the number of constraints which are violated by the candidate solution $x$. The second of these populations, $P_2$, encodes the set of weight combinations ($w_1$ and $w_2$) that will be used to compute the fitness value of the candidate solutions in $P_1$ whereas $P_2$ contains the penalty coefficients that will be used in the fitness function evaluation. Benchmark problems from the literature, especially mechanical engineering optimization, are used in the numerical tests but only inequality constraints were considered in the experiments.

The co-evolutionary idea was also analyzed in He and Wang (2007) and He et al. (2008). In these works, the penalty factors are adapted by a co-evolutionary particle swarm optimization approach (CPSO). Two kinds of swarms are used in He and Wang (2007) and He et al. (2008): one population of multiple swarms is used to solve the search problem and other one is responsible to adapt the penalty factors. Each particle $j$ in the second population represents the penalty coefficients for a set of particles in the first one. The two populations evolve by a given $G_1$ and $G_2$ number of generations. The adopted fitness function is the one proposed by Richardson et al. (1989), where not only the amount of violation contributes to the quality of a given candidate solution but also the number of of violated constraints. According to He and Wang (2007) and He et al. (2008),

$$F_j(x) = f(x) + \text{sum\_viol}(x) \times w_{j,1} + \text{num\_viol}(x) \times w_{j,2},$$

where $f(x)$ is the objective function value, and $w_{j,1}$ and $w_{j,2}$ are the penalty coefficients from the particle $j$ in the second swarm population. The penalty factors $w_{j,1}$ and $w_{j,2}$ are evolved according to the following fitness:

$$G(j) = \begin{cases} \frac{\text{sum\_feas}}{\text{num\_feas}} - \text{num\_feas}, \text{ if there is at least one feasible solution in the subset} \\ \max(G_{\text{valid}}) + \frac{\sum_{i=1}^{pop} \text{sum\_viol}(x^i)}{\sum_{i=1}^{pop} \text{num\_viol}(x^i)} - \sum_{i=1}^{pop} \text{num\_viol}(x^i), \quad \text{otherwise,} \end{cases}$$

where sum_feas denotes the sum of objective function values of feasible solutions, num_feas is the number of feasible individuals, and max$(G_{\text{valid}})$ denotes the maximum

$G$ over all valid particles; the valid particles are those ones which operate over a subset of particles where there is at least one feasible solution.

### 1.5.3 Using Other Tools

It is interesting to note that, despite all the effort that has been devoted to the research of penalty techniques in the context of nature inspired metaheuristics in the last 20 years or so, the subject still draws the attention of the researchers, and new tools are being constantly introduced to this arena. Fuzzy logic and rough set theory are just two recent examples that will be mentioned in the following.

Wu etal. (2001) proposed a fuzzy penalty function strategy using information contained in individuals. The fitness function of an infeasible individual is

$$F(x) = f(x) + rG(x) \tag{1.46}$$

where $G(x)$ is the amount of constraint violation from inequality and equality constraints, and $r$ is the penalty coefficient.

$f$ and $G$ are taken as fuzzy variables with the corresponding linguistic values such as very large, large, small, very small, etc. The ranges for $f$ and $G$ are defined by $D_f = [f_{\min}, f_{\max}]$ and $D_G = [G_{\min}, G_{\max}]$. Those ranges must then be partitioned—which is a problem dependent, non-trivial task—and linguistic values are associated with each part. The sets $A$ and $B$ are introduced as fuzzy sets for $f$ and $G$, respectively, and $r^k, k = 1, \ldots, l$ is defined as a fuzzy singleton for $r$ which is inferred from appropriate membership functions and finally used in (1.46).

In their numerical experiments, three partitions were used for both $f$ and $G$ with triangle membership functions, and five points were used for the output. The rule base contained 9 rules in the form

$$\text{If } f \text{ is } A_i \text{ and } G \text{ is } B_j \text{ then } r = r^k.$$

