Struct Multidisc Optim (2008) 35:431–445
DOI 10.1007/s00158-007-0143-1

RESEARCH PAPER

A double-multiplicative dynamic penalty approach for constrained evolutionary optimization

Simone Puzzi · Alberto Carpinteri

Received: 13 July 2006 / Revised: 5 February 2007 / Accepted: 25 March 2007 / Published online: 2 June 2007

Abstract The present paper proposes a double-multiplicative penalty strategy for constrained optimization by means of genetic algorithms (GAs). The aim of this research is to provide a simple and efficient way of handling constrained optimization problems in the GA framework without the need for tuning the values of penalty factors for any given optimization problem. After a short review on the most popular and effective exterior penalty formulations, the proposed penalty strategy is presented and tested on five different benchmark problems. The obtained results are compared with the best solutions provided in the literature, showing the effectiveness of the proposed approach.

Keywords Genetic algorithms · Exterior penalty function · Constraint handling · Constrained optimization

1 Introduction

Over the last decades, interest in search algorithms that rely on analogies to natural processes, for instance simulated annealing and classifier systems, has been continuously growing. Among these methods, one important class is that based on some evolution and hereditary mimicking. The most popular and widespread techniques in this group include genetic algorithms (GAs; Holland 1975; Goldberg 1989), evolutionary programming (Fogel 1966), evolution strategies (Rechenberg 1973; Schwefel 1981), and biological evolution methods (Arutyunyan and Drozdov 1988; Azegami 1990; Mattheck and Burkhardt 1990). The present trend is to decrease the differences among these paradigms and refer, in generic terms, to evolutionary algorithms (EAs) or evolution programs (EPs) when talking about any of them (Michalewicz 1996).

S. Puzzi (✉) · A. Carpinteri
Department of Structural Engineering and Geotechnics,
Politecnico di Torino,
Corso Duca degli Abruzzi, 24,
10129 Turin, Italy
e-mail: simone.puzzi@polito.it

A. Carpinteri
e-mail: Alberto.Carpinteri@polito.it

The reason behind the success of these approaches is that traditional nonlinear programming methods are local search techniques and are, thus, prone to converge toward local optima. This is a serious drawback in many practical structural optimization problems where the design space may be nonconvex or even disjoint (Hajela 1990). In particular, GAs are very popular due to their ability to naturally handle design variables of discrete type. GAs have been successfully applied to optimization problems in different fields (Fogel 1995; Gen and Cheng 1997). Their advantages are related to their speed, compared to enumerative techniques, and to the lack of the drawbacks concerning gradient-based search algorithms in nonconvex and multi-modal spaces.

GAs are a particular class of EAs that uses techniques inspired by evolutionary biology, such as inheritance, mutation, selection, and reproduction (in this context, also called crossover or recombination). Although the analogy with biology is far from strict, GAs mimic and exploit the genetic dynamics that underlie natural evolution to search for optimal solutions of general optimization problems. To explain this, let us consider an optimization problem in which a generic function $F(\boldsymbol{x})$, with $\boldsymbol{x} \in \mathbb{R}^n$ (but could also be $\boldsymbol{x} \in \mathbb{N}^n$, $\boldsymbol{x} \in \mathbb{Z}^n$, or $\boldsymbol{x} \in \mathbb{C}^n$), has to be maximized. The variable $\boldsymbol{x}$ may be subject to any kind of constraint; thus, the search space may be bounded. To deal with this problem, a GA (as any other EP) must have the following components:

- A genetic representation (encoding) for potential solutions, $\boldsymbol{x}$, to the problem
- A criterion for creating an initial population of potential solutions (both random and deterministic criteria are possible)
- A fitness function, $F(\boldsymbol{x})$, that plays the role of the environment; by rating the candidate solutions in terms of their fitness, higher chances of survival can be assigned to better solutions so that their features are more likely to be transmitted to the next generation
- Evolutionary operators that alter the population (usually crossover and mutation)
- Parameters for the algorithm itself (probabilities of applying the genetic operators, population size, etc.)

The random aspects of the GA result in the search algorithm exploring the whole solution space without focusing on small portions of it. Obviously, the number of individuals in the population, the number of generations, and the parameters that control the "genetic" operators, together with the implementation of the procedures, greatly influence the quality of the obtained results.

In GA-based optimization, the main problem usually concerns the constraint treatment; GAs, as well as other EPs, do not directly take constraints into account. That is, the regular search operators, mutation, and recombination are "blind" to constraints. Consequently, although the parents satisfy some constraints, they might lead to offspring that violate them. Technically, this means that the EAs perform an unconstrained search, as quoted by Coello (2002). This observation suggests that the presence of constraints in a problem makes GAs intrinsically unsuited to handle this problem.

Therefore, it is necessary to find ways of incorporating the constraints (that exist in almost any real-world application) into the fitness function. The most common way of incorporating constraints into a GA has been penalty functions, but several approaches have been proposed:

1. Penalty functions
2. Special representations and operators
3. Repair algorithms
4. Separation of objectives and constraints

The penalty function approach is the most diffused, and attention will be focused on it in the next section.

The use of special representation and operators is aimed at simplifying the shape of the search space in certain problems (usually the most difficult and which are characterized by very small feasibility domains) for which the classical representation scheme (e.g., the binary representation used in traditional GAs) might not work satisfactorily. The drawback of this technique is that it is necessary to design special genetic operators that work in a similar way to the traditional operators used with a binary representation. The main drawback of the use of special representations and operators is that their generalization to other problems, even similar to the one tackled, might be very difficult.

Repair algorithms have the same drawback; in several optimization problems, for instance, the travelling salesman problem or the knapsack problem, it is relatively easy to "repair" an infeasible individual (i.e., to make an infeasible individual feasible). Such a repaired version can be used either for evaluation only, or it can also replace (with some probability) the original individual in the population. Obviously, this approach is problem dependent, as a specific repair algorithm has to be designed for each particular problem. In addition, the "repair" operation might, in some cases, be even harder than the optimization itself and harm the evolutionary process by introducing a bias into the search (Smith and Coit 1997). Nevertheless, repair algorithms have been successfully used and are always a good choice when the "repair" may be done easily (Coello 2002).

Probably, the most promising trend is the last approach, which considers constraints and objectives separately. The use of multi-objective optimization techniques seems to be particularly encouraging in terms of obtained results. Nevertheless, Michalewicz and Schoenauer (1996) noted that, although many sophisticated penalty-based methods have been reported as promising tools for constrained evolutive searches, the traditional static penalty function method is very often used, as it is relatively reliable and robust due to the simplicity of the penalty function. A further reason for the success of this approach is the fact that this technique can be equally applied, without any modification, to other global optimization algorithms that do not use derivatives, such as evolution strategies (Rechenberg 1973; Schwefel 1981), simulated annealing (Kirkpatrick et al. 1983), particle swarm optimization (Eberhart and Kennedy 1995), or Direct (Jones et al. 1993), just to mention a few.

In this paper, we propose a new double-multiplicative penalty strategy to deal with constrained optimization. As will be shown, coupling this approach to a real-coded GA provides very good results in several benchmark tests.

2 Traditional penalty function strategies

The use of penalty functions is the most common way of treating constrained optimization problems. The idea behind this method is to transform a constrained-optimization problem into an unconstrained one by subtracting (or adding) a certain value from (or to) the objective function based on the amount of constraint violation present in a certain solution.

In classical optimization, two main kinds of penalty functions have been considered: exterior and interior, although hybrid approaches have also been proposed (Haftka and Starnes 1976). In the case of interior penalty functions, the penalty term is chosen so that its value is small at points far from the constraint boundaries and tends to infinity as the boundaries are approached. In this way, constraint boundaries act as barriers during the optimization process. The main drawback of interior penalties is the requirement for initial feasible solutions; it is a serious drawback, as the problem of finding a feasible solution is very hard in many applications. In the case of exterior penalty functions, on the other hand, an initial feasible solution is not required, as the penalty term is zero for feasible solutions and increases as the constraint violations increase.

Exterior penalty approaches can be further divided into different categories. A first distinction may be made between static and dynamic penalty functions, where the first class always provides the same value of the penalty term for a given solution $\boldsymbol{x}$ through the whole selection process, whereas the second class provides an increase in the penalty term, to increase the selective pressure during the evolution progress.

