Optimization
Vol. 00, No. 00, November 2014, 1–23

RESEARCH ARTICLE

Firefly Penalty-based Algorithm for Bound Constrained Mixed-Integer Nonlinear Programming

M. Fernanda P. Costa[a] *, Ana Maria A. C. Rocha[b], Rogério B. Francisco[a] and Edite M. G. P. Fernandes[b]

[a] *Centre of Mathematics, University of Minho, 4710-057 Braga, Portugal*
[b] *Algoritmi Research Centre, University of Minho, 4710-057 Braga, Portugal*

(*v1.0 released February 2011*)

In this article, we aim to extend the firefly algorithm (FA) to solve bound constrained mixed-integer nonlinear programming (MINLP) problems. An exact penalty continuous formulation of the MINLP problem is used. The continuous penalty problem comes out by relaxing the integrality constraints and by adding a penalty term to the objective function that aims to penalize integrality constraint violation. Two penalty terms are proposed, one is based on the hyperbolic tangent function and the other on the inverse hyperbolic sine function. We prove that both penalties can be used to define the continuous penalty problem, in the sense that it is equivalent to the MINLP problem. The solutions of the penalty problem are obtained using a variant of the metaheuristic FA for global optimization. Numerical experiments are given on a set of benchmark problems aiming to analyze the quality of the obtained solutions and the convergence speed. We show that the firefly penalty-based algorithm compares favorably with the penalty algorithm when the deterministic DIRECT or the simulated annealing solvers are invoked, in terms of convergence speed.

Keywords: mixed-integer programming; firefly algorithm; penalty function;

AMS Subject Classification: 90C30; 90C26; 90C11; 90C56; 90C59

1. Introduction

This article aims to analyze the merit, performance-wise, of a penalty approach for globally solving mixed-integer nonlinear programming (MINLP) problems. A continuous relaxation of the MINLP problem is carried out by converting it to a finite sequence of nonlinear programming (NLP) problems with only continuous variables. The problem to be addressed has the form:

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

where f is a nonlinear continuous function. The study carried out in this article assumes that the compact set C is $C = \{x \in \mathbb{R}^n : l_i \leq x_i \leq u_i,\ i \in I\}$. Since $x_j \in \mathbb{Z}$

*Corresponding author. Email: mfc@math.uminho.pt

ISSN: 0233-1934 print/ISSN 1029-4945 online
© 2014 Taylor & Francis
DOI: 10.1080/0233193YYxxxxxxxx
http://www.informaworld.com

for $j \in I_d$, we define the feasible region of problem (1) as follows:

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

This way, $|I_c|$ is the number of continuous variables, $|I_d|$ gives the number of integer variables, and l and u are the vectors of the lower and upper bounds on the variables respectively.

The continuous relaxation of a MINLP is obtained by relaxing the integrality conditions from $x_j \in \mathbb{Z}$, $j \in I_d$ to $x_j \in \mathbb{R}$, $j \in I_d$. Many approaches using continuous relaxation have been devised to solve MINLP problems [35]. The most used and simple approach solves the continuous relaxation of the MINLP problem followed by rounding off the continuous values (of the integer variables) to their nearest integral values. This type of approach does not work well on problems with a large number of variables.

In general, the presence of integer variables implies that the feasible region W is not convex. We may distinguish convex from nonconvex MINLP problems. If the problem functions are 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. Surveys on integer and mixed-integer programming can be found in [14, 25].

Exact and heuristic methods have been proposed to solve MINLP problems. For convex MINLP, some quite effective exact methods that have been devised based on the convex property include branch-and-bound (BB) [10, 13], outer approximation [1], branch-and-reduce [36], branch-and-cut, generalized Benders decomposition, LP/NLP-based BB and hybrid methods [11, 14]. They are capable of solving large problems with hundreds of variables although the computational time required to reach a solution may grow exponentially. By contrast, the continuous relaxation of a nonconvex MINLP is itself a global optimization problem, and therefore likely to be NP-hard. Spatial BB has emerged from a combination of the standard BB approach and standard NLP methods, like for example sBB and BARON solvers. COUENNE (Convex Over- and Under-ENvelopes for Nonlinear Estimation) is a popular spatial BB solver for solving nonconvex MINLP [10]. The main issues addressed by the paper are related with bounds tightening and branching strategies. When solving nonconvex MINLP, most algorithms have difficulty in reaching a feasible point. This issue has been addressed in the literature by feasibility pumps [12, 19]. Recent derivative-free methods for locating a local minimizer of MINLP problems are presented in [2, 3, 9, 30, 31]. In the first paper, the generalized pattern search algorithm for linearly constrained (continuous) optimization was extended to mixed variable problems and the constraints are treated by the extreme barrier approach. In [3, 9], pattern search based methods are used to tackle continuous and discrete variables simultaneously. The approaches in [30, 31] use a discrete line search to deal with the discrete variables, although in [31] a simple penalty is used to handle the nonlinear constraints.

Recently, exact penalty approaches have also been extended to general nonlinear integer programming problems. Furthermore, it has been shown that a general class of penalty functions can be used to solve general MINLP problems [15, 32, 33, 35, 37].

When a global solution to a nonconvex NLP problem is required, global optimization solvers capable of converging to a global minimizer may be used, like for example, the exact methods BARON [33, 38], αBB [4] and DIRECT [21, 27, 29]. In the context of penalty function techniques for solving general constrained

NLP problems, a diversity of metaheuristic algorithms (like differential evolution, electromagnetism-like mechanism, genetic algorithm and artificial fish swarm algorithm) have been appearing in the literature [5, 7, 16, 17, 20]. Recently, a metaheuristic algorithm, termed as firefly algorithm (FA), that mimics the social behavior of fireflies based on the flashing and attraction characteristics of fireflies, has been developed for continuous optimization [40, 41]. Several variants of the firefly algorithm do already exist in the literature. Improvements have been made to FA aiming to accelerate convergence (see, for example, [28, 34, 42]). A Gaussian FA [22] and a hybrid FA with harmony search is proposed in [24]. Another modified FA is presented in [39], allowing the movement of the brightest firefly along a direction randomly chosen from a set of directions, provided that its brightness improves. A convergence analysis based on a parameter selection on FA is presented in [8]. A recent review of firefly algorithms is available in [23].

Our contribution in this article is directed to the area of continuous reformulation of bound constrained MINLP problems by relaxing the integrality conditions and adding suitable penalty terms to the objective function. Two penalty terms are proposed: one is based on the hyperbolic tangent function and the other on the inverse hyperbolic sine function. These proposals satisfy the required assumptions to prove that the bound constrained MINLP problem is equivalent to a continuous penalty relaxation, in the sense that they have the same global minimizers. Furthermore, for the NLP problems, a new fitness-based adaptive scheme is incorporated into the firefly movement equation, within the metaheuristic FA, to globally solve the continuous penalty problem. This new adaptive FA is less dependent on user provided parameters since only one control parameter is required to switch from global to local search.

