Continuous Relaxation of MINLP Problems by Penalty Functions: A Practical Comparison

M. Fernanda P. Costa[1], Ana Maria A.C. Rocha[2], and Edite M.G.P. Fernandes[2]

[1] Centre of Mathematics, University of Minho, 4800-058 Guimarães, Portugal, mfc@math.uminho.pt
[2] Algoritmi Research Centre, University of Minho, 4710-057 Braga, Portugal
arocha@dps.uminho.pt, emgpf@dps.uminho.pt

Abstract. A practical comparison of penalty functions for globally solving mixed-integer nonlinear programming (MINLP) problems is presented. The penalty approach relies on the continuous relaxation of the MINLP problem by adding a specific penalty term to the objective function. A new penalty algorithm that addresses simultaneously the reduction of the error tolerances for optimality and feasibility, as well as the reduction of the penalty parameter, is designed. Several penalty terms are tested and different penalty parameter update schemes are analyzed. The continuous nonlinear optimization problem is solved by the deterministic DIRECT optimizer. The numerical experiments show that the quality of the produced solutions are satisfactory and that the selected penalties have different performances in terms of efficiency and robustness.

Keywords: MINLP, continuous relaxation, penalty function, DIRECT

1 Introduction

In a continuous relaxation context, the mixed-integer nonlinear programming (MINLP) problem is formulated as a continuous bound constrained nonlinear programming (BCNLP) problem, by adding a special penalty function to the objective function in order to penalize integrality violation. In this study, we extend the work presented in [1,2] by using an exact method with guaranteed convergence to solve the BCNLP problem. The deterministic DIRECT optimizer [3] is selected. Further, a new penalty-type algorithm is designed. At each iteration k, the algorithm computes a δ^k-global minimizer of the BCNLP problem and reduces the error tolerances – for the optimality, δ^k, and for feasibility, η^k – when the integrality violation is at a satisfactory level. The performance of the algorithm is analyzed by using a practical comparison that involves six special penalty functions. Three well-known penalty functions taken from [4] and three other recently proposed in [1,2] are investigated. The problem to be addressed

has the form:

$$\begin{aligned} \min\ & f(x) \\ \text{subject to}\ & x \in C \subset \mathbb{R}^n \\ & x_i \in \mathbb{R} \text{ for } i \in I_c \subseteq I \equiv \{1, \ldots, n\} \\ & x_j \in \mathbb{Z} \text{ for } j \in I_d \subseteq I \text{ (in particular } x_j \in \{0, 1\}) \\ & I_c \cap I_d = \emptyset \text{ and } I_c \cup I_d = I \end{aligned} \tag{1}$$

where f is a nonlinear continuous function, $|I_c|$ and $|I_d|$ give the number of continuous and integer variables respectively and the set C, assumed to be compact, is $C = \{x \in \mathbb{R}^n : l_i \leq x_i \leq u_i,\ i \in I\}$ (l and u are the vectors of the lower and upper bounds on the variables respectively). Since $x_j \in \mathbb{Z}$ for $j \in I_d$, we define the feasible region of problem (1) as follows:

$$W = \{x \in C \subset \mathbb{R}^n : x_j \in \mathbb{Z} \text{ for } j \in I_d\}. \tag{2}$$

The continuous relaxation of a MINLP is obtained by relaxing the integrality conditions from $x_j \in \mathbb{Z}$, $j \in I_d$ to $x_j \in \mathbb{R}$, $j \in I_d$ (assuming that the f values can be computed for $x_j \in \mathbb{R}$, $j \in I_d$). We note that the presence of integer variables implies that the feasible region W is not convex. There are two classes of MINLP problems. In the context of problem (1), if the function f is convex, the MINLP problem is called convex; otherwise it is called nonconvex. A convex MINLP problem is easier to solve than a nonconvex one, since its continuous relaxation is itself a convex problem, and therefore likely to be tractable, at least in theory. By contrast, the continuous relaxation of a nonconvex MINLP is itself a global optimization problem, and therefore likely to be NP-hard. Reviews on MINLP techniques and applications are available in [5,6].