Several researchers have also investigated annealing and adaptive penalty factors in which the magnitude of the penalty term is dynamically modified on the basis of different criteria (Coello 2002); based on the fitness of the best solution found, the variability of the fitness in the population, the feasibility ratio, i.e., the ratio between the number of feasible and infeasible solutions, and so on. Coello (2000b) proposed a co-evolutionary approach in which the penalty coefficients themselves undergo an evolution process. In Coello's approach, one population is used to evolve solutions, and the second one to evolve the penalty factors, which are defined to separate the information concerning the total feasibility and the corresponding amounts of violation.

The definition of a good penalty function is very difficult and is problem dependent. Ideally, the penalty should be kept as low as possible, just above the limit, below which infeasible solutions are optimal, according to the minimum penalty rule (Davis 1987). A too-high penalty may push the GA inside the feasible region very quickly, and this could be a serious drawback in the case where the optimum lies at the boundary of the feasible solution, or in the case of multi-modal, disjoint search spaces. In other words, a large penalty discourages the exploration of the infeasible region from the beginning of the search process, resulting in premature convergence to local optima. On the other hand, if the penalty is too low, a great deal of the search time is spent exploring the infeasible region because the penalty becomes too small compared to the objective function (Smith and Coit 1997). Thus, finding a suitable value for the penalty is a difficult task, which is very often problem dependent.

Many penalty function methods require the users to define appropriate penalty parameter values. A robust penalty function needs to be problem independent; a recent trend in this direction is the development of parameter-free self-adaptive penalty function methods. The fundamentals of several popular exterior penalty methods, both static and dynamic, are summarized in the following section to highlight their characteristics before introducing the new double-multiplicative dynamic penalty.

Let us consider the following constrained optimization problem. Minimize

$$F(\boldsymbol{x}), \tag{1}$$

subject to

$$g_j(\boldsymbol{x}) = \overline{g_j}(\boldsymbol{x}) - b_j \leq 0, \quad j = 1, \ldots, m, \tag{2}$$

$$h_k(\boldsymbol{x}) = \overline{h_k}(\boldsymbol{x}) - c_k = 0, \quad k = 1, \ldots, p, \tag{3}$$

$$x_i^L \leq x_i \leq x_i^U, \quad i = 1, \ldots, n. \tag{4}$$

The goal of the optimization is to find the optimum design, $\boldsymbol{x}^*$, that minimizes the objective function values while satisfying all m inequality constraints and p equality constraints. Two sets of functions, $g_j(\boldsymbol{x})$ and $h_k(\boldsymbol{x})$ represent the inequality and equality constraint functions, respectively. These functions could be either linear or nonlinear.

A feasible design must satisfy all constraints, i.e., all $g_j(\boldsymbol{x})$ function values must be less than or equal to 0, and all $h_k(\boldsymbol{x})$ function values must equal 0. The exterior penalty function methods convert constraint violations into a penalty function, $P(\boldsymbol{x})$, and then add the penalty function to the original objective function to form a pseudo-objective function, $\phi(\boldsymbol{x})$. The constrained optimization problem defined in (1) to (4) is, thus, transformed into an unconstrained optimization problem. Minimize

$$\phi(\boldsymbol{x}) = F(\boldsymbol{x}) + P(\boldsymbol{x}) \tag{5}$$

where $P(\boldsymbol{x})$ is the penalty function that depends on the amount of constraint violation: $g_j(\boldsymbol{x})$, j=1,..., m and $h_k(\boldsymbol{x})$, k=1,..., p. When P is only a function of $\boldsymbol{x}$ and does not depend on the current generation number q, it is said to be static, while it is said to be dynamic if the opposite occurs and q concurs to define its value.

2.1 Lin and Hajela (1992)

In the approach by Lin and Hajela (1992), the function $P(\boldsymbol{x}, q)$ is computed according to the following equations:

$$\widehat{P}(\boldsymbol{x},q) = r(q)\left(\sum_{j=1}^{m}\widehat{g}_j(\boldsymbol{x})+\sum_{k=1}^{p}|h_k(\boldsymbol{x})|\right), \tag{6}$$

$$P(\boldsymbol{x},q) = \begin{cases} \widehat{P}(\boldsymbol{x},q) & \text{if } \widehat{P}(\boldsymbol{x},q) \le L \\ L + 0.2\left(\widehat{P}(\boldsymbol{x},q) - L\right) & \text{otherwise} \end{cases} \tag{7}$$

with $\widehat{g}_j(\boldsymbol{x}) = g_j(\boldsymbol{x})$ if $g_j(\boldsymbol{x}) \ge 0$, and $\widehat{g}_j(\boldsymbol{x}) = 0$ otherwise. Notice the presence of the variable penalty factor $r(q)$, which starts with a small value at the first generation and is then increased by a fixed amount at any given number of generations, making the approach dynamic. Thus, this method requires three parameters to be defined: the initial value of the penalty parameter, its change rate, and the limiter L. The deflection function defined in (7) through the limiter L is introduced to prevent very large constraint violations from degrading the reproduction plan for a given generation, as explained in the original paper.

2.2 Homaifar et al. (1994)

In the approach by Homaifar et al. (1994) in which only inequality constraints are considered, the penalty function is defined as follows:

$$P(\boldsymbol{x}) = \begin{cases} 0 & \text{if } \boldsymbol{x} \text{ is feasible} \\ \sum_{j=1}^{m} r_j g_j^2 & \text{otherwise} \end{cases} \tag{8}$$

A user-defined parameter r is also present in this case; the authors create several violation levels for each constraint and, depending on the level of violation, r_j varies accordingly. Obviously, the optimal definition of r_j is very difficult and problem dependent; it has been shown that inappropriate values for penalty factor values may result in an inefficient search process, or even in the inability to find feasible solutions (Michalewicz and Schoenauer 1996).

2.3 Joines and Houck (1994)

The Joines and Houck (1994) method is of the dynamic penalty type, with the penalty function $P(\boldsymbol{x}, q)$ defined as follows:

$$P(\boldsymbol{x},q) = r_q^{\alpha}\sum_{j=1}^{m+p} d_j^{\beta}(\boldsymbol{x}) \tag{9}$$

$$d_j(\boldsymbol{x}) = \begin{cases} 0 & \text{if } \boldsymbol{x} \text{ is feasible} \\ |g_j(\boldsymbol{x})| & 1 \le j \le m \\ |h_{j-m}(\boldsymbol{x})| & m+1 \le j \le m+p \end{cases} \tag{10}$$

$$r_q = C\,q, \tag{11}$$

where C, α, and β are user-defined parameters that are necessary to adjust the penalty scales; q is the generation number. This penalty strategy is quite close to Lin and Hajela's penalty approach, in that only one penalty parameter is used for the total constraint violation, but, in this case, the limiter L is removed and two coefficients, α and β, are added. As shown in (11), the factor r_q is proportional to the generation number q. Unfortunately, the setting of appropriate values of the three penalty parameters is quite problem dependent, particularly for the factor C in (11): if too high, it almost gives infeasible solutions a "death penalty". As quoted by Lin and Wu (2004), the exponentially increasing penalty factor in this algorithm causes fast elimination of infeasible designs due to large penalties. As a consequence, this algorithm is prone to premature convergence to non-optimal solutions.