The remaining part of this article is organized as follows. Section 2 describes the two proposed continuous penalty terms, including their properties and Section 3 reviews FA and presents the new ideas concerning the fitness-based adaptive firefly movement. Section 4 contains the results of all the numerical experiments and the conclusions are summarized in Section 5.

2. Penalty function technique

This section is concerned with the exact penalty approach that can be extended to solve MINLP problems. In this context, problem (1) is equivalent to the following continuous reformulation, which comes out by relaxing the integer constraints on the variables and adding a particular penalty term to the objective function:

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

where $\varepsilon \in (0, \bar{\varepsilon}]$ and $P(x;\varepsilon)$ is the penalty term, in the sense that they have the same global minimizers. The resulting penalty function in the NLP problem (3) is termed 'exact' since the two problems have the same global minimizers for a sufficiently small value of the penalty parameter ε [32, 33]. The following assumptions about f and the penalty P are considered.

Assumption 2.1 The function f is bounded on C and there exists an open set $A \supset W$ and positive real numbers p and L, such that for all $x, y \in A$, f satisfies the following condition

$$|f(x) - f(y)| \leq L\|x - y\|^p. \tag{4}$$

Assumption 2.2 For all $x, y \in W$ and for all $\varepsilon \in \mathbb{R}^+$,

$$P(x; \varepsilon) = P(y; \varepsilon). \tag{5}$$

Assumption 2.3 There exists an $\bar{\varepsilon}$, and for all $z \in W$ there exists a neighborhood $S(z)$ such that

$$P(x; \varepsilon) - P(z; \varepsilon) \geq \bar{L}\|x - z\|^p, \text{ for all } x \in S(z) \cap (C \backslash W) \text{ and } \varepsilon \in (0, \bar{\varepsilon}], \tag{6}$$

where $\bar{L} > L$ and p is chosen as in (4). Furthermore, let $S = \bigcup_{z \in W} S(z)$, $\exists \bar{x} \notin S$ such that

$$\lim_{\varepsilon \to 0} (P(\bar{x}; \varepsilon) - P(z; \varepsilon)) = +\infty \text{ for all } z \in W, \tag{7}$$

$$P(x; \varepsilon) \geq P(\bar{x}; \varepsilon) \text{ for all } x \in C \backslash S \text{ and for all } \varepsilon > 0. \tag{8}$$

Theorem 2.4 : *(Theorem 2.1 in [32]) Let Assumptions 2.1, 2.2 and 2.3 hold. Let W and C ($W \subseteq C \subset \mathbb{R}^n$) be compact sets. Let $\|\cdot\|$ be a suitable chosen norm. Then, $\exists \bar{\varepsilon} \in \mathbb{R}^+$ such that for all $\varepsilon \in (0, \bar{\varepsilon}]$, problems* (1) *and* (3) *have the same global minimizers.*

Proof : (See [32]) □

An example of the class of penalty terms that can be used to solve MINLP problems is the following [32, 33, 35]:

$$P(x; \varepsilon) = \sum_{j \in I_d} \min_{\substack{l_j \leq d_i \leq u_j \\ d_i \in \mathbb{Z}}} \log(|x_j - d_i| + \varepsilon). \tag{9}$$

In [32], assuming that f satisfies Assumption 2.1, it is shown that the penalty (9) satisfies Assumptions 2.2 and 2.3. Then Theorem 2.4 can be applied and it has been concluded that $\exists \bar{\varepsilon} > 0$ such that for any $\varepsilon \in (0, \bar{\varepsilon}]$, problem (1) and problem (3) based on penalty (9) have the same global minimizers.

Different penalty terms may be defined for problem (3), as long as some properties are maintained. Our proposals consider the *hyperbolic tangent* and the *inverse hyperbolic sine* functions. For these functions, we prove that Assumptions 2.2 and 2.3 hold.

2.1. *The hyperbolic tangent penalty*

The first newly proposed penalty term for solving MINLP via the equivalence between problems (1) and (3) is based on the hyperbolic tangent function, $\tanh(\cdot)$, which is differentiable and strictly increasing on the set C containing $[l, u]$:

$$P^t(x; \varepsilon) = \frac{1}{\varepsilon} \sum_{j \in I_d} \min_{\substack{l_j \leq d_i \leq u_j \\ d_i \in \mathbb{Z}}} \tanh(|x_j - d_i| + \varepsilon) \tag{10}$$

and $\varepsilon \in \mathbb{R}^+$ is the penalty parameter.

According to Theorem 2.4, we prove that penalty P^t can be used in problem (3).

Property 2.5 For the penalty term in (10), there exists $\bar{\varepsilon} > 0$ such that for any $\varepsilon \in (0, \bar{\varepsilon}]$, problems (1) and (3) have the same global minimizers.

The proof of this result is closely related to the proof given in [32] and is left for the Appendix A of the paper.

Figure 1 shows the behavior of the hyperbolic tangent penalty (10) as ε decreases (for $\varepsilon = 1, 0.5, 0.25, 0.125, 0.0625, 0.03125$). The figure also shows the penalties based on the 'log' penalty (P) in (9), for $\varepsilon = 1$ and $\varepsilon = 0.03125$. We observe that the penalty in (10) increases faster than the penalty 'log' as ε decreases and the integrality violation increases.

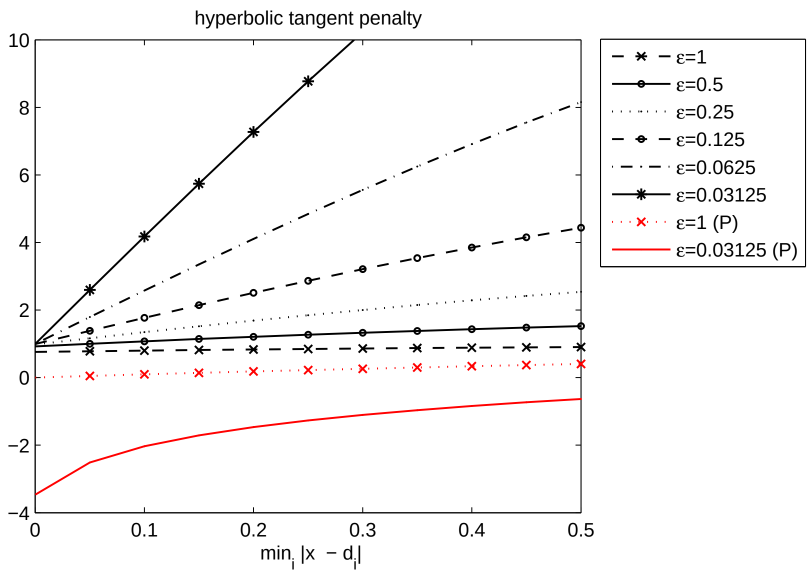

Figure 1.: Behavior of the 'tanh' penalty for six different values of ε

2.2. *The inverse hyperbolic sine penalty*

The inverse hyperbolic sine, asinh(t), is a differentiable function, for all real t, that is suitable to define the penalty term in the problem (3) when solving MINLP problems like (1). Function asinh(t) is nonnegative and strictly increasing on $[0, +\infty)$. The penalty term takes the form