Lin (2013) proposed perhaps the first constraint-handling approach which applies the information granulation of rough set theory to address the indiscernibility relation among penalty coefficients in constrained optimization. Adaptive penalty coefficients for each constraint $w_{tk}, k = 1, \ldots, m$ were defined in a way that a high penalty is assigned to the coefficient of the most difficult constraint. In addition, the coefficients are also depended on the current generation number $t$. Using the standard definition for the violation of the $j$th constraint ($v_j(x)$), the fitness function reads as

$$F(x) = f(x) + \sum_{j=1}^{m} w_{tk} v_j^2(x)$$

where $w_{tk} = (C \times t)^{\pi(k,t)}$ and $C$ is a "severity" factor. The exponent $\pi(k, t)$, initialized as $\pi(k, 0) = 2$ for all $k$, is defined as

$$\pi(k,t) = \begin{cases} \pi(k, t-1) \times \gamma_k, & \text{if } \mu_k = 1 \\ \pi(k, t-1) & \text{if } \mu_k = 0 \end{cases}$$

according to the discernible mask $\mu$ and the representative attribute value $\gamma_k$ of the superior class $X_{\text{good}}$ (see the paper for details). If the $k$th constraint is discernible (i.e., $\mu_k = 1$), the exponent $\pi(k, t)$ is adjusted by the representative attribute value ($\gamma_k$); otherwise, the exponent retains the same value as in the previous generation.

## 1.6 Discussion

### *1.6.1 User-Defined Parameters*

Some of the proposals considered do not require from the user the definition of penalty parameters, and can as such be considered "parameterless". This is very useful for the practitioner. However, it should be noted that essentially all proposals do embody some fixed values that are hidden from the user and, as a result, cannot be changed. Furthermore, all proposals involve design decisions which were made—with variable level of justification—and incorporated into the definition of the technique. It seems natural to assume that some of those could possibly be changed—a research opportunity—leading to improved results.

### *1.6.2 Comparative Performance*

In order to test the performance of a constraint handling technique, several test-problems have been used over the years. The most popular suite of continuous constrained optimization problems is that containing the 24 problems used for the competition held during the 2006 IEEE Congress on Evolutionary Computation which are described in Liang et al. (2006). Later, larger problems were considered in another competition, held during the 2010 edition of the same conference. The details can be found in Mallipeddi and Suganthan (2010).

It can be noticed that the claims concerning the performance of each proposal in the papers reviewed have been deliberately omitted. This is due to several factors. One of them is that a statistical study in order to assure a possible statistically significant superiority of the proposed technique over others from the literature is often missing. Another criticism is that often the claimed superiority of the proposed technique can only be observed after the fourth or fifth significant digit of the final results, with no consideration for the facts (i) that the original model itself may not have such accuracy, and (ii) that the compared solutions may be indistinguishable from the practical point of view.

Another major issue that makes it impossible to rigorously assess the relative performance of the adaptive penalty techniques (APTs) reviewed is that the final results depend not only on the penalty technique considered but also on the search engine (SE) adopted. The competing results often derive from incomparable arrangements such as APT-1 embedded in SE-1 (a genetic algorithm, for instance) *versus* APT-2 applied to SE-2 (an evolution strategy, for instance). The results using stochastic ranking (SR) within an evolution strategy (ES) (Runarsson and Yao 2000) were shown to outperform APM embedded in a binary-coded genetic algorithm (GA) (Lemonge and Barbosa 2004) when applied to a standard set of benchmark constrained optimization problems in $\mathbb{R}^n$. This seems to be due—at least in part—to the fact that the ES adopted performs better in this continuous domain than a standard GA. A proper empirical assessment of the constraint handling techniques considered (SR versus APM) should be performed by considering settings such as (SR+GA versus APM+GA) and (SR+ES versus APM+ES). An attempt to clarify this particular question is presented by Barbosa et al. (2010b). It is clear that there is a need for more studies of this type in order to better assess the relative merits of the proposals reviewed here.

The standard way of assessing the relative performance of a set $A$ of $n_a$ algorithms $a_i$, $i \in \{1, \ldots, n_a\}$, is to define a set $P$ of $n_p$ "representative" problems $p_j$, $j \in \{1, \ldots, n_p\}$, and then test all algorithms against all problems, measuring the performance $t_{p,a}$ of algorithm $a \in A$ when applied to problem $p \in P$.

In order to evaluate $t_{p,a}$ one can alternatively (i) define a meaningful goal (say, level of objective function value) and then measure the amount of resources (say, number of function evaluations) required by the algorithm to achieve that goal, or (ii) fix a given amount of resources to be allocated to each algorithm and then measure the goal attainment.

Considering that $t_{p,a}$ is the CPU time spent by algorithm $a$ to reach the stated goal in problem $p$ a performance ratio can be defined as

$$r_{p,a} = \frac{t_{p,a}}{\min\{t_{p,a} : a \in A\}}. \tag{1.47}$$

Although each $t_{p,a}$ or $r_{p,a}$ is worth considering by itself, one would like to be able to assess the performance of the algorithms in $A$ on a large set of problems $P$ in a user-friendly graphical form. This has been achieved by Dolan and Moré (2002) who introduced the so-called performance profiles, an analytical tool for the visualization and interpretation of the results of benchmark experiments. For more details and an application in the constrained optimization case, see Barbosa et al. (2010a).