2.4 Yokota et al. (1995)

In the Yokota et al. (1995) approach, the novelty is the form of the pseudo-objective function that is rewritten in a multiplicative form instead of in the additive form of (5):

$$\phi(\boldsymbol{x}) = F(\boldsymbol{x})P(\boldsymbol{x}). \tag{12}$$

In this case, the penalty amount for a given constraint violation depends not only on the degree of violation, but also on the value of the objective function. This implicitly produces a beneficial ranking of infeasible solutions in which those solutions characterized by a lower (and therefore better) value of the objective function are penalized less (if the degree of constraint violation is equal) and, therefore, favored in the selection procedure. In this case (same amount of constraint violation), the ranking provided by the additive and the multiplicative method is the same, but the difference in the fitness function (including the penalty term) between the two solutions is higher with the multiplicative approach. The penalty function proposed by these authors could be written as follows:

$$P(\boldsymbol{x}) = 1 + \frac{1}{m} \sum_{j=1}^{m} \left[\frac{\widehat{g}_j(\boldsymbol{x})}{b_j} \right]^{\beta} \tag{13}$$

with the parameter β used to adjust the severity of the penalty. $\widehat{g}_j(\boldsymbol{x})$ is defined as in the case of the Lin and Hajela (1992) strategy: $\widehat{g}_j(\boldsymbol{x}) = g_j(\boldsymbol{x})$ if $g_j(\boldsymbol{x}) \geq 0$, and $\widehat{g}_j(\boldsymbol{x}) = 0$ otherwise. The introduction of the term $1/b_j$ has the effect of normalizing each constraint in a very simple and direct way, ensuring that the amount of constraint function value for different constraints can be compared and become suitable for being directly summed up into the penalty function. In this approach, the penalty is static, as the same constraint violation amount always causes the same amount of penalty regardless of the stage of the evolution. This is the main drawback of the approach because the pressure to push the location of the maximum pseudo-objective function value back to the true constrained maximum is not increased during the search progress. Eventually, it should be noted that Yokota et al. (1995) formulated the optimization problem in terms of maximization, and therefore, the sign in (13) was originally a minus.

2.5 Le Riche and Haftka (1995)

In the approach by Le Riche and Haftka (1995) and Le Riche et al. (1995), the penalty function is additive [see (5)] and has the following structure, according to the standard notation used in Gürdal et al. (1999):

$$P(\boldsymbol{x}) = p\, g_{\max} + \delta. \tag{14}$$

where $g_{\max} = \max(g_j)$ for $j = 1,\ldots, m$ is the largest constraint violation. Both p and δ are user-defined parameters that penalize infeasible designs, but differ substantially. Parameter p penalizes solutions by considering the constraint violation amount, although in a simplified way, as only the maximum violation plays a role. Parameter δ, on the contrary, is a small fixed penalty that applies even for very small constraint violations. This small value can offer some benefit, as it can substantially reduce the need for large penalty multipliers to force satisfaction of inequality constraints when violations are not severe. Adding this term might be worthwhile particularly when the penalty function is intended for algorithms that do not require continuity of the objective functions, as in the case of GAs. However, as in the case of any user-defined parameter, the choice of the small-fixed amount might be problem dependent.

2.6 Gen and Cheng (1996)

Gen and Cheng (1996) modified the approach by Yokota et al. (1995) to make it dynamic. This was not done by using the current generation number, but by means of the following variable penalty function formulation:

$$P(\boldsymbol{x}) = 1 + \frac{1}{m} \sum_{j=1}^{m} \left[\frac{\widehat{g}_j(\boldsymbol{x})}{\max\left\{\varepsilon, \widehat{g}_{j,\max}(\boldsymbol{x})\right\}} \right]^{\beta}. \tag{15}$$

This strategy uses a different denominator to normalize each constraint. At the beginning of the genetic search, the largest constraint violation in the population is usually large so that the penalty due to a given amount of constraint violation is small. As the search progresses, the largest constraint violation in the population usually decreases, which subsequently increases the penalty for the same amount of constraint violation. The term $\boldsymbol{\varepsilon}$ is introduced to prevent division by zero if no solution in the population violates the jth constraint.

2.7 Lin and Wu (2004)

To overcome the problems of the aforementioned penalty-based strategies, Lin and Wu (2004) proposed a self-organizing adaptive penalty strategy (called SOAPS), with the aim at providing a parameter-free method with self-adaptivity of the selection pressure, independent penalty adjustability for each constraint, and automatic normalization of the constraints. The main idea behind their approach is that the constrained optimal design is almost always located at the boundary between feasible and infeasible domains. This penalty approach has, thus, been developed in an appropriate way to maintain equal numbers of designs on each side of the constraint boundary during the evolutive process so that the chance of locating their offspring around the boundary is maximum. The penalty approach is additive in this case [see (5)], and the penalty function is defined as follows:

$$P(\boldsymbol{x}, q) = \frac{q + 100}{100} \frac{1}{m} \sum_{j=1}^{m} r_j^q\, \widehat{g}_j(\boldsymbol{x}), \tag{16}$$

where q is the generation number, r_j^q the penalty parameter relative to the jth constraint at generation q, and $\widehat{g}_j(\boldsymbol{x})$ the constraint violation for the jth constraint. The dynamic penalty parameters r_j^q are computed iteratively:

$$r_j^q = r_j^{q-1} \left[1 - \frac{f_j^q - 0.5}{5} \right], \qquad q \geq 1, \tag{17}$$

where f_j^q is the feasibility ratio for the jth constraint at generation q. The initial values of the r_j^q parameters are computed as follows:

$$r_j^0 = \frac{QR_{obj}^1}{QR_{conj}^1}. \tag{18}$$

QR_{obj}^1 is the interquartile range (IQR) of the objective function values of the initial population and QR_{conj}^1 the IQR of the jth constraint function values of the initial population. The definition of the parameters r_j^0 in (17), together with their evolution [see (16)], is very effective in providing an automatic scaling of the constraint violations $\widehat{g}_j(\boldsymbol{x})$, thanks to the IQRs, but requires the evaluation of the constraint violation for all solutions in the population at each generation.

Basically, the dynamic increase of the selective pressure is given by the term $(q+100)/100$, which is individually corrected for each constraint by the term in square brackets within (16). In virtue of its characteristics, this approach has been shown to perform better than others in several benchmark tests. It is worthwhile noting that Wu and Lin (2004) proposed an improved version of this penalty, resulting in a better performance in problems with equality constraints.

3 The double-multiplicative dynamic penalty strategy

The penalty approach we propose takes some ideas from the previously reviewed approaches and strategies. The main goal of the proposed approach is to define a problem-independent penalty function, which should be parameter free and provide automatic constraint normalization. Furthermore, we are looking for a penalty strategy that could be easily implemented in a steady-state GA without the need to refer to the whole population to be computed (as in the case of penalty functions based on IQRs or feasibility ratios).

The chosen form of the penalty function is multiplicative [see (12)]; this choice is motivated by the fact that it implicitly provides a stronger ranking of infeasible solutions by considering not only their constraint violation degrees, but also their objective function values in the penalty computation. To easily achieve the normalization of the constraints, we chose the same normalization term as Yokota et al. (1995): $1/b_j$.

Then, to define a dynamic penalty, we make use of the generation-dependent term used in the SOAPS [see (16)]. This simple procedure allows the penalty function to be computed without the need to refer to the whole population, as in the case of SOAPS, where the IQRs come into play. We have instead retained the prefactor $(q+Q)/Q$ from the SOAPS strategy, with Q as the total number of generations, which makes the penalty dynamic without introducing any user-defined parameter.

Before proceeding, it can be observed that the positive effect of the multiplicative approach could be exploited not only for the evaluation function $\phi(\boldsymbol{x})$, but also for a better definition of the penalty function $P(\boldsymbol{x})$. In the Yokota et al. (1995) and Gen and Cheng (1996) strategies, the constraint violations are added and then multiplied by the term $1/m$ (m is the total number of constraints). In other words, the multiplicative penalty term is given by 1 plus a sort of "mean" constraint violation (a true mean in the Yokota et al. approach, if β=1). This could be a drawback in the case of optimization problems with a high number of constraints and only a low number of constraint violations, as often occurs in structural optimization, where only a few constraints may be active in the optimum solution: such a case would produce very mild penalties due to the large value of m, which could prevent the solution from moving toward the feasible zone of the search space. If we transform the penalty term $P(\boldsymbol{x})$ by multiplying terms of the form "1 plus the constraint violation," instead of adding the bare constraint violations, we can observe that even a single constraint violation may lead to an adequate penalty, depending on the degree of violation amount. Finally, we considered the inclusion of a small fixed penalty for infeasibility as in the case of the Le Riche and Haftka (1995) approach, but decided not to use it to keep our penalty function parameter free.