$$P^a(x; \varepsilon) = \sum_{j \in I_d} \min_{\substack{l_j \le d_i \le u_j \\ d_i \in \mathbb{Z}}} \operatorname{asinh}\left(\frac{1}{\varepsilon}|x_j - d_i| + \varepsilon\right) \tag{11}$$

where $\varepsilon > 0$ is the penalty parameter. The behavior of the inverse hyperbolic sine penalty for $\varepsilon = 1, 0.5, 0.25, 0.125, 0.0625, 0.03125$ is shown in Figure 2. For comparison, the 'log' penalty (P) for $\varepsilon = 1$ and $\varepsilon = 0.03125$ is also displayed. In this case, as ε decreases and the integrality violation increases, the behavior of penalty (11) is somewhat similar to the 'log' penalty.

Once again, using Theorem 2.4 we can prove the following:

Property 2.6 For the penalty term in (11), there exists $\bar{\varepsilon} > 0$ such that for any $\varepsilon \in (0, \bar{\varepsilon}]$, problems (1) and (3) have the same global minimizers.

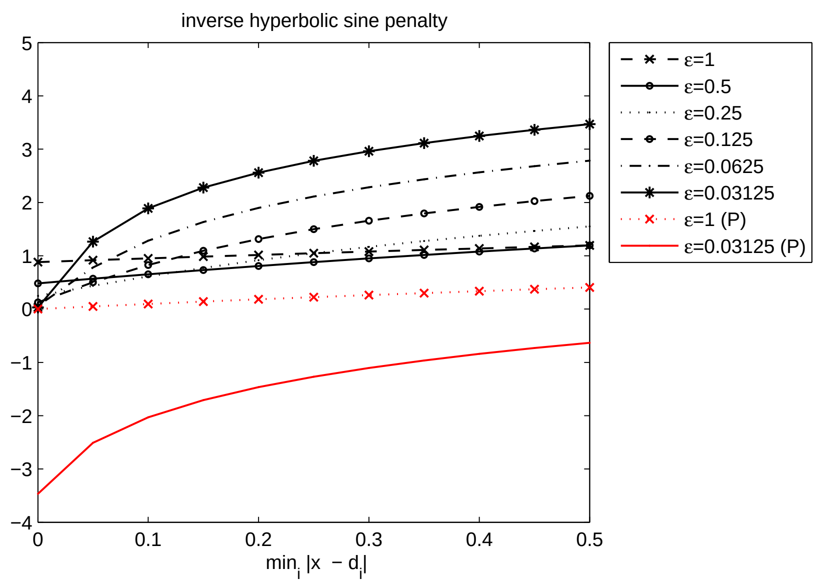

Figure 2.: Comparison of 'asinh' penalties for different ε values

The proof of this result is closely related to the proof given in [32] and is left for the Appendix A of the paper.

2.3. *The exact penalty algorithm*

We now briefly describe the exact penalty method aiming to find a global minimizer of a MINLP problem [33]. See Algorithm 1. In this article, we implement the exact penalty method with the newly proposed penalty terms (10) and (11), since they satisfy Assumptions 2.2 and 2.3, as shown in Subsections 2.1 and 2.2. To compute a $\delta^{(k)}$ approximation to the global minimizer of the penalty problem (3) that satisfies

$$\psi(x^{(k)};\varepsilon^{(k)}) \leq \psi(x;\varepsilon^{(k)}) + \delta^{(k)}, \text{ for all } x \in C \tag{12}$$

for fixed $\varepsilon^{(k)}$ and $\delta^{(k)}$, we propose the stochastic population-based strategy known as FA. The motivation and advantages for using FA (in Algorithm 2 below) are described in the next section.

The penalty parameter is updated $\varepsilon^{(k+1)} = \sigma_\varepsilon \varepsilon^{(k)}$, $\sigma_\varepsilon \in (0,1)$, when the computed approximate global minimizer $x^{(k)}$, of the penalty function $\psi(x;\varepsilon^{(k)})$, is not feasible for problem (1) and condition

$$\psi(x^{(k)};\varepsilon^{(k)}) - \psi(z^{(k)};\varepsilon^{(k)}) \leq L(\varepsilon^{(k)})\|x^{(k)} - z^{(k)}\|^p \tag{13}$$

holds, where $z^{(k)} \in W$ is the point that minimizes the distance between $x^{(k)}$ and the set $S(z^{(k)}) = \{x \in \mathbb{R}^n : \|x - z^{(k)}\|_\infty < \rho\}$, for a sufficiently small positive ρ, i.e., $z_j^{(k)} \in \mathbb{Z}, j \in I_d$ results from rounding $x_j^{(k)}$ to the nearest integer, and $z_i^{(k)} = x_i^{(k)}, i \in I_c$ (see Proposition 2 in [33]). On the other hand, the parameter $\varepsilon^{(k)}$ remains unchanged when $x^{(k)}$ is feasible or when $x^{(k)}$ is infeasible but condition (13) is not satisfied.

```
Data: $f^*$ (or $k_{\max}$), $\delta_{\min} \ll 1$, $\varepsilon^{(1)} > 0$, $\delta^{(1)} > 0$, $\sigma_\varepsilon \in (0,1)$, $\sigma_\delta \in (0,1)$, $p$
Set $k = 1$;
Randomly generate $x^{(0)} \in C$;
while $x^{(k-1)} \notin W$ or $f(x^{(k-1)}) > f^* + \delta_{\min}$ do
    Given $x^{(k-1)}$, compute an approximate global minimizer $x^{(k)}$ of
    problem (3) using Algorithm 2 such that
        $\psi(x^{(k)};\varepsilon^{(k)}) \le \psi(x;\varepsilon^{(k)}) + \delta^{(k)}$,   for all $x \in C$
    if $x^{(k)} \notin W$ and $\psi(x^{(k)};\varepsilon^{(k)}) - \psi(z^{(k)};\varepsilon^{(k)}) \le L(\varepsilon^{(k)})\|x^{(k)} - z^{(k)}\|^p$ then
        Set $\varepsilon^{(k+1)} = \sigma_\varepsilon \varepsilon^{(k)}$, $\delta^{(k+1)} = \delta^{(k)}$;
    else
        Set $\varepsilon^{(k+1)} = \varepsilon^{(k)}$, $\delta^{(k+1)} = \sigma_\delta \delta^{(k)}$;
    end
    Set $k = k + 1$
end
```

Algorithm 1: Penalty algorithm

In our implementation of the penalty algorithm we set the function $L(\varepsilon^{(k)})$ to depend on the selected penalty term ('tanh'-based in (10) or 'asinh'-based in (11)) using the upper bounds defined for $\bar{L}$ in (A1) or (A4) respectively, and guaranteeing that $L(\varepsilon^{(k)})$ is positive for $\varepsilon^{(k)} > 0$ and $L(\varepsilon^{(k)}) \to 0$ when $\varepsilon^{(k)} \to 0$.