One has also to consider that it is not an easy task to define a set $P$ which is "representative" of the domain of interest, as one would like $P$ (i) to span the target problem-space and, at the same time, (ii) to be as small as possible, in order to alleviate the computational burden of the experiments. Furthermore, it would also be interesting to assess the relative performance of the test-problems themselves with respect to the solvers. Are all test-problems relevant to the final result? Are some test-problems too easy (or too difficult) so that they do not have the ability to discriminate the solvers? Efforts in this direction, exploring the performance profile concept, were attempted in Barbosa et al. (2013).

### *1.6.3 Implementation Issues*

Although not always considered in the papers reviewed, the simplicity of the technique (both conceptually and from the implementation point of view) is relevant. It seems quite desirable that the proposed technique could be easily implemented as an additional module to any existing metaheuristic for unconstrained optimization with a minimum interference with the current code. In this respect, techniques resorting to coevolution would typically require another population, an additional set of parameters, and would lead to more interference and modifications to the original code.

### *1.6.4 Extensions*

It seems natural to expect that most of, if not all, the proposals reviewed here can be easily extended to the practically important case of constrained *multi-objective* optimization. Although papers presenting such extension have not been reviewed here, it seems that there is room, and indeed a need, to explore this case.

The same can perhaps be said of the relevant case of *mixed* (discrete and continuous) decision variables, as well as the more complex problem of constrained *multi-level* optimization.

## 1.7 Conclusion

This chapter presented a review of the main adaptive penalty techniques available for handling constraints within nature inspired metaheuristics in general and evolutionary techniques in particular. The main types of evidence taken from the search process in order to inform the decision-making process of continuously adapting the relevant parameters of the penalty technique have been identified.

As the different adaptive techniques have not been implemented on a single given search engine, the existing comparative studies, which are usually based on the final performance on a set of benchmark problems, are not very informative of the relative performance of each penalty technique, as the results are also affected by the different search engines adopted in each proposal. The need for better comparative studies investigating the relative performance of the different adaptive techniques when applied within a single search engine in larger and more representative sets of benchmark problems are also identified.

**Acknowledgments** The authors thank the reviewers for their comments, which helped improve the quality of the final version, and acknowledge the support from CNPq (grants 308317/2009-2, 310778/2013-1, 300192/2012-6 and 306815/2011-7) and FAPEMIG (grant TEC 528/11).

## References

Barbosa HJC, Lemonge ACC (2002) An adaptive penalty scheme in genetic algorithms for constrained optimization problems. In: Langdon WB, Cantú-Paz E, Mathias KE, Roy R, Davis D, Poli R, Balakrishnan K, Honavar V, Rudolph G, Wegener J, Bull L, Potter MA, Schultz AC, Miller JF, Burke EK (eds) Proceedings of the genetic and evolutionary computation conference (GECCO). Morgan Kaufmann, San Francisco

Barbosa HJC, Lemonge ACC (2003a) An adaptive penalty scheme for steady-state genetic algorithms. In: Cantú-Paz E, Foster JA, Deb K, Davis LD, Roy R, O'Reilly U-M, Beyer H-G, Standish R, Kendall G, Wilson S, Harman M, Wegener J, Dasgupta D, Potter MA, Schultz AC, Dowsland KA, Jonoska N, Miller J (eds) Genetic and evolutionary computation (GECCO). Lecture Notes in Computer Science. Springer, Berlin, pp 718–729

Barbosa HJC, Lemonge ACC (2003b) A new adaptive penalty scheme for genetic algorithms. Inf Sci 156:215–251

Barbosa HJC, Lemonge ACC (2008) An adaptive penalty method for genetic algorithms in constrained optimization problems. Front Evol Robot 34:9–34

Barbosa HJC, Bernardino HS, Barreto AMS (2010a) Using performance profiles to analyze the results of the 2006 CEC constrained optimization competition. In: 2010 IEEE congress on evolutionary computation (CEC), pp 1–8

Barbosa HJC, Lemonge ACC, Fonseca LG, Bernardino HS (2010b) Comparing two constraint handling techniques in a binary-coded genetic algorithm for optimization problems. In: Deb K, Bhattacharya A, Chakraborti N, Chakroborty P, Das S, Dutta J, Gupta SK, Jain A, Aggarwal V, Branke J, Louis SJ, Tan KC (eds) Simulated evolution and learning. Lecture Notes in Computer Science. Springer, Berlin, pp 125–134