So far, the penalty approach can be rewritten in mathematical form as follows:

$$P(\boldsymbol{x}, q) = \prod_{j=1}^{m} \left[1 + \frac{\widehat{g}_j(\boldsymbol{x})}{b_j} \left(\frac{q+Q}{Q} \right) \right], \tag{19}$$

where Q is the total number of generations and $\widehat{g}_j(\boldsymbol{x})$ is defined as in the case of the Lin and Hajela (1992) strategy: $\widehat{g}_j(\boldsymbol{x}) = g_j(\boldsymbol{x})$ if $g_j(\boldsymbol{x})\geq 0$, and $\widehat{g}_j(\boldsymbol{x}) = 0$ otherwise. Notice that no parameter has to be defined by the user, as in the self-adaptive approaches. This penalty term multiplies the objective function $F(\boldsymbol{x})$, see (12).

To include also equality constraints, the penalty strategy of (19) can be rewritten as follows:

$$P(\boldsymbol{x}, q) = \prod_{j=1}^{m} \left[1 + \frac{\widehat{g}_j(\boldsymbol{x})}{b_j} \left(\frac{q+Q}{Q} \right) \right] \prod_{k=1}^{p} \left[1 + \frac{|h_k(\boldsymbol{x})|}{c_k} \left(\frac{q+Q}{Q} \right) \right]. \tag{20}$$

It should be noted, however, that for GAs and other gradient-less algorithms, equality constraints are much more of a challenge than inequality constraints, as the algorithms do not have any information that allows designs to remain feasible. Thus, it is likely that the proposed formulation might work better in the case with only inequality constraints.

4 Assessment of the proposed dynamic penalty function

To verify the effectiveness of the proposed approach, we tested the penalty strategy of (19) on five very well-known problems that have already been used as benchmarks for comparing different GA techniques. Four problems deal with inequality constraints; these problems are a ten-variable function problem (Hock and Schittkowski 1981), a five-variable function problem (Himmelblau 1972), the welded beam problem (Rao 1996), and the ten-bar cantilever truss problem (Venkayya 1971). The fifth problem, a six-variable function with three inequality and three equality constraints (Floudas and Pardalos 1990), deals with the more challenging case of equality constraints.

4.1 Algorithm details

A real-valued GA, to some extent similar to the Genocop I system proposed by Michalewicz (1996), has been implemented. The available unary operators are the same: uniform, non-uniform, and boundary mutation; the binary operators are a little different, and there are three: the arithmetical crossover (Michalewicz and Janikow 1991; Wright 1991), the heuristic crossover (Wright 1991), and the classical multi-point crossover (Eshelman et al. 1989) with two crossing points that is performed both on the original real-valued vector representation and on the binary representation (with conversion from real to binary coding after mate selection and re-conversion to the real representation just after crossover).

In short, in the arithmetical crossover, two individuals, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, are chosen; then, they are combined to produce a single offspring $\boldsymbol{y}=a\boldsymbol{x}_1+(1-a)\boldsymbol{x}_2$, with $\alpha\in[0,\ 1]$ as a random value. The heuristic crossover is a uncommon crossover operator, as it makes use of the objective function values; it generates a single offspring $\boldsymbol{y}$ from the two parents $\boldsymbol{x}_1$and $\boldsymbol{x}_2$, according to the following rule: $\boldsymbol{y}=a(\boldsymbol{x}_2-\boldsymbol{x}_1)+\boldsymbol{x}_2$ with a as a random number between 0 and 1, and the parent $\boldsymbol{x}_2$ is no worse than $\boldsymbol{x}_1$, i.e., $f(x_2)\leq f(x_1)$ for minimization problems. It should be noted that this crossover does not guarantee that two solutions satisfying (4) generate a solution that satisfies it; for this reason, as proposed by Michalewicz (1996), when the child violates (4), the heuristic crossover is performed again with another value of the random parameter a, and this procedure is repeated until the child satisfies these bounds. With respect to the constraints of (2) and (3), instead, none of the three crossover operators guarantees that two feasible parents generate a feasible child. However, because constraints given by (2) and (3) are taken into account by the penalty formulation, no repair strategy has been implemented in our algorithm.

In each test case, the probabilities of crossover and mutation are set to 0.3 and 0.05, respectively. The crossover type is selected randomly among the available operators according to the following probability values: 0.5 for binary crossover, 0.25 for arithmetic, and 0.25 for heuristic crossover. In the case, where the variables are less than 10, the two-point crossover is always performed on the binary representation, while in the case of more variables, the cross-over is performed on both representation types, with equal probabilities.

A similar procedure is followed for the choice of the mutation type. Three mutation types have been implemented: the uniform mutation, the non-uniform mutation, and the boundary mutation (Michalewicz 1996). The uniform mutation requires a single parent $\boldsymbol{x}$ and generates a single offspring $\boldsymbol{y}$. The operator selects a random component $k\in(1,\ldots,\ n)$ of the vector $\boldsymbol{x}=(x_1,\ldots,\ x_k,\ldots,\ x_n)$ and produces $\boldsymbol{y}=(x_1,\ldots,\ y_k,\ldots,\ x_n)$ where y_k is a random value (with uniform probability distribution) in the range $\left[x_k^L, x_k^U\right]$ [see (4)]. The non-uniform mutation is a little different in that y_k is defined as follows:

$$y_k = \begin{cases} x_k + \Delta\left(q, x_k^U - x_k\right) \text{if } a \text{ is } 0 \\ x_k - \Delta\left(q, x_k - x_k^L\right) \text{if } a \text{ is } 1 \end{cases}. \tag{21}$$

where a is a random digit. The parameter q is, as usual, the generation number, and the function Δ is defined as:

$$\Delta(q,y) = y\,r\left(1-\frac{q}{Q}\right). \tag{22}$$

with Q as the total generation number and r a random value in the range [0, 1]. This operator returns a value in the range [0, y] so that the probability of $\Delta(q,\ y)$ being close to 0 increases as the generation number approaches Q. This property causes the operator to initially uniformly search the space and then very locally toward the end of the genetic search. The boundary mutation generates the offspring $\boldsymbol{y}$ from the parent $\boldsymbol{x}$ by randomly selecting a component $\boldsymbol{x}_k$ and setting it to its upper value x_k^U or to its lower value x_k^L, according to the value of a random binary digit. The probabilities of different operators are defined as follows: 0.4 for uniform mutation, 0.4 for non-uniform mutation, and 0.2 for boundary mutation. Selection is the classical roulette wheel selection with linear fitness scaling. No parameter tuning or self-adaptation have been used in any of the performed tests.

4.2 Ten-variable function

The first testing problem involves the minimization of a ten-variable function subjected to eight inequality constraints, three linear and five nonlinear (Hock and Schittkowski 1981). The constraint functions are normalized according to the procedure followed by Lin and Wu (2004) and the objective function is rescaled so that the problem is expressed in the following form:

Minimize

$$F(\boldsymbol{x})=\frac{1}{100}\left[x_1^2+x_2^2+x_1x_2-14x_1-16x_2+(x_3-10)^2+4(x_4-5)^2+(x_5-3)^2+\ldots\right.$$
$$\left.+2(x_6-1)^2+5x_7^2+7(x_8-11)^2+2(x_9-10)^2+(x_{10}-7)^2+100\right], \tag{23}$$

subject to

$$g_1(\boldsymbol{x})=[4x_1+5x_2-3x_7+9x_8]/105-1\le 0, \tag{24}$$

$$g_2(\boldsymbol{x})=\left[3(x_1-2)^2+4(x_2-3)^2+2x_3^2-7x_4\right]\Big/120-1\le 0, \tag{25}$$

$$g_3(\boldsymbol{x})=[10x_1-8x_2-17x_7+2x_8]/10\le 0, \tag{26}$$

$$g_4(\boldsymbol{x})=\left[x_1^2+2(x_2-2)^2-2x_1x_2+14x_5-6x_6\right]\Big/100\le 0, \tag{27}$$

$$g_5(\boldsymbol{x})=[-8x_1+2x_2+5x_9-2x_{10}]/12-1\le 0, \tag{28}$$

$$g_6(\boldsymbol{x})=\left[5x_1^2+8x_2+(x_3-6)^2-2x_4\right]\Big/40-1\le 0, \tag{29}$$

$$g_7(\boldsymbol{x})=\left[3x_1+6x_2+12(x_9-8)^2-7x_{10}\right]\Big/100\le 0, \tag{30}$$

$$g_8(\boldsymbol{x})=\left[1/2(x_1-8)^2+2(x_2-4)^2+3x_5^2-x_6\right]\Big/30-1\le 0, \tag{31}$$

$$-10\le x_i\le 10,\quad \text{for}\ \ i=1,\ldots,10. \tag{32}$$

The optimum solution is $\boldsymbol{x}^*$=(2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927) with $F(\boldsymbol{x}^*)$=0.7931 after rescaling (24.3062091 in the original formulation by Hock and Schittkowski 1981). At the global optimum, the first six constraints are active. In addition, this problem has a very small feasible region, with a feasibility percentage computed with over 1,000,000 points equal to 0.0003% (Michalewicz and Schoenauer 1996).