The algorithm terminates when a feasible solution is found within $\delta_{\min}$ of the known global optimum solution. However, if the optimal value is not known, the algorithm may use another condition, for example, one based on the number of iterations, $k > k_{\max}$, for a given threshold value $k_{\max}$ (or on the total number of function evaluations, $nf > nf_{\max}$, where $nf_{\max}$ is the threshold value).

The convergence results presented in [33] for a general equivalence framework, where a general constrained NLP problem is considered equivalent to a sequence of penalty problems by adding a suitable penalty term to the objective function (in the sense that they have the same global minimizers), have been extended to the exact penalty method with the penalty term (9), for solving a MINLP problem via an NLP continuous reformulation.

Based on the results reported in [33], we now state the main convergence results of Algorithm 1.

We assume that the sequence of iterates $\{x^{(k)}\}$ is well defined, i.e., the approximate global minimizer $x^{(k)}$ of the penalty function $\psi(\cdot)$ that satisfies condition (12) can be found. Since we use the metaheuristic FA to compute the approximate global minimizer, there is no theoretical guarantee that the approximate solution can be found in finite time. Relying on the probability theory a stochastic convergence may be established for FA. This type of convergence is different from the convergence concept of classical analysis. The probabilistic version of pointwise convergence from elementary real analysis is the convergence with probability one. This means that, in the limit, the convergence of FA to an approximate minimizer of the penalty function will be guaranteed with probability one. This issue will be addressed in a future study.

Lemma 2.7: *Let $\{x^{(k)}\}$ be the sequence of iterates produced by Algorithm 1. Then either*

- *$\exists \bar{k}$ such that $\varepsilon^{(k)} = \bar{\varepsilon}$, for any $k \ge \bar{k}$, and every accumulation point $\bar{x}$ of the sequence $\{x^{(k)}\}$ is feasible ($\bar{x} \in W$); or*
- *$\varepsilon^{(k)} \to 0$ and every accumulation point $\bar{x}$ of a subsequence $\{x^{(k_i)}\}$ is feasible,*

with k_i belonging to the set of indices in which condition (13) *is satisfied;*

holds.

The following lemma states that the parameter ε is updated only a finite number of times.

Lemma 2.8: *Let $\{x^{(k)}\}$ and $\{\varepsilon^{(k)}\}$ be sequences produced by Algorithm 1. Then $\exists \bar{k}$ such that $\varepsilon^{(k)} = \bar{\varepsilon}$, for any $k \geq \bar{k}$.*

The following result is a consequence of Lemmas 2.7 and 2.8 and $\delta^{(k)} \to 0$.

Theorem 2.9: *Every accumulation point $\bar{x}$ of the sequence of iterates $\{x^{(k)}\}$ produced by Algorithm 1 is a global minimizer of the problem* (1).

Since our penalty term proposals satisfy Assumptions 2.2 and 2.3, the proofs of Lemmas 2.7 and 2.8 and Theorem 2.9 follow the same arguments used in the proofs of Lemma 1, Lemma 2 and Theorem 2 in [33].

3. Solving NLP continuous problems

Firefly algorithm is a bio-inspired metaheuristic algorithm that is capable of converging to a global solution of an optimization problem. It is inspired by the flashing behavior of fireflies at night. According to [39–41], the three main rules used to construct FA are the following:

- all fireflies are unisex, meaning that any firefly can be attracted to any other brighter one;
- the brightness of a firefly is determined from the encoded objective function;
- attractiveness is directly proportional to brightness but decreases with distance.

The performance of any metaheuristic depends on the diversification (or exploration) and intensification (or exploitation) features of the algorithm. Diversification aims to explore the search space and to compute a diversity of solutions. The exploration aspect of a metaheuristic relies on randomization. Intensification aims to locally exploit a region identified as a promising one. FA is a population-based metaheuristic that does not require any derivative information and its performance does not depend on the properties of the objective function. The balance between local and global searches is crucial for the success of FA. Both searches emerge from the equation of the firefly movement (see equation (14) below), in particular, they depend on how the control parameters vary. Furthermore, the performance of the algorithm is only moderately affected by the size of the population of fireflies. The algorithm is not dependent on large population sizes to be able to converge to high quality solutions.

3.1. *Classical FA*

We use the point $x = (x_1, x_2, \dots, x_n)^T$ to represent the location of a firefly in the search space C and the position of the jth point of a population of m points is represented by $x^j \in \mathbb{R}^n$. In the classical FA, a firefly i is attracted to another more brighter firefly j, and the movement may be determined by:

$$x^i = x^i + \beta(x^j - x^i) + \alpha \mathcal{L}(0,1)\sigma^i/2, \tag{14}$$

where the second term on the right hand side of (14) is due to the attraction while the third term is due to randomization with $\alpha \in (0,1)$ being the randomization parameter. Here, $\mathcal{L}(0,1)$ is a random number from the standard Lévy distribution (with a location parameter equals to 0, a scale parameter equals to 1, and the shape parameter equals to 0.5), and the vector σ^i gives the variation of x^i relative to the position of the best firefly, x^1, as follows:

$$\sigma^i = \left(|x_1^i - x_1^1|, \ldots, |x_n^i - x_n^1|\right)^T .$$

The parameter β gives the attractiveness of a firefly

$$\beta = \beta_0 \exp\left(-\gamma \|x^i - x^j\|^2\right), \tag{15}$$

and varies with the light intensity seen by adjacent fireflies and the distance between themselves. β_0 is the attraction parameter when the distance is zero. The parameter γ characterizes the variation of the attractiveness, and is crucial to speed the convergence of the algorithm. In theory, γ could take any value in the set $[0,\infty)$. When $\gamma \to 0$, the attractiveness is constant $\beta = \beta_0$ and a flashing firefly can be seen anywhere in the domain. When $\gamma \to \infty$ the attractiveness is almost zero in the sight of other fireflies or the fireflies are short-sighted. In particular, when $\beta_0 = 0$, the algorithm behaves like a simple random walk. In summary, the three control parameters of FA are: the randomization parameter α, the attractiveness β, and the absorption coefficient γ [23, 43]. In the herein presented implementation of FA, we use previously proposed updates for α and γ that decrease with the iteration counter, k^{FA}, of the algorithm [18]. Thus, α is allowed to decrease linearly with k^{FA}, from an upper level $\alpha_{\max}$ to a lower level $\alpha_{\min}$:

$$\alpha = \alpha_{\max} - k^{FA} \frac{\alpha_{\max} - \alpha_{\min}}{k_{\max}^{FA}}, \tag{16}$$

where $k_{\max}^{FA}$ is the threshold number of allowed iterations. In order to make the attraction term on the right hand side of (14) to dominate the movement mostly at the end of the iterative process, γ is also decreased with k^{FA} from $\gamma_{\max}$ to $\gamma_{\min}$, as follows:

$$\gamma = \gamma_{\max} \exp\left(\frac{k^{FA}}{k_{\max}^{FA}} \ln\left(\frac{\gamma_{\min}}{\gamma_{\max}}\right)\right). \tag{17}$$