An exact continuous reformulation of MINLP problems using a specific class of penalty terms is studied in [4,7]. The equivalence between the MINLP problem and the continuous reformulated penalty problem is therein established. This issue concerned with penalty functions for problems with binary variables has been addressed in [8,9]. Although any bounded MINLP problem can be reformulated as a mixed-binary programming problem, this strategy can be troublesome and is not recommended in practice. The use of particular penalty terms to penalize integer constraints violation directly is preferred. In this context, a penalty term based on the 'erf' function is proposed and compared with other penalty alternatives available in the literature (see [1]). In [2], two different penalty functions are proposed. The hyperbolic tangent function and the inverse hyperbolic sine function are designed and their properties are established. The equivalence property between problem (1) and its continuous relaxation is studied. Another proposal, that can be seen in [10], transforms the MINLP problem into an equivalent nonlinear programming (NLP) problem by adding to the original constraints some linear and quadratic constraints. Then, a penalty function is used to transform the NLP problem into an unconstrained one. In [11], a new exact and smooth penalty function for the MINLP problem is presented by augmenting only one variable whatever the number of constraints.

In the sequence of the specific penalty approach, a continuous reformulation of the problem (1) comes out, by relaxing the integer constraints on the variables

and adding a particular penalty term to the objective function f,

$$\begin{aligned} \min\ & \psi(x;\varepsilon) \equiv f(x) + P(x;\varepsilon) \\ \text{subject to}\ & x \in C \\ & x_i \in \mathbb{R} \text{ for } i = 1,\ldots,n, \end{aligned} \tag{3}$$

where $\varepsilon \in \mathbb{R}^+$ is the penalty parameter and $P(x;\varepsilon)$ is the penalty term. Under suitable assumptions on the function f and the penalty $P(\cdot;\varepsilon)$, the function $\psi(\cdot;\varepsilon)$ in (3) is 'exact' in the sense that there exists a positive $\bar{\varepsilon}$ such that for $\varepsilon \in (0,\bar{\varepsilon}]$, problems (1) and (3) are equivalent, i.e., problems (1) and (3) have the same global minimizers [4,7].

In this study, we aim to address two important issues related to a penalty-type algorithm for solving MINLP problems. First, a new algorithm that is able to compute a sequence of approximations with error tolerances increasingly smaller on the optimality and on the integer infeasibility, is proposed. Second, using a set of penalty terms already available in the literature, we analyze their practical behavior when solving a benchmark set of bound constrained MINLP problems. We have also used an exact global optimizer for solving the continuous BCNLP problem, the DIRECT optimizer. Furthermore, this study also aims to analyze the relative performance of the penalty functions, within the exact penalty algorithm context, when different ε initialization and update schemes are tested.

This paper is organized as follows. In Section 2, we present the new penalty-based algorithm, by explaining the main ideas behind the penalty parameter and error tolerance updates, and illustrate the penalty functions used in the comparative experiments. In Section 3, the DIRECT optimizer is briefly described. Section 4 presents the results of the numerical experiments and some comparisons and Section 5 contains the conclusions of the present study.

2 Penalty algorithm

In this section, we propose a penalty algorithm to solve MINLP problems. Based on the equivalence statement between the problems (1) and (3), when some suitable penalty terms are used to penalize integrality violation, a finite sequence of BCNLP problems (3) is solved. It has been proven that there exists a positive $\bar{\varepsilon}$ such that for $\varepsilon \in (0,\bar{\varepsilon}]$, both problems have the same global minimizers [4]. For the design of the penalty algorithm, we need the following definitions:

Definition 1. *Let ε^k be fixed at iteration k and let $\psi(x;\varepsilon^k)$ be a continuous objective function defined over a bounded space $C \subset \mathbb{R}^n$. The approximation $x^k \in C$ (to the global optimal of problem (3)) is a δ^k-global minimizer of the problem (3) if $\psi(x^k;\varepsilon^k) \leq \min_{x\in C} \psi(x;\varepsilon^k) + \delta^k$, where $\delta^k > 0$ is the error bound which reflects the accuracy required for the approximation.*

Definition 2. *Let η^k be fixed at iteration k and let $z^k = [x^k]_r$ be a feasible approximation, where $z_i^k \in \mathbb{Z}, i \in I_d$ results from rounding x_i^k to the nearest*

integer and $z_j^k = x_j^k$ *for* $j \in I_c$. *The point* $x^k \in C$ *is an* η^k*-feasible approximation to the problem* (1) *if* $\|x^k - z^k\|_\infty \leq \eta^k$ *where* $\eta^k > 0$ *is the error bound which reflects the accuracy required for the approximation.*

Thus, it is shown in Algorithm 1 the main steps for finding a global solution to problem (1). At iteration k and for a fixed value of ε^k, the algorithm computes x^k, a δ^k-global minimizer of problem (3), which is an approximation to the global minimizer of (1). It is assumed that the sequence $\{x^k\}$ converges to the optimal solution x^* of (1), as long as $\eta^k \to 0$ and $\delta^k \to 0$, and a finite set of decreasing ε^k values are tested. The algorithm imposes the lower bound $\underline{\varepsilon}$ to prevent the BCNLP problem of becoming very hard to be solved. On the other hand, in practical terms, the sufficiently small positive error bounds $\underline{\eta}$ and $\underline{\delta}$, for η^k and δ^k respectively, are used to reflect the accuracy required for the solution.