To compare the obtained results with those by Wu and Lin (2004), the population size is set to 100 and all evolutive searches are terminated after 2,000 generations. The convergence histories of the best feasible designs in 20 searches at the end of each generation are shown in Fig. 1. The results are also compared with those provided by the most effective approaches found in literature. As can be seen from the synthetic data reported in Table 1, the present approach performs better than the two SOAPS versions, both on average and in terms of worst solution. It is remarkable that the algorithm found a point into the domain of attraction of the global optimum in 18 out of a total of 20 runs.

4.3 Five-variable function

The second testing problem involves the minimization of a five-variable function subjected to six nonlinear inequality constraints (Himmelblau 1972). This problem is defined as follows.

Minimize

$$F(\boldsymbol{x})=5.3578547x_3^2+0.8356891x_1x_5+37.293239x_1-40792.141, \tag{33}$$

subject to

$$g_1(\boldsymbol{x})=85.334407+0.0056858x_2x_5+0.00026x_1x_4-0.0022053x_3x_5\le 0, \tag{34}$$

$$g_2(\boldsymbol{x})=80.51249+0.0071317x_2x_5+0.0029955x_1x_2+0.0021813x_3^2\le 0, \tag{35}$$

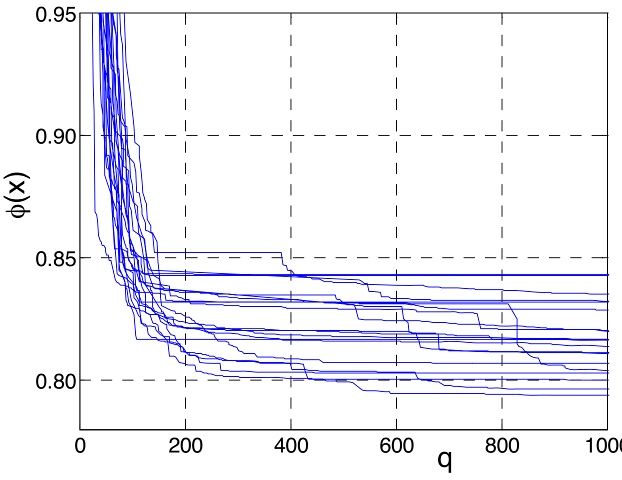

Fig. 1 Convergence histories of 20 runs in the ten-variable problem

$$g_3(\boldsymbol{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 \leq 0, \tag{36}$$

$$0 \leq g_1(\boldsymbol{x}) \leq 92, \tag{37}$$

$$90 \leq g_2(\boldsymbol{x}) \leq 110, \tag{38}$$

$$20 \leq g_3(\boldsymbol{x}) \leq 25, \tag{39}$$

$$78 \leq x_1 \leq 102, \tag{40}$$

$$33 \leq x_2 \leq 45, \tag{41}$$

$$27 \leq x_3 \leq 45, \tag{42}$$

$$27 \leq x_4 \leq 45, \tag{43}$$

$$27 \leq x_5 \leq 45. \tag{44}$$

This problem has been used before as a benchmark for several penalty-based GA techniques (Gen and Cheng 1997; Coello 2002). To compare the obtained results with those by Coello (2002), the population size is set to 50, and all evolutive searches are terminated after 100 generations, thus, having an upper limit of 5,000 function evaluations. Results reported in Table 2 refer to 30 runs and are compared with the best approaches by Coello (2000a, b): the MGA and the co-evolutionary penalty. Other approaches, including mathematical programming techniques, and different penalty-based GA have not been reported, as they perform less. From the table, it can be observed that the present approach performs better than the others, both on average and in terms of the best solution found. The new best solution is $\boldsymbol{x}^*$=(78.0000, 33.0000, 27.0710, 45.0000), which provides the function value $F(\boldsymbol{x}^*)$=−31025.5576. It is remarkable that the present approach has found a better solution than the co-evolutionary penalty approach in 22 runs over a total of 30. This is even more remarkable if it is considered that, in the present approach, the function evaluations are limited to a maximum of 5,000, while the co-evolutionary penalty technique performed a considerably higher number of fitness function evaluations (900,000). However, in one run, the search method converged to a local minimum, providing the worst result that is worse than those provided by the compared methods. Nevertheless, the standard deviation is slightly smaller than those of the other approaches. The convergence histories of the best designs in ten searches at the end of each generation are shown in Fig. 2.

Table 1 Search results in the ten-variable problem (20 runs, population size=100, number of generations=1,000)

Strategy	SOAPS (Lin and Wu 2004)	SOAPS-II (Lin and Wu 2004)	Present approach
No. of global optimum	3/20	6/20	18/20
No. of feasible solutions	20/20	20/20	20/20
Best	0.797	0.793	0.794
Average	0.852	0.818	0.815
Worst	0.902	0.888	0.843

Table 2 Search results in the five-variable problem (30 runs, population size=50, number of generations=100)

Strategy	MGA (Coello 2000a)	Co-evolutionary penalty (Coello 2000b)	Present approach
Best	−31,005.7966	−31,020.859	−31,025.5576
Mean	−30,862.8735	−30,984.2407	−30,997.7252
Worst	−30,721.0418	−30,792.4077	−30,690.1867
Standard deviation	73.240	73.6335	72.3465

4.4 The welded beam

This test is the first engineering problem we faced to test the performance of our penalty approach. It is a very well-known benchmark test for optimization methods and consists in the determination of the minimum design cost of a welded beam subject to seven constraints on shear stress (τ), bending stress in the beam (σ), buckling load on the bar (P_c), end deflection of the beam (δ), and side constraints (Rao 1996). There are four design variables, namely $h(x_1)$, $l(x_2)$, $t(x_3)$, and $b(x_4)$, as illustrated in Fig. 3. In mathematical form, the problem is stated as follows.

Minimize

$$F(\boldsymbol{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2), \tag{45}$$

subject to

$$g_1(\boldsymbol{x}) = \frac{\tau(\boldsymbol{x})}{\tau_{\max}} - 1 \leq 0, \tag{46}$$

$$g_2(\boldsymbol{x}) = \frac{\sigma(\boldsymbol{x})}{\sigma_{\max}} - 1 \leq 0, \tag{47}$$

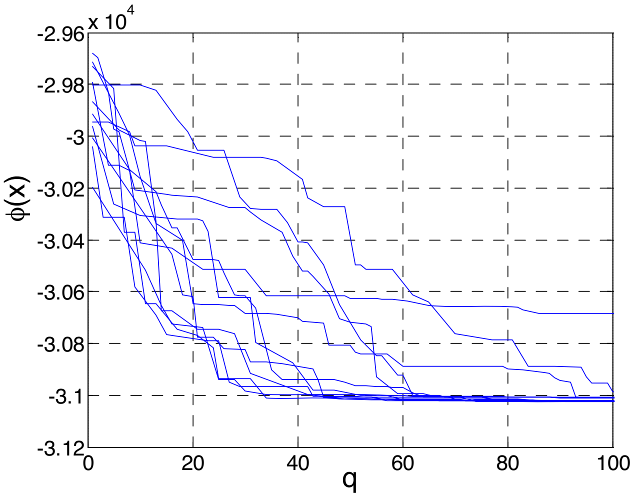

Fig. 2 Convergence histories of 10 runs (over a total of 30) in the five-variable problem

$$g_3(\boldsymbol{x}) = x_1 - x_4 \leq 0, \tag{48}$$

$$g_4(\boldsymbol{x}) = \frac{1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2)}{5} - 1 \leq 0, \tag{49}$$