Algorithm 2 presents the main steps of a classical FA. We note that the best point of the population is the position of the brightest firefly. When fireflies are ranked according to their fitness, i.e., according to the objective value $\psi^k(\cdot) \equiv \psi(\cdot; \varepsilon^{(k)})$, the best firefly is x^1.

```
Data: $k^{FA}_{\max}$, $f^*$, and from Algorithm 1: $x^1 = x^{(k-1)}$, $\varepsilon^{(k)}$ and $\delta^{(k)}$.
Set $k^{FA} = 1$;
Randomly generate $x^i$, $i = 2, \dots, m$;
Compute $\psi^k(x^i) \equiv \psi(x^i; \varepsilon^{(k)}), i = 1, \dots, m$, rank fireflies (from lowest to largest $\psi$);
while $k^{FA} \le k^{FA}_{\max}$ and $\psi^k(x^1) > f^* + \delta^{(k)}$ do
    forall $x^i, i = 2, \dots, m$ do
        forall $x^j$, $j < i$ do
            if $\psi^k(x^j) < \psi^k(x^i)$ then
                Move firefly $i$ towards $j$ using (14);
                Compute $\psi^k(x^i)$;
            end
        end
    end
    Compute $\psi^k(x^i), i = 1, \dots, m$, rank fireflies (from lowest to largest $\psi$);
    Set $k^{FA} = k^{FA} + 1$;
end
```

Algorithm 2: Classical FA

3.2. *Adaptive FA*

An opportune parameter choice is fundamental to increase the overall efficiency of the algorithm. Here we propose an adaptive FA in the sense that the local search activity relies on the relative variation in fitness between the firefly i and each of the other brighter fireflies. The brighter is the firefly (relative to firefly i) the higher is the attraction, i.e., when $\psi^k(x^j) < \psi^k(x^l) < \psi^k(x^i)$, the attractiveness is higher when firefly i moves towards firefly j than towards firefly l, being equal to one when the movement is towards the brightest firefly x^1. In the context of solving the NLP continuous problem (3), the movement that results from firefly i be attracted to the brighter firefly j is defined by

$$x^i = x^i + \exp(-\alpha) \frac{\psi^k(x^i) - \psi^k(x^j)}{\psi^k(x^i) - \psi^k(x^1)} (x^j - x^i) + \alpha \mathcal{L}(0,1) \sigma^i / 2, \tag{18}$$

where the attractiveness depends on the variation in fitness ψ between the firefly i and the brighter firefly j, relative to the difference between firefly i and the brightest firefly of all. Thus, it has the ability to easily adapt to problem landscape. The randomization term is similar to (14). We note here that the second term of equation (18) is scaled by the parameter $\exp(-\alpha)$ that aims to enforce the second term dominance at the end of the iterative process, while the third term dominates at the beginning of the iterations (see the above referred updating scheme (16)). The adaptive FA possesses good exploring as well as exploiting capabilities. Unlike the classical and other variants of FA, the adaptive FA has just one control parameter.

4. Numerical experiments

Several sets of experiments were carried out aiming to compare the convergence speed and the solution quality produced by the algorithm when tested with both penalty terms $P^t(\cdot)$ in (10) and $P^a(\cdot)$ in (11). Convergence speed is measured comparing the number of function evaluations needed to reach a certain threshold

Table 1.: Comparison of 'FA' and 'adaptive FA'.

		$\lvert f_{\text{best}} - f^*\rvert$ using penalty (10)		$\lvert f_{\text{best}} - f^*\rvert$ using penalty (11)	
Prob.	f^*	'FA'	'adaptive FA'	'FA'	'adaptive FA'
ACK_5	0	8.882e-16	8.882e-16	8.882e-16	8.882e-16
AP	-0.3523	8.603e-05	8.603e-05	8.604e-05	8.607e-05
Bea	0	1.573e-18	2.444e-21	1.034e-18	6.296e-20
BL	0	0.000e+00	0.000e+00	0.000e+00	0.000e+00
BF1	0	0.000e+00	0.000e+00	0.000e+00	0.000e+00
Buk	0	0.000e+00	0.000e+00	0.000e+00	0.000e+00
DA	-24777	4.831e-01	4.817e-01	4.817e-01	4.817e-01
DP_2	0	1.583e-20	1.680e-19	6.073e-19	3.931e-18
DP_4	0	5.064e-17	3.242e-17	2.371e-18	5.089e-19
Him	0	0.000e+00	0.000e+00	0.000e+00	0.000e+00
LM2_5	0	1.146e-31	1.500e-32	1.701e-27	2.122e-27
NF2	0	2.038e-04	0.000e+00	0.000e+00	0.000e+00
RG_5	0	0.000e+00	0.000e+00	0.000e+00	0.000e+00
S10	-10.5319	4.384e-03	4.384e-03	4.384e-03	4.384e-03
SS_5	0	2.581e-17	8.969e-31	0.000e+00	2.782e-26

of objective function value (available in literature). Solution quality is measured in terms of difference between the obtained result and the best known optimal value, f^*. The numerical experiments were carried out on a PC Intel Core 2 Duo Processor E7500 with 2.9GHz and 4Gb of memory RAM. The algorithms were coded in Matlab Version 8.1 (R2013a).

4.1. *Experimental setting*

During most experiments the size of the population in the 'FA' and 'adaptive FA' is made to depend on the problem's dimension and is set to $m = \min\{5n, 50\}$. Twenty-two instances of 14 MINLP problems are used for benchmarking. Seven problems have $n = 2$ variables, two have $n = 4$ variables, problem Dixon-Price is tested with $n = 2$ and $n = 4$, problem Sum Squares is tested with $n = 5$ and $n = 10$, and problems Ackley, Levy and Rastrigin are tested with $n = 5, 10$ and 20. See the Appendix B for the full description of the problems and their acronyms. The parameters in Algorithm 1 are set as follows: $\varepsilon^{(1)} = 10$, $\delta^{(1)} = 1$, $\delta_{\min}$=1e-04, $\sigma_\varepsilon = 0.1$, $\sigma_\delta = 0.1$, $p = 1$, $k_{\max} = 20$. On the other hand, in Algorithm 2 we set: $k^{FA}_{\max} = 100$, $\alpha_{\max} = 0.5$, $\alpha_{\min} = 0.001$, $\gamma_{\max} = 10$, $\gamma_{\min} = 0.001$, and $\beta_0 = 1$.