Barbosa HJC, Bernardino HS, Barreto AMS (2013) Using performance profiles for the analysis and design of benchmark experiments. In: Di Gaspero L, Schaerf A, Stutzle T (eds) Advances in metaheuristics. Operations Research/computer Science Interfaces Series, vol 53. Springer, New York, pp 21–36

Bean J, Alouane A (1992) A Dual Genetic Algorithm For Bounded Integer Programs. Technical Report Tr 92-53, Department of Industrial and Operations Engineering, The University of Michigan

Beaser E, Schwartz JK, Bell CB, Solomon EI (2011) Hybrid genetic algorithm with an adaptive penalty function for fitting multimodal experimental data: application to exchange-coupled non-Kramers binuclear iron active sites. J Chem Inf Model 51(9):2164–2173

Coello CAC (2000) Use of a self-adaptive penalty approach for engineering optimization problems. Comput Ind 41(2):113–127

Coello CAC (2002) Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Comput Methods Appl Mech Eng 191 (11–12):1245–1287

Coit DW, Smith AE, Tate DM (1996) Adaptive penalty methods for genetic optimization of constrained combinatorial problems. INFORMS J Comput 8(2):173–182

Costa L, Santo IE, Oliveira P (2013) An adaptive constraint handling technique for evolutionary algorithms. Optimization 62(2):241–253

Courant R (1943) Variational methods for the solution of problems of equilibrium and vibrations. Bull Am Math Soc 49:1–23

Dolan E, Moré JJ (2002) Benchmarking optimization software with performance profiles. Math Program 91(2):201–213

Eiben AE, Smith JE (2003) Introduction to evolutionary computing. Springer, New York

Eiben AE, Jansen B, Michalewicz Z, Paechter B (2000) Solving CSPs using self-adaptive constraint weights: how to prevent EAs from cheating. In: Whitley, LD (ed) Proceedings of the genetic and evolutionary computation conference (GECCO). Morgan Kaufmann, San Francisco, pp 128–134

Farmani R, Wright J (2003) Self-adaptive fitness formulation for constrained optimization. IEEE Trans Evol Comput 7(5):445–455

Gan M, Peng H, Peng X, Chen X, Inoussa G (2010) An adaptive decision maker for constrained evolutionary optimization. Appl Math Comput 215(12):4172–4184

Gen M, Cheng R (1996) Optimal design of system reliability using interval programming and genetic algorithms. Comput Ind Eng, (In: Proceedings of the 19th international conference on computers and industrial engineering), vol 31(1–2), pp 237–240

Hamida H, Schoenauer M (2000) Adaptive techniques for evolutionary topological optimum design. In: Parmee I (ed) Proceedings of the international conference on adaptive computing in design and manufacture (ACDM). Springer, Devon, pp 123–136

Hamida S, Schoenauer M (2002) ASCHEA: new results using adaptive segregational constraint handling. In: Proceedings of the IEEE service center congress on evolutionary computation (CEC), vol 1. Piscataway, New Jersey, pp 884–889

Harrell LJ, Ranjithan SR (1999) Evaluation of alternative penalty function implementations in a watershed management design problem. In: Proceedings of the genetic and evolutionary computation conference (GECCO), vol 2. Morgan Kaufmann, pp 1551–1558

He Q, Wang L (2007) An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Eng Appl Artif Intell 20(1):89–99

He Q, Wang L, zhuo Huang F (2008) Nonlinear constrained optimization by enhanced co-evolutionary PSO. In: IEEE congress on evolutionary computation, CEC 2008. (IEEE World Congress on Computational Intelligence), pp 83–89

Hughes T (1987) The finite element method: linear static and dynamic finite element analysis. Prentice Hall Inc, New Jersey

Koziel S, Michalewicz Z (1998) A decoder-based evolutionary algorithm for constrained parameter optimization problems. In: Eiben A, Bäck T, Schoenauer M, Schwefel H-P (eds) Parallel problem solving from nature (PPSN). LNCS, vol 1498. Springer, Berlin, pp 231–240

Krempser E, Bernardino H, Barbosa H, Lemonge A (2012) Differential evolution assisted by surrogate models for structural optimization problems. In: Proceedings of the international conference on computational structures technology (CST). Civil-Comp Press, p 49

Lemonge ACC, Barbosa HJC (2004) An adaptive penalty scheme for genetic algorithms in structural optimization. Int J Numer Methods Eng 59(5):703–736