Data: f^*, $k_{\max}$, $0 < \underline{\varepsilon} \ll 1$, $0 < \underline{\delta} \ll 1$, $0 < \underline{\eta} \ll 1$, $\varepsilon^1 > \underline{\varepsilon}$, $\delta^1 > \underline{\delta}$, $\eta^1 > \underline{\eta}$, $\sigma_\varepsilon \in (0,1)$, $\sigma_\delta \in (0,1)$, $\sigma_\eta \in (0,1)$;
Set $k = 1$;
repeat
  Compute a δ^k-global minimizer x^k of problem (3) such that

$$\psi(x^k; \varepsilon^k) \leq \psi(x; \varepsilon^k) + \delta^k, \quad \text{for all } x \in C; \tag{4}$$

  if $\|x^k - z^k\|_\infty > \eta^k$ **then**
    Set $\varepsilon^{k+1} = \max\{\sigma_\varepsilon \varepsilon^k, \underline{\varepsilon}\}$, $\delta^{k+1} = \delta^k$, $\eta^{k+1} = \eta^k$;
  else
    Set $\varepsilon^{k+1} = \varepsilon^k$, $\delta^{k+1} = \max\{\sigma_\delta \delta^k, \underline{\delta}\}$, $\eta^{k+1} = \max\{\sigma_\eta \eta^k, \underline{\eta}\}$;
  end
  Set $k = k + 1$;
until *(*$\|x^k - z^k\|_\infty \leq \underline{\eta}$ *and* $|f(x^k) - f^*| \leq \underline{\delta}$*) or* $k > k_{\max}$;

Algorithm 1: Penalty algorithm for MINLP problems

To check when the update of the penalty parameter is timely, the algorithm resorts to Definition 2. Thus, the penalty parameter is reduced whenever the computed approximation x^k is not an η^k-feasible approximation; otherwise the value is maintained. Further, when x^k satisfies the definition of an η^k-feasible approximation, the error tolerance parameters δ^k and η^k are reduced so that a better approximation to the global solution of the problem (3) can be found. The algorithm stops when the computed iterate x^k is in a $\underline{\eta}$ vicinity of a feasible point and is within an error of $\underline{\delta}$ of the global minimum f^*. We note that the use of the known global solution to stop the algorithm has the goal of analyzing its real effectiveness.

We note that any global optimizer for BCNLP problems can be used to compute the δ^k-global minimizer x^k. Since finding a global minimizer is much more difficult than finding a local one, the algorithms specially tailored for local optimization may converge to a local minimizer and the convergence entirely depends on the starting approximation. Thus, the global optimizer DIRECT is used to find a solution that satisfies (4) (see the details in the next section).

We now illustrate the set of penalty terms that will be used to solve MINLP problems, in this penalty algorithm context. Three popular penalty terms [4,7,8] are:

$$P(x;\varepsilon) = \sum_{j \in I_d} \min_{\substack{l_j \le d_i \le u_j \\ d_i \in \mathbb{Z}}} \log\left(|x_j - d_i| + \varepsilon\right), \tag{5}$$

$$P(x;\varepsilon) = \frac{1}{\varepsilon} \sum_{j \in I_d} \min_{\substack{l_j \le d_i \le u_j \\ d_i \in \mathbb{Z}}} \left\{[|x_j - d_i| + \varepsilon]^p\right\}, \ 0 < p < 1, \tag{6}$$

and

$$P(x;\varepsilon) = \frac{1}{\varepsilon} \sum_{j \in I_d} \min_{\substack{l_j \le d_i \le u_j \\ d_i \in \mathbb{Z}}} \left\{[1 + \exp\left(-\rho|x_j - d_i|\right)]^{-1}\right\}, \ \rho > 0. \tag{7}$$

The two most recently proposed penalty terms for solving MINLP problems are based on the hyperbolic tangent function, $\tanh(\cdot)$,

$$P(x;\varepsilon) = \frac{1}{\varepsilon} \sum_{j \in I_d} \min_{\substack{l_j \le d_i \le u_j \\ d_i \in \mathbb{Z}}} \tanh\left(|x_j - d_i| + \varepsilon\right) \tag{8}$$

and on the inverse hyperbolic sine, $\operatorname{asinh}(\cdot)$, both differentiable and strictly increasing functions on $[0, +\infty)$ [2],

$$P(x;\varepsilon) = \sum_{j \in I_d} \min_{\substack{l_j \le d_i \le u_j \\ d_i \in \mathbb{Z}}} \operatorname{asinh}\left(\frac{1}{\varepsilon}|x_j - d_i| + \varepsilon\right). \tag{9}$$