4.2. *Comparison between 'FA' and 'adaptive FA'*

First, a comparison between 'FA' and the herein proposed 'adaptive FA', based on the solution quality is presented. Algorithm 1 is terminated when the number of iterations exceeds $k_{\max}$. Table 1 shows the difference between the objective function value of the best run (out of 10 independent runs) and the known global solution, $\lvert f_{\text{best}} - f^*\rvert$. Both penalty terms (10) and (11) are tested. The results correspond to instances of the 14 problems with small dimensions. We conclude that 'adaptive FA' performs slightly better than 'FA' and the penalty algorithm associated with the hyperbolic tangent function, described in (10), produces higher quality solutions on 8 cases against 6 of the penalty (11) (among the 30 results of both variants, where 16 are ties).

Table 2.: Population size effect, using 'adaptive FA'.

		$\|f_{\text{best}} - f^*\|$ of penalty (10)			$\|f_{\text{best}} - f^*\|$ of penalty (11)		
Prob.	f^*	$m = 10$	$m = 25$	$m = 50$	$m = 10$	$m = 25$	$m = 50$
ACK_5	0	2.850e-09	8.882e-16	8.882e-16	1.712e-10	8.882e-16	4.441e-15
LM2_5	0	2.935e-17	1.500e-32	1.500e-32	1.796e-17	2.122e-27	2.000e-27
RG_5	0	2.842e-13	0.000e+00	0.000e+00	4.320e-12	0.000e+00	0.000e+00
SS_5	0	7.436e-19	8.969e-31	1.409e-17	9.307e-17	2.782e-26	4.084e-27

4.3. *Population size effect on performance*

In this section, we present some results concerning with the effect on the algorithm performance of the size of the population of fireflies. We use four instances with $n = 5$ and test three values of m: $2n$, $5n$ and $10n$. Table 2 reports on $|f_{\text{best}} - f^*|$. Both proposed penalties (10) and (11) are tested. Based on the results we may conclude that the quality is higher with larger population sizes, although the difference between the use of $m = 5n$ and $m = 10n$ is almost not noticeable.

4.4. *Other experimental tests*

Finally, we use all the instances above mentioned and analyze the performance of the exact penalty technique in terms of the quality of the obtained solutions. Table 3 shows the difference $|f_{\text{best}} - f^*|$ obtained after running each instance 10 times with each tested penalty. For these experiments, the penalty algorithm is terminated solely with the condition $k > k_{\max}$ with $k_{\max} = 20$. We also note that the population size is again set to $m = \min\{5n, 50\}$. For comparison, we present the results for the penalties (10), (11) and (9). We also aim to compare the performance of the algorithm when using different values for the initial $\varepsilon^{(1)}$: 10 and 100. Since larger dimensional instances are now included in the test set, we allow the 'adaptive FA' to run for a maximum of 50 iterations.

Apart four instances, BL, BF1, Buk and Him, that perform evenly and extremely well with the three tested penalties, we observe that the quality of the obtained solutions is higher with the penalty (10) with 42% of the tested cases (considering the total of 26 sets of runs with different $|f_{\text{best}} - f^*|$ values, out of 44 sets), than with the penalties (11) (with 35% of the cases) and (9) (with only 23% of the cases). We also observe that the quality is higher when $\varepsilon^{(1)} = 10$ is used (71% against 29%, among 14 instances with different $|f_{\text{best}} - f^*|$ values) with penalty (11) and (59% against 41% among 17 instances) with penalty (9), and it is a tie when penalty (10) is used.

On the other hand, Table 4 shows the average number of function evaluations, 'nf_{avg}', and the standard deviation of function evaluations, 'St.D.', required to reach a feasible solution within $\delta_{\min}$ of the known global optimum solution, i.e., the algorithm terminates when both conditions $x^{(k)} \in W$ and $f(x^{(k)}) \leq f^* + \delta_{\min}$ (for $\delta_{\min}$ =1.0e-04) are met. During these experiments, to check if $x^{(k)} \in W$, where $z^{(k)}$ is defined as in (13), we use $\|x^{(k)} - z^{(k)}\|_\infty \leq \sigma_{\min}$, where the tolerance error $\sigma_{\min}$ is set to 1.0e-05.

The goal now is to analyze the convergence speed of the Algorithm 1 when using the proposed penalties and $\varepsilon^{(1)} = 100$. For these experiments we also set $k^{FA}_{\max} = 50$. We note that the average number of function evaluations is computed only from the runs that terminated with the conditions $x^{(k)} \in W$ and $f(x^{(k)}) \leq f^* + \delta_{\min}$. These are considered successful runs and the table reports the percentage of successful runs, 'SR' (%). In the table, the character '–' means that only one run in 10 was successful, thus nf_{avg} is the number of function evaluations of that run and St.D.

Table 3.: Reports on solution quality $|f_{\text{best}} - f^*|$, using $k_{\max} = 20$.

		penalty (10)		penalty (11)		penalty (9)	
Prob.	f^*	$\varepsilon^{(1)} = 10$	$\varepsilon^{(1)} = 100$	$\varepsilon^{(1)} = 10$	$\varepsilon^{(1)} = 100$	$\varepsilon^{(1)} = 10$	$\varepsilon^{(1)} = 100$
ACK_5	0	8.882e-16	8.882e-16	8.882e-16	8.882e-16	7.994e-15	4.441e-15
ACK_10	0	1.155e-14	2.220e-14	3.286e-14	4.707e-14	1.998e-13	2.212e-13
ACK_20	0	2.049e+00	2.027e+00	1.884e+00	1.425e+00	2.639e+00	2.026e+00
AP	-0.3523	8.606e-05	8.603e-05	8.605e-05	8.601e-05	8.603e-05	8.607e-05
Bea	0	4.933e-18	3.896e-20	7.654e-19	1.057e-17	2.776e-19	2.415e-18
BL	0	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00
BF1	0	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00
Buk	0	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00
DA	-24777	4.817e-01	4.817e-01	4.817e-01	4.817e-01	4.817e-01	4.817e-01
DP_2	0	1.392e-19	2.939e-19	1.019e-19	9.919e-18	3.001e-18	7.392e-18
DP_4	0	2.159e-18	1.029e-18	2.778e-21	5.908e-19	1.762e-16	4.226e-17
Him	0	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00
LM2_5	0	1.500e-32	4.134e-31	1.500e-32	3.215e-27	1.184e-28	2.815e-26
LM2_10	0	7.228e-29	3.757e-26	2.412e-25	6.729e-25	4.323e-28	1.141e-26
LM_20	0	1.164e-12	9.289e-14	7.292e-13	9.477e-14	2.051e-12	3.806e-16
NF2	0	1.803e-03	0.000e+00	7.557e-02	0.000e+00	8.109e-02	0.000e+00
RG_5	0	0.000e+00	0.000e+00	0.000e+00	0.000e+00	9.468e-09	0.000e+00
RG_10	0	2.985e+00	4.033e+00	3.980e+00	4.975e+00	4.975e+00	5.586e+00
RG_20	0	8.958e+00	7.961e+00	8.959e+00	1.095e+01	7.790e+00	1.112e+01
S10	-10.5319	4.384e-03	4.384e-03	4.384e-03	4.384e-03	4.423e-03	4.384e-03
SS_5	0	2.821e-30	1.440e-18	1.077e-26	1.963e-26	2.246e-28	1.200e-25
SS_10	0	5.327e-27	4.490e-18	5.393e-25	4.904e-20	1.110e-25	1.951e-24