$$g_5(\boldsymbol{x}) = 1 - \frac{x_1}{0.125} \leq 0, \tag{50}$$

$$g_6(\boldsymbol{x}) = \frac{\delta(x)}{\delta_{\max}} - 1 \leq 0, \tag{51}$$

$$g_7(\boldsymbol{x}) = 1 - \frac{P_c(x)}{P} \leq 0, \tag{52}$$

where

$$\tau(\boldsymbol{x}) = \sqrt{(\tau')^2 + \frac{2\tau'\tau''x_2}{2R} + (\tau'')^2}, \tag{53}$$

$$\tau' = \frac{P}{\sqrt{2}x_1x_2}, \; \tau'' = \frac{MR}{J}, \tag{54}$$

$$M = P\left(L + \frac{x_2}{2}\right), \; R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}, \tag{55}$$

$$J = 2\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right], \tag{56}$$

$$\sigma(\boldsymbol{x}) = \frac{6PL}{x_4x_3^2}, \tag{57}$$

$$\delta(\boldsymbol{x}) = \frac{4PL^3}{Ex_4x_3^3}, \tag{58}$$

$$P_c(\boldsymbol{x}) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right), \tag{59}$$

$$0.10 \leq x_1, x_4 \leq 2.0, \quad 0.10 \leq x_2, x_3 \leq 10.0, \tag{60}$$

where $\tau_{\max}$=13,600 psi, $\sigma_{\max}$=30,000psi, $d_{\max}$=0.25 in., P=6,000 lb, L=14 in., E=30×10^6 psi, and G=12×10^6 psi. The best solution available in literature, to the best of the authors' knowledge, is $\boldsymbol{x}$*=(0.2060, 3.4670, 9.0310, 0.2060), with an objective value F($\boldsymbol{x}$)=1.7329 (Lin and Wu 2004). Twenty runs were performed with a population size of 100 and a total number of 100 generations, with a limit of 10,000 function evaluations. In this case, as can be seen in Table 3, the proposed approach again found a better solution: $\boldsymbol{x}$*=(0.2023, 3.2118, 9.0311, 0.2060), with F($\boldsymbol{x}$*)=1.6857. The performance of the proposed method is quite satisfactory: The best solution is very good and also the average value is only surpassed by the co-evolutionary penalty method (Coello 2002), which, however, required by far more function evaluations. The worst solution is instead not very good and is comparable to that obtained by Wu and Lin (2004) with the Jones and Houck's method. The convergence histories of the best designs in ten searches at

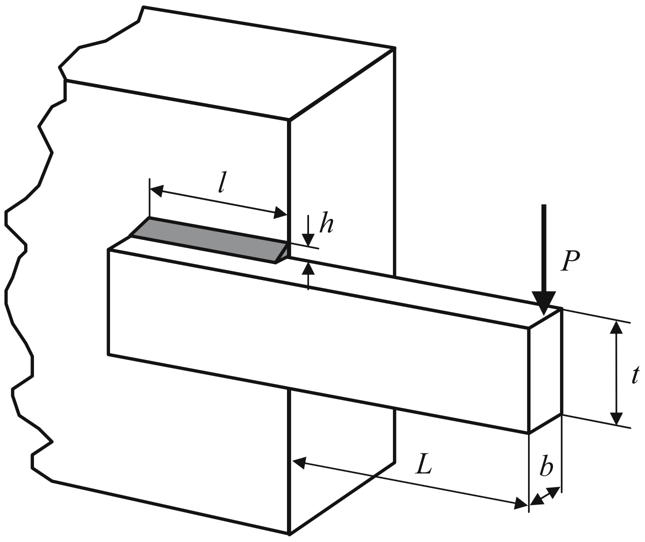

Fig. 3 Welded beam geometry

Table 3 Search results in the welded beam problem (20 runs, population size=100, number of generations=100)

Strategy	MGA (Coello 2000a)	Co-evolutionary penalty (Coello 2000b)	SOAPS (Lin and Wu 2004)	Joines and Houck (1994) (Wu and Lin 2004)	Present approach
No. of feasible solutions	NA	NA	20/20	20/20	20/20
Best	1.8245	1.7483	1.733	1.777	1.6857
Average	1.9190	1.7720	1.792	1.959	1.8304
Worst	1.9950	1.7858	1.911	2.226	2.1880
SD	0.05377	0.01122	NA	NA	0.1795

the end of each generation are shown in Fig. 4. Convergence to local optima is evident in four runs out of ten.

4.5 The ten-bar truss problem

The second engineering problem consists in the minimum weight design of the ten-bar cantilever truss shown in Fig. 5. Due to its simple configuration, this problem has been used as a benchmark to verify the efficiency of several optimization methods (Ali et al. 2003). The design variables are the cross-sectional areas of the ten bars, while constraints are imposed on the stress of each bar (both in tension and in compression) and on the maximum vertical displacement of the nodes to which the concentrated loads are applied.

Two versions of this problem have been studied in literature, one with continuously variable cross-sectional areas (Camp et al. 1998), and one in which each cross-sectional area must be selected from a discrete set (Rajeev and Krishnamoorty 1992). Because we have used a real-coded GA throughout this paper, we chose to deal with the first problem, although less realistic from the engineering point of view.

The problem is expressed in mathematical form as follows. Minimize the total weight of the truss:

$$F(\boldsymbol{x}) = \gamma\, l\left(\sum_{j=1}^{6} x_j + \sqrt{2}\sum_{j=7}^{10} x_j\right), \tag{61}$$

subject to the stress constraints, both in tension and in compression,

$$g_i(\boldsymbol{x}) = \frac{\sigma_i(\boldsymbol{x})}{\sigma_{\max}} - 1 \le 0, \quad i = 1, \ldots, 10, \tag{62}$$

$$g_{10+i}(\boldsymbol{x}) = -\frac{\sigma_i(\boldsymbol{x})}{\sigma_{\max}} - 1 \le 0, \quad i = 1, \ldots, 10\ ; \tag{63}$$

and with constraints on the maximum vertical displacements of the nodes to which the forces are applied:

$$g_{21}(\mathbf{x}) = \frac{\delta_2(\mathbf{x})}{\delta_{\max}} - 1 \le 0, \tag{64}$$

$$g_{22}(\mathbf{x}) = \frac{\delta_3(\mathbf{x})}{\delta_{\max}} - 1 \le 0. \tag{65}$$

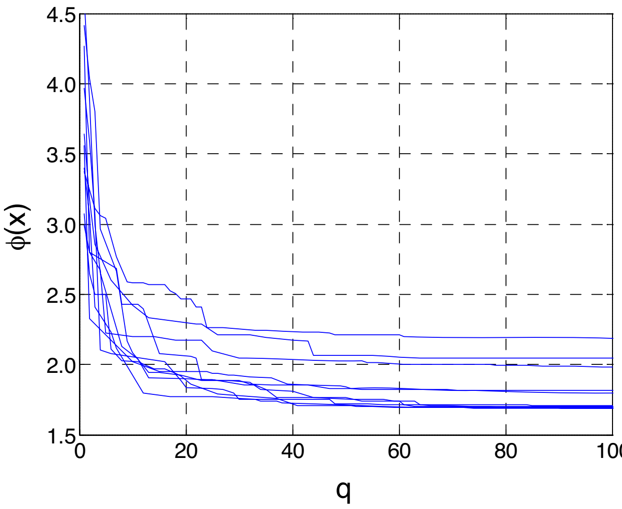

Fig. 4 Convergence histories of 10 runs (over a total of 20) in the welded beam problem

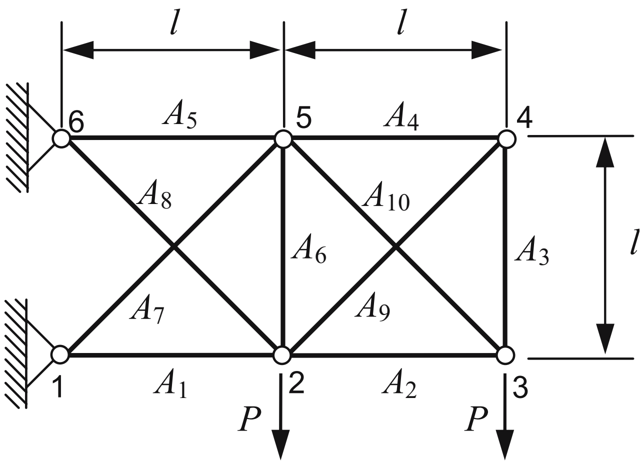

Fig. 5 Geometry of the ten-bar truss problem

Table 4 Search results in the ten-bar truss problem (10 runs, population size=100, number of generations=200)