Finally, the penalty term proposed in [1] uses the 'erf' function,

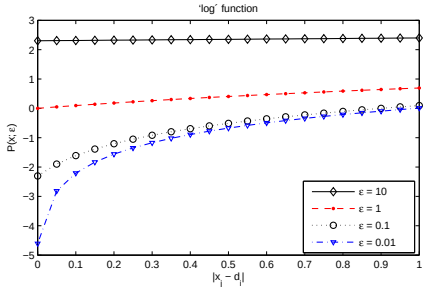
$$P(x;\varepsilon) = \frac{1}{\varepsilon} \sum_{j \in I_d} \min_{\substack{l_j \le d_i \le u_j \\ d_i \in \mathbb{Z}}} \left\{\operatorname{erf}\left(|x_j - d_i| + \varepsilon\right)\right\} \tag{10}$$

and has been also selected for this study.

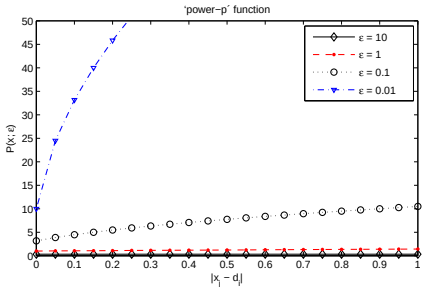
The six plots in Fig. 1 show the behavior of the above illustrated penalty terms when four different values of ε are used (10, 1, 0.1 and 0.01). Figure 1(a) displays the penalty term based on the 'log' function (see (5)), while Fig. 1(b) and Fig. 1(c) illustrate the behavior of the penalty terms presented in (6) and (7) respectively. In Fig. 1(d), we show the behavior of the hyperbolic tangent function, and in Fig. 1(e) the inverse hyperbolic sine is plotted. Finally, the 'erf' function is plotted in Fig. 1(f). As it can be seen, the functions (6), (7), (8), (9) and (10) are positive for $\varepsilon > 0$ while the function (5) may reach negative values. The hyperbolic tangent function and the 'erf' function have similar behaviors as a function of ε and the penalty functions that provide a faster penalty increase, as the integrality violation increases, are the functions (6), (7), (8) and (10).

3 DIRECT optimizer

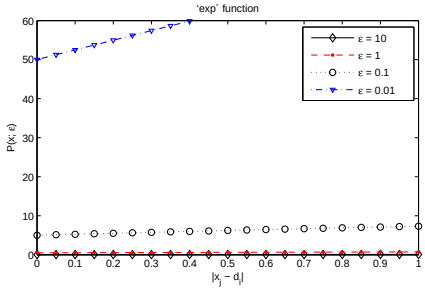
When solving a nonconvex optimization problem, a global optimizer is recommended so that there exists some guarantee of convergence to a global solution and to avoid convergence to a local one.



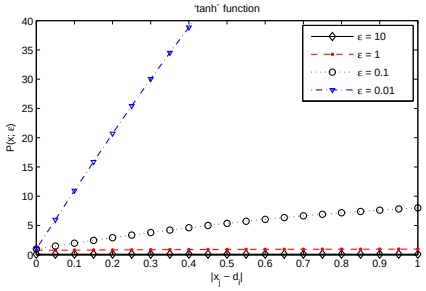
(a) Penalty (5).



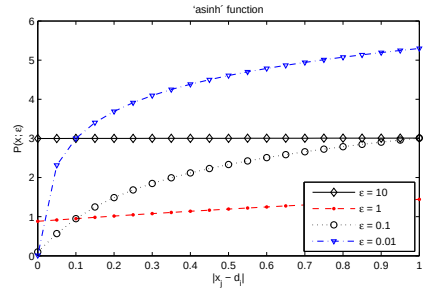
(b) Penalty (6).



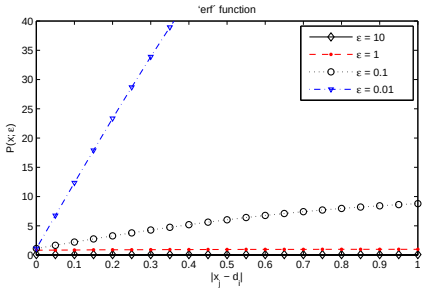
(c) Penalty (7).