does not apply. The instances where the algorithm was not able to reach a feasible solution within the specified error tolerance, in any run, are identified with '0' in 'SR'. 'Emphasized' values in parentheses correspond to the objective function value of the best run (in the column of the average number of function evaluations) and a measure of the proximity of the obtained solution $x^{(k)}$ to the known global solution x^* (in the column of St.D.).

From the results we may conclude that the herein proposed penalties (10) and (11) have comparable convergence behavior. The penalty (10) solves 59% of tested instances with 90–100% of successful runs (and 73% of instances with 80–100% of successful runs) and has 14% of instances with only unsuccessful runs, and the penalty (11) successfully solves 68% of instances with 90–100% of successful runs (and 73% of instances with 80–100% of successful runs), but has 18% of instances with only unsuccessful runs. On the other hand, the penalty (9) used in [32, 33] is successful in 90–100% of the runs when solving 64% of the instances (and is successful in 80–100% of the runs with 68% of the instances), and produces only unsuccessful runs on 23% of the instances. We note that the penalty algorithm associated with the function (10) performs slightly better than with the other two cases in comparison, with an overall average percentage of successful runs of 75%, against 74% of the function (11) and 72% of penalty (9).

We now use two well-known global search techniques available in the literature to solve the NLP continuous problems (3) in this exact penalty context. The first is the deterministic and exact global search procedure 'DIRECT' [27]; the second is a point-by-point stochastic global algorithm known as 'simulated annealing' (SA) [26]. The function 'simulannealbnd' from the Global Optimization Toolbox$^{\text{TM}}$ of Matlab is invoked.

The results for comparison are reported in Table 5. The penalty algorithm is terminated with the conditions

$$x^{(k)} \in W \ \text{ and } \ f(x^{(k)}) \le f^* + \ 1.0\text{e-}03 \tag{19}$$

where the tolerance error $\sigma_{\min}$ is now set to 1.0e-03, and $f(x^{(k)})$ is the solution

Table 4.: Reports on convergence speed, using $\delta_{\min}$ =1.0e-04 and $\sigma_{\min}$ =1.0e-05.

	penalty (10)			penalty (11)			penalty (9)		
Prob.	nf_{avg}	St.D.	SR	nf_{avg}	St.D.	SR	nf_{avg}	St.D.	SR
ACK_5	3.585e+04	6.860e+03	100	4.417e+04	1.360e+04	90	3.740e+04	3.785e+03	90
ACK_10	2.429e+05	1.449e+04	60	2.537e+05	4.038e+04	70	2.471e+05	5.254e+04	50
ACK_20	6.808e+05	–	10	(*1.9e+00*)	(*1.0e+00*)	0	(*1.8e+00*)	(*9.1e-01*)	0
AP	7.769e+03	1.596e+04	80	3.624e+03	2.881e+04	90	4.922e+03	6.051e+03	100
Bea	4.668e+03	4.316e+03	100	3.040e+03	1.972e+03	100	2.742e+03	1.533e+03	100
BL	3.958e+03	2.232e+03	100	4.368e+03	2.107e+03	100	3.762e+03	1.295e+03	100
BF1	3.585e+03	1.741e+03	100	4.476e+03	1.659e+03	100	3.777e+03	1.253e+03	100
Buk	1.202e+04	1.682e+04	100	1.305e+04	1.127e+04	100	6.092e+03	6.206e+03	100
DA	(*-2.5e+04*)	(*0.0e+00*)	0	(*-2.5e+04*)	(*0.0e+00*)	0	(*-2.5e+04*)	(*0.0e+00*)	0
DP_2	1.044e+04	1.854e+04	100	3.055e+03	2.264e+04	90	2.594e+03	1.108e+03	100
DP_4	1.226e+04	3.906e+03	100	1.410e+04	3.060e+03	90	1.059e+04	1.595e+04	70
Him	6.280e+03	5.827e+03	100	6.233e+03	2.557e+03	100	4.284e+03	1.574e+03	100
LM2_5	3.483e+04	9.766e+03	100	3.265e+04	4.510e+03	100	2.759e+04	5.135e+03	100
LM2_10	2.077e+05	2.556e+04	100	1.947e+05	2.139e+04	100	2.274e+05	2.818e+04	100
LM_20	5.856e+05	5.003e+04	80	5.670e+05	9.963e+04	90	8.435e+05	2.145e+04	80
NF2	9.558e+04	1.162e+05	100	5.870e+04	1.128e+05	90	2.832e+04	2.765e+04	100
RG_5	2.026e+05	2.147e+05	50	1.333e+05	1.467e+05	40	(*9.9e-01*)	(*9.9e-01*)	0
RG_10	(*2.0e+00*)	(*1.0e+00*)	0	(*1.0e+00*)	(*1.0e+00*)	0	(*4.0e+00*)	(*9.9e-01*)	0
RG_20	(*1.1e+01*)	(*1.0e+00*)	0	(*8.3e+00*)	(*1.0e+00*)	0	(*8.0e+00*)	(*1.0e+00*)	0
S10	3.954e+04	1.632e+05	80	4.345e+04	2.888e+04	80	2.447e+04	9.547e+03	100
SS_5	2.743e+04	5.375e+03	100	3.234e+04	6.598e+03	100	2.960e+04	6.394e+03	100
SS_10	1.962e+05	2.550e+04	100	2.155e+05	3.197e+04	100	2.025e+05	2.288e+04	100

reported by ‘DIRECT’ and by ‘SA’, and is the best solution (among m) obtained by our proposed ‘adaptive FA’. For these tests, the initial value for $\varepsilon^{(1)}$ is set to 100. While we use ‘nf_{avg}’ and ‘St.D.’ as criteria to analyze the convergence speed of our ‘adaptive FA’ and ‘SA’, only the number of function evaluations, ‘nf’, is required as convergence criterion for ‘DIRECT’, as long as the obtained solution satisfies conditions (19), identified in the table with ‘1’ in the columns marked with ‘Suc’. On the other hand, if one of the conditions in (19) is not satisfied, the penalty algorithm stops after 50 iterations, the run is identified with ‘0’ (unsuccessful run) and the best reported solution is shown ‘emphasized’ inside parentheses.