Strategy	Camp et al. (1998)	Qian et al. (1982)	Venkayya (1971)	Present approach
A_1	24.07	23.545	23.416	23.350
A_2	13.96	14.96	14.904	14.846
A_3	0.56	0.297	0.53	0.101
A_4	0.10	0.10	0.128	0.100
A_5	28.92	30.67	30.416	30.621
A_6	0.10	0.10	0.101	0.100
A_7	21.95	21.07	21.084	20.779
A_8	7.69	8.578	8.578	7.419
A_9	0.10	0.10	0.101	0.100
A_{10}	22.09	20.96	21.077	21.518
Weight	5,076.31	5,089.82	5,090.17	5,024.48

The cross-sectional areas may vary in the following range:

$$0.1\ \text{in}^2 \le x_i \le 35.0\ \text{in}^2, \quad i = 1, \dots, 10. \tag{66}$$

The density of the bars is γ=0.1 lb/in.^{2}, the elasticity modulus E=104 ksi, the maximum allowable stress is $\sigma_{\max}$=25 ksi, and the maximum vertical displacement is $\delta_{\max}$=2 in. The concentrated loads are both equal to P= 100 kips. The numerical quantities have been expressed in engineering units rather than in SI units for the sake of comparison with results by other authors.

The best solution proposed in literature, to the best of the authors' knowledge, is $\boldsymbol{x}$*=(23.545, 14.96, 0.297, 0.1, 30.67, 0.1, 21.07, 8.578, 0.1, 20.96) with a total weight $F(\boldsymbol{x})$=5069.4 lb. Better solutions found in literature violated at least one of the displacement constraints, usually that on node 3 (see Fig. 5). For each search, the population size is selected as 100, and the genetic search is terminated after 200 generations. The best solution of ten genetic searches is reported in Table 4, compared to the best already-known solutions. The proposed method was not only able to find a better solution than those reported in literature, but also performed very well on average, as can be seen in the convergence history of ten runs; see Fig. 6. Over ten runs, the algorithm found ten feasible solutions with an average value of the weight equal to 5,043.6 lb, the worst solution with a weight of 5,079.4 lb, and a very small statistical deviation, namely 14.94 lb.

4.6 Six-variable function

As already mentioned, equality constraints are much more difficult for EAs than for traditional gradient-based methods. Thus, the last testing problem involves the minimization of a six-variable function subjected to three linear inequality constraints and three linear equality constraints (Floudas and Pardalos 1990). This problem is defined as follows.

Minimize

$$F(\boldsymbol{x}) = x_1^{0.6} + x_2^{0.6} + x_3^{0.4} + 2x_4 + 5x_5 - 4x_3 - x_6 + 25, \tag{67}$$

subject to

$$g_1(\boldsymbol{x}) = x_1 + 2x_4 - 4 \le 0, \tag{68}$$

$$g_2(\boldsymbol{x}) = x_2 + x_5 - 4 \le 0, \tag{69}$$

$$g_3(\boldsymbol{x}) = x_3 + x_6 - 6 \le 0, \tag{70}$$

$$h_1(\boldsymbol{x}) = x_2 - 3x_1 - 3x_4 = 0, \tag{71}$$

$$h_2(\boldsymbol{x}) = x_3 - 2x_2 - 2x_5 = 0, \tag{72}$$

$$h_2(\mathbf{x}) = 4x_4 - x_6 = 0, \tag{73}$$

$$0 \le x_1 \le 3, \tag{74}$$

$$0 \le x_2, x_3 \le 4, \tag{75}$$

$$0 \le x_4, x_5 \le 2, \tag{76}$$

$$0 \le x_6 \le 6. \tag{77}$$

The best solution proposed in literature, to the best of the authors' knowledge, is $\boldsymbol{x}$*=(0.167, 2, 4, 0.5, 0, 2) with a corresponding value of the objective function $F(\boldsymbol{x})$=11.598

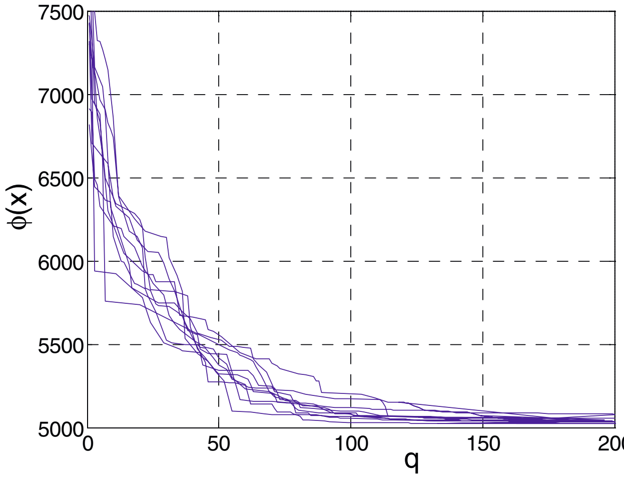

Fig. 6 Convergence histories of ten runs in the ten-bar truss problem

Table 5 Search results in the six-variable problem (20 runs, population size=200, number of generations=1,000)

Strategy	Lin and Hajela (1992) ($r0$=2.0)	Lin and Hajela (1992) ($r0$=5.0)	SOAPS (Lin and Wu 2004)	SOAPS II (Wu and Lin 2004)	Present approach[a]
No. of feasible solutions	20/20	20/20	20/20	20/20	20/20
Best	11.606	11.645	11.686	11.597	11.597
Average	11.847	12.369	12.616	11.647	11.616
Worst	12.484	13.376	13.741	11.803	11.668
Standard deviation	NA	NA	NA	NA	0.021

[a] (78)

(Wu and Lin 2004). In the original problem formulation, the upper limits for variables x_2, x_4, and x_6 were not defined; those specified in this study are equal to those used by Wu and Lin (2004), for the sake of comparison. For the same reason, the precision on the six variables is selected as 0.001, the population size is set to 200, and all evolutive searches are terminated after 1,000 generations, thus, resulting in an upper limit of 200,000 function evaluations.

In this case, the performance of the proposed penalty strategy (20) was not satisfactory, in particular, compared with that of the SOAPS-II strategy presented by Wu and Lin (2004); to improve the proposed strategy, we could have used a dynamic constraint relaxation technique such as that originally proposed by Hamida and Schoenauer (2002) in ASCHEA. Nevertheless, we decided not to use such a strategy because the choice of the amount of relaxation and its change rate during the evolution would have been crucial to the performance of the algorithm. We instead started from the observation that traditional penalty formulations, such as that by Lin and Hajela (1992), perform better, in the case of equality constraints, if the parameters provide not-too-severe penalties (this behavior has been reported, in the case of this testing problem, by Wu and Lin 2004). Therefore, we decided to modify (20) to lower the penalties associated to equality constraints through the introduction of a factor K:

$$P(\boldsymbol{x}, q) = \prod_{j=1}^{m}\left[1 + \frac{\widehat{g}_j(\boldsymbol{x})}{b_j}\left(\frac{q+Q}{Q}\right)\right]\prod_{k=1}^{p}\left[1 + K(q)\frac{|h_k(\boldsymbol{x})|}{c_k}\left(\frac{q+Q}{Q}\right)\right]. \tag{78}$$

The best results were obtained using the following generation-dependent factor:

$$K(q) = \frac{1}{r(Q-1)}[(r-1)q + (Q-r)], \tag{79}$$

which equals $1/r$ at the first generation and then increases linearly up to 1 at the end of the evolutionary process. By doing so, at the beginning of the search, the equality constraints provide a penalty that is r times milder than that of inequality constraints for the same amount of constraint violation. At the end of the process, on the other hand, the penalties provided by the equality and inequality constraints are equal if the violations are equal.