(d) Penalty (8).



(e) Penalty (9).



(f) Penalty (10).

Fig. 1. Behavior of the penalty terms, for four different values of ε.

This section aims to briefly describe the main ideas behind a partition-based algorithm, called DIRECT [3], that is capable of searching for a global optimal solution to the BCNLP problem by defining a sequence of partitions of the search region C. The DIRECT (DIviding RECTangles) algorithm has been originally proposed to obtain global solutions to continuous BCNLP problems [3], where the objective penalty function ψ must satisfy a Lipschitz condition

$$|\psi(x_1;\cdot) - \psi(x_2;\cdot)| \leq K\|x_1 - x_2\| \text{ for all } x_1, x_2 \in C,$$

for $K > 0$, by producing finer and finer partitions of the hyperrectangles generated from C. The algorithm is a modification of the standard Lipschitzian approach that does not require any derivative information or the value of the Lipschitz constant K. DIRECT has the ability to explore potentially optimal regions aiming to converge to the global optimum, while avoiding being trapped in local optima. This search is carried out by dividing all hyperrectangles that are potentially optimal. Notice the following definition:

Definition 3. *Given the partition $\{C^i : i \in J\}$ of C, let ν be a positive constant and let $\psi_{\min}$ be the current best objective function value. A hyperrectangle j is said to be potentially optimal if there exists some rate-of-change constant $\hat{K} > 0$ such that*

$$\begin{aligned} &\psi(c_j;\cdot) - \hat{K}d_j \leq \psi(c_i;\cdot) - \hat{K}d_i, \text{ for all } i \in J \\ &\psi(c_j;\cdot) - \hat{K}d_j \leq \psi_{\min} - \nu|\psi_{\min}| \end{aligned} \tag{11}$$

where c_j is the center and d_j is a measure of the size of the hyperrectangle j.

We note that the use of $\hat{K}$ intends to show that it is not the Lipschitz constant. The division of the potentially optimal hyperrectangles is carried out only along the dimensions of maximal size, the division is into thirds and follows a specific order. The reader is referred to [3,12,13] for details. A subsequential convergence result for the DIRECT algorithm is established in [14]. In [3], the DIRECT algorithm for solving the BCNLP problem is described by its six main steps:

1. In the initialization step, the search space C is normalized to an n-dimensional unit hypercube, the center point c_1 and $\psi(c_1;\cdot)$ are determined, $\psi_{\min}$, the iteration counter and the set of indices of partition-based hyperrectangles are initialized.
2. In the selection step, for each partition of C, the set of potentially optimal hyperrectangles are identified.
3. In the sampling step, for each potentially optimal hyperrectangle, the set of dimensions with the maximum size is identified, and the objective function ψ is sampled at center points along those identified dimensions.
4. In the division step, using a specific order, hyperrectangles are divided into thirds along those identified dimensions.
5. The iteration step aims to update $\psi_{\min}$, the set of indices of partition-based hyperrectangles, and the set of potentially optimal hyperrectangles not yet explored.
6. In the termination step, the stopping conditions are checked so that steps 2–5 are repeated or the algorithm is terminated.

In the Algorithm 1 context, the δ^k-global minimizer of problem (3) is found when the condition (4) illustrated in the algorithm is satisfied. However, during this algorithm's practical evaluation (assuming that f^* is provided) the DIRECT optimizer terminates when the condition

$$\psi(x^k;\varepsilon^k) \leq f^* + \mathcal{C}(\varepsilon^k) + \delta^k$$

holds for the iterate x^k. For fixed ε^k, the term $\mathcal{C}(\varepsilon^k) \equiv P(x^*;\varepsilon^k) = P(\bar{x};\varepsilon^k)$ is constant for any $\bar{x} \in W$, and equals $|I_d|\log(\varepsilon^k)$ for the penalty (5), $|I_d|(\varepsilon^k)^{p-1}$ for the penalty (6), $|I_d|(2\varepsilon^k)^{-1}$ for the penalty (7) (with $\rho = 1$), $|I_d|(\varepsilon^k)^{-1}\tanh(\varepsilon^k)$ for the penalty (8), $|I_d|\,\mathrm{asinh}(\varepsilon^k)$ for the penalty (9) and $|I_d|(\varepsilon^k)^{-1}\,\mathrm{erf}(\varepsilon^k)$ for the penalty (10).

Alternatively, DIRECT is stopped when the first, a maximum number of iterations, or a maximum number of function evaluations, is reached.