Lemonge ACC, Barbosa HJC, Bernardino HS (2012) A family of adaptive penalty schemes for steady-state genetic algorithms. In: 2012 IEEE congress on evolutionary computation (CEC). IEEE, pp 1–8

Liang J, Runarsson TP, Mezura-Montes E, Clerc M, Suganthan P, Coello CC, Deb K (2006) Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization. Technical report, Nanyang Technological University, Singapore

Lin C-H (2013) A rough penalty genetic algorithm for constrained optimization. Inf Sci 241: 119–137

Lin C-Y, Wu W-H (2004) Self-organizing adaptive penalty strategy in constrained genetic search. Struct Multidiscip Optim 26(6):417–428

Luenberger DG, Ye Y (2008) Linear and nonlinear programming. Springer, New York

Mallipeddi R, Suganthan PN (2010) Problem definitions and evaluation criteria for the CEC 2010 competition on constrained real-parameter optimization. Technical report, Nanyang Technological University, Singapore

Mezura-Montes E, Coello CAC (2011) Constraint-handling in nature-inspired numerical optimization: past, present and future. Swarm Evol Comput 1(4):173–194

Michalewicz Z (1995) A survey of constraint handling techniques in evolutionary computation methods. In: Proceedings of the 4th annual conference on evolutionary programming. MIT Press, pp 135–155

Michalewicz Z, Schoenauer M (1996) Evolutionary algorithms for constrained parameter optimization problems. Evol Comput 4(1):1–32

Montemurro M, Vincenti A, Vannucci P (2013) The automatic dynamic penalisation method (ADP) for handling constraints with genetic algorithms. Comput Methods Appl Mech Eng 256:70–87

Nanakorn P, Meesomklin K (2001) An adaptive penalty function in genetic algorithms for structural design optimization. Comput Struct 79(29–30):2527–2539

Puzzi S, Carpinteri A (2008) A double-multiplicative dynamic penalty approach for constrained evolutionary optimization. Struct Multidiscip Optim 35(5):431–445

Rasheed K (1998) An adaptive penalty approach for constrained genetic-algorithm optimization. In: Koza J, Banzhaf W, Chellapilla K, Deb K, Dorigo M, Fogel D, Garzon M, Goldberg D, Iba H, Riolo R (eds) Proceedings of the third annual genetic programming conference. Morgan Kaufmann, San Francisco, pp 584–590

Richardson JT, Palmer MR, Liepins GE, Hilliard M (1989) Some guidelines for genetic algorithms with penalty functions. In: Proceedings of the international conference on genetic algorithms. Morgan Kaufmann, San Francisco, pp 191–197

Rocha AMAC, Fernandes EMDGP (2009) Self-adaptive penalties in the electromagnetism-like algorithm for constrained global optimization problems. In: Proceedings of the 8th world congress on structural and multidisciplinary optimization, Lisbon, Portugal

Runarsson T, Yao X (2000) Stochastic ranking for constrained evolutionary optimization. IEEE Trans Evol Comput 4(3):284–294

Salcedo-Sanz S (2009) A survey of repair methods used as constraint handling techniques in evolutionary algorithms. Comput Sci Rev 3(3):175–192

Schoenauer M, Michalewicz Z (1996) Evolutionary computation at the edge of feasibility. In: Proceedings of parallel problem solving from nature (PPSN). LNCS, Springer, pp 245–254

Tessema B, Yen GG (2006) A self adaptive penalty function based algorithm for constrained optimization. In: IEEE congress on evolutionary computation, CEC 2006. IEEE, pp 246–253

Tessema B, Yen G (2009) An adaptive penalty formulation for constrained evolutionary optimization. IEEE Trans Syst, Man Cybern, Part A: Syst Hum 39(3):565–578

Vincenti A, Ahmadian MR, Vannucci P (2010) BIANCA: a genetic algorithm to solve hard combinatorial optimisation problems in engineering. J Glob Optim 48(3):399–421

Wang Y, Cai Z, Zhou Y, Fan Z (2009) Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique. Struct Multidiscip Optim 37(4):395–413

Wu B, Yu X, Liu L (2001) Fuzzy penalty function approach for constrained function optimization with evolutionary algorithms. In: Proceedings of the 8th international conference on neural information processing. Citeseer, pp 299–304

Wu W-H, Lin C-Y (2004) The second generation of self-organizing adaptive penalty strategy for constrained genetic search. Adv Eng Softw 35(12):815–825

Yokota T, Gen M, Ida K, Taguchi T (1995) Optimal design of system reliability by an improved genetic algorithm. Trans Inst Electron Inf Comput Eng J78-A(6):702–709 (in Japanese)