In the simulations, we used r=20; it is, however, clear that in this case the approach presents a user-defined parameter, namely r. Whether this parameter is problem dependent or not should be investigated by performing tests on different problems. Should it be independent, the approach could still be parameter free.

The results reported in Table 5 refer to 20 runs and are compared with the results by Wu and Lin (2004). The convergence history of ten runs is shown in Fig. 7. Over 20 runs, the algorithm found 20 feasible solutions, and all of them were in the global optimum domain of attraction. As can be seen from the table, the proposed technique yields better results than other techniques and even better than those obtained with the SOAPS-II approach, which requires (and profits from) the constraint violation information of each solution in the current population, whereas our approach does not.

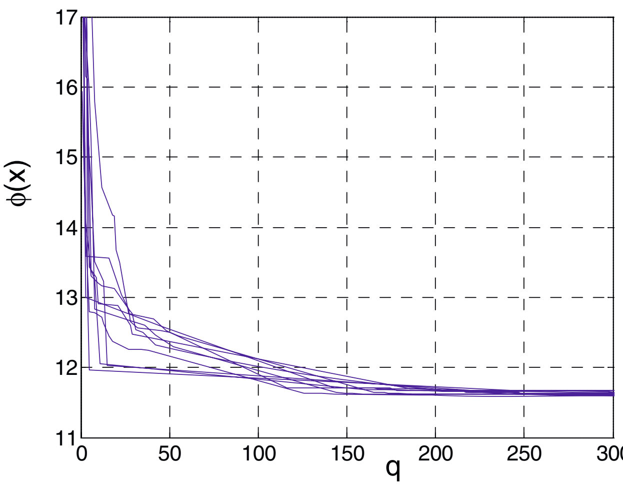

Fig. 7 Convergence histories of ten runs in the six-variable problem with equality constraints

5 Conclusions

Penalty function strategies are the most commonly used methods for GAs to solve constrained optimizations. Although the most recent trends indicate that multi-objective techniques are a promising venue for future research into constrained optimization, traditional static penalty approaches are very often used due to their relative simplicity and robustness. Nevertheless, the tuning process on penalty parameters is often inefficient, as the optimal parameters themselves are problem dependent. To overcome this problem, self-adaptive penalty functions have been proposed in literature. In this paper, we presented a new double-multiplicative penalty function approach, which is parameter free, coupled with a real-coded GA. The proposed approach performed well in four test problems with inequality constraints, showing very good GA search capabilities compared to the best solutions provided in literature. In the case with equality constraints, the algorithm also provided very competitive results if the penalties associated to the equality constraints are made lighter. To this aim, the introduction of a user-defined parameter was required; it is not yet clear, however, if this parameter is problem dependent—this aspect deserves further investigation.

Acknowledgment Support by the European Community is gratefully acknowledged by the authors. Thanks are also due to the Italian Ministry of Education, University and Scientific Research (MIUR). We gratefully acknowledge the helpful comments and suggestions of the editor and two anonymous reviewers.

References

Ali N, Behdinan K, Fawaz Z (2003) Applicability and viability of a GA based finite element analysis architecture for structural design optimization. Comput Struct 81:2259–2271

Arutyunyan NK, Drozdov AD (1988) Optimization problems in the mechanics of growing solids. Mech Compos Mater 24:359–369

Azegami H (1990) Proposal of a shape-optimization method using a constitutive equation of growth. JSME Int J 33:64–71

Camp C, Pezeshk S, Cao G (1998) Optimized design of two-dimensional structures using a genetic algorithm. J Struct Eng 124:551–559

Coello CAC (2000a) Constraint-handling using an evolutionary multiobjective optimization technique. Civ Eng Environ Syst 17:319–346

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

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 191:1245–1287

Davis L (1987) Genetic algorithms and simulated annealing. Pitman, London

Eberhart RC, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceedings of the sixth international symposium on micromachine and human science. IEEE, New York, pp 39–43

Eshelman LJ, Caruana RA, Schaffer JD (1989) Biases in the crossover landscape. In: Schaffer J (ed) Proceedings of the third international conference on genetic algorithms. Morgan Kaufmann, San Mateo, pp 10–19

Floudas CA, Pardalos CM (1990) A collection of test problems for constrained global optimization algorithms. Berlin, Springer

Fogel DB (1995) Evolutionary computation. Toward a new philosophy of machine intelligence. Institute of Electrical and Electronic Engineers, New York

Fogel LJ (1966) Artificial intelligence through simulated evolution. Wiley, New York

Gen M, Cheng R (1996) Interval programming using genetic algorithms. In: Proceedings of the sixth international symposium on robotics and manufacturing, Montpellier, France

Gen M, Cheng R (1997) Genetic algorithms and engineering design. Wiley, New York

Goldberg DE (1989) Genetic algorithms in search, optimization and machine learning. Addison–Wesley, Reading

Gürdal Z, Haftka RT, Hajela P (1999) Design and optimization of laminated composites. Wiley, New York

Haftka RT, Starnes JH Jr (1976) Applications of a quadratic extended interior penalty functions for structural optimization. AIAA J 14:718–724

Hajela P (1990) Genetic search—an approach to the nonconvex optimization problem. AIAA J 26:1205–1210

Hamida SB, Schoenauer M (2002) ASCHEA: new results using adaptive segregational constraint handling. In: Proceedings of the congress on evolutionary computation. IEEE, Hawaii, pp 884–889

Himmelblau DM (1972) Applied nonlinear programming. McGraw-Hill, New York

Hock W, Schittkowski K (1981) Test examples for nonlinear programming codes, lecture notes in economics and mathematical systems 187. Springer, Berlin

Holland JH (1975) Adaptation in natural and artificial system. University of Michigan Press, Ann Arbor

Homaifar A, Lai SHY, Qi X (1994) Constrained optimization via genetic algorithms. Simulation 62:242–254

Joines J, Houck C (1994) On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs. In: Fogel D (ed) Proceedings of the first IEEE conference on evolutionary computation, Orlando, Florida. IEEE, New York, pp 579–584

Jones DR, Perttunen CD, Stuckman BE (1993) Lipschitzian optimization without the Lipschitz constant. J Optim Theory Appl 79:157–181

Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680

Le Riche RG, Haftka RT (1995) Improved genetic algorithm for minimum thickness composite laminate design. Comput Eng 5:143–161

Le Riche RG, Knopf-Lenoir C, Haftka RT (1995) A segregated genetic algorithm for constrained structural optimization. In: Eshelman LJ (ed) Proceedings of the sixth international conference on genetic algorithms (ICGA95), Pittsburgh, PA. Morgan Kaufmann, San Mateo, CA

Lin C-Y, Hajela P (1992) Genetic algorithms in optimization problems with discrete and integer design variables. Eng Optim 19:309–327

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

Mattheck C, Burkhardt S (1990) A new method of structural shape optimization based on biological growth. Int J Fatigue 12:185–190

Michalewicz Z (1996) Genetic algorithm + data structures = evolution programs, 3rd edn. Springer, Heidelberg

Michalewicz Z, Janikow C (1991) Genetic algorithms for numerical optimization. Stat Comput 1:2

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

Qian LX, Zhang WX, Siu YK, Zhang JT (1982) Efficient optimum design of structures—program DDDU. Comput Methods Appl Mech Eng 30:209–224

Rajeev S, Krishnamoorty CS (1992) Discrete optimization of structures using genetic algorithms. J Struct Eng ASCE 121:1233–1250

Rao SS (1996) Engineering optimization. Wiley, New York

Rechenberg I (1973) Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann–Holzboog Verlag, Stuttgart

Schwefel H-P (1981) Numerical optimization for computer models. Wiley, Chichester

Smith AE, Coit DW (1997) Constraint handling techniques—penalty functions. In: Back et al. (ed) Handbook of evolutionary computation. Oxford University Press, Oxford

Venkayya VB (1971) Design of optimum structures. Int J Comput Struct 1:265–309

Wright AH (1991) Genetic algorithms for real parameter optimization. In: Rawlins G (ed) Foundations of genetic algorithms. First workshop on the foundations of genetic algorithms and classifier systems. Morgan Kaufmann, San Mateo, pp 205–208

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

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