4 Numerical Experiments

In this section, a practical comparison of the penalty terms illustrated in Section 2, for solving MINLP problems, in the context of the proposed Algorithm 1 and using the DIRECT optimizer for solving the continuous BCNLP problem (3), is presented. The numerical experiments were carried out on a PC Intel Core 2 Duo Processor E7500 with 2.9GHz and 4Gb of memory RAM. The algorithm was coded in Matlab Version 8.1 (R2013a).

The comparisons rely on 18 instances of 14 well-known MINLP problems. Eight problems have $n = 2$ variables, three have $n = 4$ variables, problem Dixon-Price is tested with $n = 2$ and $n = 4$, problem Sum Squares is tested with $n = 5$, and problems Ackley, Levy and Rastrigin are tested with $n = 5$ and 10. See [2] for the full description of the problems.

The parameters in the Algorithm 1 are set as follows: $\delta^1 = 1$, $\eta^1 = 1$, $\underline{\delta}$=1E-04, $\underline{\eta}$=1E-08, $\sigma_\delta = 0.1$, $\sigma_\eta = 0.1$ and $k_{\max} = 20$. The threshold number of iterations and function evaluations in the DIRECT solver are set to 100 and 50000 respectively. Other parameters for the penalty functions are $p = 0.5$ and $\rho = 1$.

The two parameters ε^1 (initialization of the penalty parameter) and σ_ε (the reduction factor) are analyzed in terms of penalty sensitivity, i.e., we aim to conclude if any of the penalty terms performs consistently better than the others for some pair of values of the parameters. During these experiments $\underline{\varepsilon}$ is set to 1E-12. The comparisons are made by using a graphical procedure to visualize the differences in the performance of the six penalty terms, in relative terms on the 18 instances, known as performance profiles [15]. The performance is analyzed in terms of the efficiency of the penalty. Thus, for the performance metric, P_m, the number of function evaluations is used. The performance function of case j in comparison is the (cumulative) distribution function $F_j(\tau)$ (for P_m), i.e., is the 'probability' that the case j is within a factor τ of the best possible case. For each j, the $F_j(\tau)$ is a (weakly) monotonically increasing function in τ. A performance profile is the plot of all functions $F_j(\cdot)$. The higher the 'probability'

the better. A higher value for $\tau = 1$ means that the corresponding case achieves the smallest metric value, i.e., the fewer number of function evaluations, mostly.

The performance profile for the configuration $\varepsilon^1 = 10$ and $\sigma_\varepsilon = 0.1$ is shown in Fig. 2. We also show in Table 1 the percentage of success (in %) for each penalty. Here, the run with a certain penalty is considered a success if the algorithm terminates with the solution x^k that satisfies the conditions $\|x^k - z^k\|_\infty \leq \underline{\eta}$ and $|f(x^k) - f^*| \leq \underline{\delta}$, for the given values of $\underline{\eta}$ and $\underline{\delta}$.

From the Fig. 2 (and for $\tau = 1$) we may conclude that the probability that the 'log' function, as well as the 'asinh' function, are the winners on a given problem is about 0.39 since each one of them requires fewer function evaluations than the other penalty functions in seven of the 18 instances. The 'power-p' function has a probability of winning of 0.22 (requiring fewer function evaluations in four of the 18 instances). The other penalties in comparison did not reach the smallest number of function evaluations in any of the tested instances. Although being one of the most efficient (with the configuration $\varepsilon^1 = 10$, $\sigma_\varepsilon = 0.1$), the 'log' function is the least robust with a percentage of success of 78%, against 83% of the other penalties (see the percentages along the first row of the Table 1).

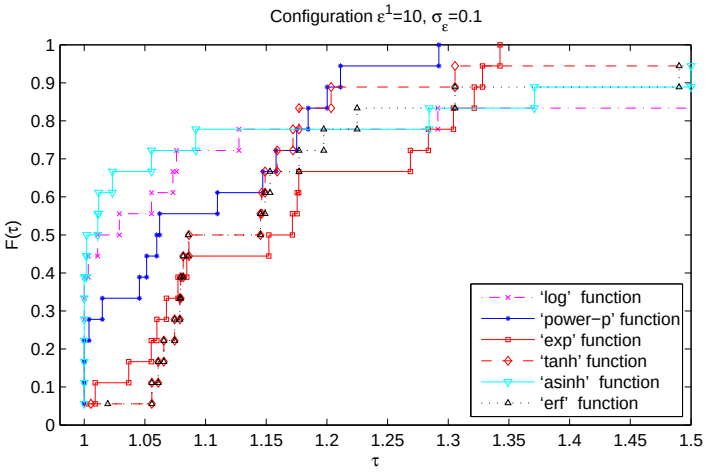


Fig. 2. Comparison of the six penalty terms, when $\varepsilon^1 = 10$ and $\sigma_\varepsilon = 0.1$, based on the performance profile.

Figure 3 shows the performance profile when $\varepsilon^1 = 1$ and $\sigma_\varepsilon = 0.5$ are considered. We may conclude that the penalty with the highest probability of winning, requiring fewer function evaluations to reach the global solution, is the 'power-p', 0.44, followed by the 'asinh' function with a probability of 0.22, by the 'erf' with a probability of 0.17 and by 'log' and 'tanh', both with probability of 0.11. The percentages of success of this configuration are shown along the second row of the Table 1. The penalties 'exp', 'tanh' and 'erf' are the most robust.

Table 1. Percentage of success (%) for each penalty and each configuration

Configuration	'log'	'power-p'	'exp'	'tanh'	'asinh'	'erf'
$\varepsilon^1 = 10$, $\sigma_\varepsilon = 0.1$	78	83	83	83	83	83
$\varepsilon^1 = 1$, $\sigma_\varepsilon = 0.5$	61	78	83	83	72	83
$\varepsilon^1 = 0.1$, $\sigma_\varepsilon = 0.5$	56	67	67	67	56	67

On the other hand, by analyzing the performance functions in Fig. 4, where the configuration $\varepsilon^1 = 0.1$ and $\sigma_\varepsilon = 0.5$ is tested, it is possible to conclude that the most efficient penalty is the 'power-p' with a probability of winning of 0.61, followed by the 'erf' penalty with a probability of 0.22, by the 'log' with probability of 0.11 and by the 'exp' with probability of 0.06. Further, from the percentages shown in the third row of the Table 1, we conclude that this is the configuration that produces the worst results, in the sense that the required final error tolerances for the optimality, $\underline{\delta}$, and for the feasibility, $\underline{\eta}$, are attained for a much smaller number of instances, being even so the penalties 'power-p', 'exp', 'tanh' and 'erf' the most robust.

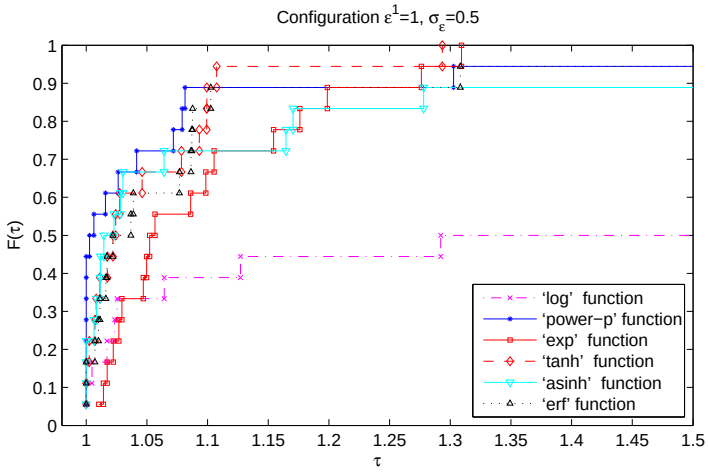


Fig. 3. Comparison of the six penalty terms, when $\varepsilon^1 = 1$ and $\sigma_\varepsilon = 0.5$, based on the performance profile.

5 Conclusions

In this paper, we have developed a new penalty algorithm for solving nonconvex MINLP problems. The algorithm is based on a continuous relaxation of the MINLP problem by adding to the objective function a specific penalty term

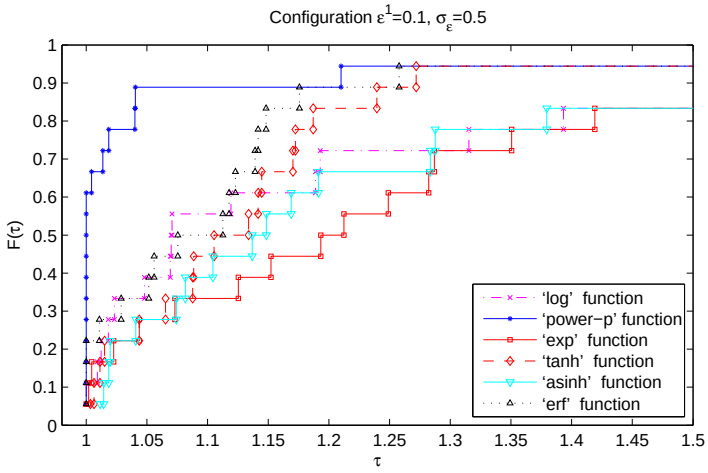


Fig. 4. Comparison of the six penalty terms, when $\varepsilon^1 = 0.1$ and $\sigma_\varepsilon = 0.5$, based on the performance profile.

that aims to penalize integrality violation. The algorithm computes (a sequence of) δ^k-global minimizers of a BCNLP problem, for a decreasing sequence of δ^k values, using the DIRECT optimizer. A sequence of decreasing error tolerances, incorporated in the algorithm to measure integer infeasibility, is used to check when the update of the penalty parameter is timely. Six penalty terms are tested in the proposed algorithm context.

Numerical experiments have been carried out to analyze the penalty term sensitivity to the initialization and update scheme of the penalty parameter. The comparisons are based on the performance profile relative to an efficiency metric. With this comparative study, we are able to conclude that the selected penalties behave differently as far as the efficiency metric is concerned. Two penalties, one based on a power function and the other based on the 'asinh' function, are consistently the most efficient, and the penalties with the power term and with the 'log' function are less sensitive to changes in ε. On the other hand, penalties 'asinh' and 'erf' are the ones whose performances vary the most with the penalty parameter values. It has been shown that the configuration that uses 10 as the initial value and 0.1 as the reduction factor produces the best results in terms of number of instances that converged to the global solution with the required accuracy. Overall, in comparative terms, we may say that the penalties 'exp', 'tanh' and 'erf' are consistently more robust and the penalties 'power-p' and 'asinh' require in general fewer function evaluations.

Acknowledgments. The authors wish to thank two anonymous referees for their comments and suggestions.

This work has been supported by COMPETE: POCI-01-0145-FEDER-007043 and FCT - Fundação para a Ciência e Tecnologia, within the projects UID/CEC/00319/2013 and UID/MAT/00013/2013.

References

1. Francisco, R.B., Costa, M.F.P., Rocha, A.M.A.C., Fernandes, E.M.G.P.: Comparison of penalty functions on a penalty approach to mixed-integer optimization. In AIP Conf. Proc. vol. 1738 (ICNAAM 2015) 300008-1–300008-4 (2016)
2. Costa, M.F.P., Rocha, A.M.A.C., Francisco, R.B., Fernandes, E.M.G.P.: Firefly penalty-based algorithm for bound constrained mixed-integer nonlinear programming. Optimization, 65(5), 1085–1104 (2016)
3. Jones D.R., Perttunen C.D., Stuckman B.E.: Lipschitzian optimization without the Lipschitz constant. Journal of Optimization Theory and Applications, 79(1), 157–181 (1993)
4. Lucidi, S., Rinaldi, F.: Exact penalty functions for nonlinear integer programming problems. Journal of Optimization Theory and Applications, 145(3), 479–488 (2010)
5. Burer, S., Letchford, A.N.: Non-convex mixed-integer nonlinear programming: A survey. Surveys in Operations Research and Management Science, 17(2), 97–106 (2012)
6. Boukouvala, F., Misener, R., Floudas, C.A.: Global optimization advances in mixed-integer nonlinear programming, MINLP, and constrained derivative-free optimization, CDFO. European Journal of Operational Research, 252(3), 701–727 (2016).
7. Lucidi, S., Rinaldi, F.: An exact penalty global optimization approach for mixed-integer programming problems. Optimization Letters, 7(2), 297–307 (2013)
8. Murray, W., Ng, K.-M.: An algorithm for nonlinear optimization problems with binary variables. Computational Optimization and Applications, 47(2), 257–288 (2010)
9. Shandiz, R.A., Mahdavi-Amiri, N.: An exact penalty approach for mixed integer nonlinear programming problems. American Journal of Operations Research, 1(3), 185–189 (2011)
10. Yu, C., Teo, K.L.,Bai, Y.: An exact penalty function method for nonlinear mixed discrete programming problems, Optimization Letters, 7(1), 23–38 (2013)
11. Ma, C., Zhang, L.: On an exact penalty function method for nonlinear mixed discrete programming problems and its applications in search engine advertising problems. Applied Mathematics and Computation, 271, 642–656 (2015)
12. Gablonsky J.M., Kelley C.T.: A locally–biased form of the DIRECT algorithm. Journal of Global Optimization, 21(1), 27–37 (2001)
13. Finkel D.E.: DIRECT Optimization Algorithm User Guide. Center for Research in Scientific Computation, North Carolina State University (2003)
14. Finkel D.E., Kelley C.T.: Convergence analysis of the DIRECT algorithm. Technical Report CRSC-TR04-28, Center for Research in Scientific Computation, North Carolina State University (2004)
15. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Mathematical Programming, Series A, 91(2), 201–213 (2002)