Table 5 also displays the CPU time, ‘T’, in seconds, required to achieve the reported solution. For the proposed ‘adaptive FA’ and for ‘SA’, the times correspond to the averaged values registered during the successful runs, ‘T_{avg}’ (among 10 runs), or the time of the best run when all of them are unsuccessful. All the other statistics concerned with the ‘adaptive FA’ have the same meaning as in the previous table.

For a fair comparison between ‘adaptive FA’, ‘DIRECT’ and ‘SA’ (when invoked in the penalty algorithm context) we use the same stopping criterion. Thus, each algorithm is terminated when the number of penalty function evaluations exceeds a threshold value, $npenf_{\max}$, set to 5000. We also test ‘DIRECT’ with the stopping parameter values defined by default. We also run two sets of experiments with the ‘SA’: the first set stops the solver when 5000 function evaluations are reached; the second uses the stopping criteria and parameters defined by default in Matlab.

We observe that the penalty strategy based on the ‘adaptive FA’ is capable of solving (i.e., conditions (19) are met for at least one run) 82% of the instances, and has 90–100% of successful runs in 73% of the instances. In general, the unsuccessful runs occur when solving the larger dimensional problems. The total time required to solve the entire set of tested problems is 6.33e+02 seconds.

When the solver ‘DIRECT’ is used to solve the bound constrained NLP continuous penalty problems, we obtain different convergence speeds. The version with the setting $npenf_{\max}$=5000 is able to successfully solve 55% of the instances (Suc = ‘1’ in the table), while the version with the default parameter values solves only 32%.

Table 5.: Comparison of 'nf_{avg}', 'nf', 'SR', 'Suc', 'T_{avg}' and 'T', using $\delta_{\min}$ =1.0e-03 and $\sigma_{\min}$ =1.0e-03, and penalty (10).

	'adaptive FA'[(a)]				'DIRECT'[(a)]			'DIRECT' (default[†])			'SA'[(a)]				'SA' (default[‡])			
Prob.	nf_{avg}	St.D.	T_{avg}[♯]	SR	nf	T[§]	Suc	nf	T[§]	Suc	nf_{avg}	St.D.	T_{avg}[♯]	SR	nf_{avg}	St.D.	T_{avg}[♯]	SR
ACK_5	1.865e+04	2.511e+03	3.6e+00	90	4.849e+03	1.1e+00	1	1.300e+01	2.1e-02	1	(*5.6e-02*)	(*1.7e-02*)	7.6e+01	0	(*1.7e+00*)	(*9.6e-01*)	7.6e+01	0
ACK_10	1.088e+05	1.721e+04	3.3e+01	80	5.147e+03	1.7e+00	1	2.300e+01	8.1e-03	1	(*7.9e+00*)	(*3.0e+00*)	1.6e+02	0	(*8.2e+00*)	(*3.0e+00*)	1.8e+02	0
ACK_20	(*3.6e+00*)	(*1.5e-03*)	1.4e+02	0	5.775e+03	3.0e+00	1	4.300e+01	2.2e-02	1	(*7.6e+01*)	(*8.3e+00*)	2.1e+02	0	(*1.5e+01*)	(*8.4e+00*)	5.0e+02	0
AP	8.541e+03	5.341e+03	7.8e-01	100	6.810e+02	9.9e-02	1	(*0.0e+00*)	1.8e-01	0	7.227e+03	4.831e+03	3.5e+00	100	1.094e+04	9.296e+03	5.3e+00	100
Bea	9.552e+03	4.998e+03	8.7e-01	100	1.029e+03	1.7e-01	1	(*6.8e-01*)	1.9e-01	0	2.253e+04	2.491e+04	1.1e+01	100	1.086e+04	7.553e+03	5.3e+00	100
BL	8.041e+03	4.232e+03	9.2e-01	100	2.663e+03	3.9e-01	1	(*3.1e+00*)	2.7e-01	0	5.433e+03	2.870e+03	2.7e+00	100	7.651e+03	4.375e+03	3.8e+00	100
BF1	9.544e+03	5.525e+03	1.1e+00	100	1.247e+03	2.0e-01	1	1.500e+01	4.0e-03	1	3.300e+03	2.232e+03	1.7e+00	100	2.364e+03	1.019e+03	1.2e+00	100
Buk	4.121e+04	5.279e+04	4.8e+00	100	(*1.0e+02*)	2.0e+00	0	(*1.0e+02*)	2.2e-01	0	2.359e+04	1.386e+04	1.2e+01	30	2.049e+04	1.014e+04	1.0e+01	80
DA	(*-2.5e+04*)	(*0.0e+00*)	3.0e+01	0	(*-2.5e+04*)	2.2e+00	0	(*-2.1e+04*)	2.0e-01	0	0.000e+00	0.000e+00	0.0e+00	0	1.307e+04	2.507e+03	1.3e+00	100
DP_2	1.106e+04	6.187e+03	1.1e+00	100	6.630e+02	9.6e-02	1	(*1.0e+00*)	1.7e-01	0	6.348e+03	5.685e+03	3.1e+00	90	5.253e+03	2.275e+03	2.5e+00	100
DP_4	2.919e+04	4.331e+04	2.9e+00	90	(*3.8e-03*)	6.7e+00	0	(*1.0e+00*)	2.2e-01	0	7.510e+04	1.906e+04	3.6e+01	40	5.335e+04	2.000e+04	2.6e+01	30
Him	2.713e+04	2.553e+04	3.1e+00	100	2.450e+02	4.9e-02	1	(*1.3e+01*)	1.7e-01	0	4.113e+03	1.479e+03	2.0e+00	100	4.254e+03	2.854e+03	2.1e+00	100
LM2_5	1.527e+04	2.445e+03	2.9e+00	100	(*5.1e-06*)	1.1e+01	0	(*9.9e-01*)	2.0e-01	0	(*2.4e-03*)	(*1.8e-01*)	7.6e+01	0	(*2.8e-03*)	(*9.2e-02*)	7.4e+01	0
LM2_10	6.428e+04	1.230e+04	2.0e+01	100	(*8.3e-03*)	3.5e+01	0	(*1.4e+00*)	3.9e-01	0	(*6.3e-01*)	(*1.3e+00*)	1.7e+02	0	(*6.0e-01*)	(*1.2e+00*)	1.8e+02	0
LM_20	2.134e+05	2.740e+04	1.2e+02	90	(*3.1e-01*)	1.5e+02	0	(*2.4e+00*)	1.1e+00	0	(*7.4e+00*)	(*2.9e+00*)	2.2e+02	0	(*4.6e+00*)	(*2.7e+00*)	8.9e+02	0
NF2	9.375e+04	8.267e+04	1.5e+01	90	(*4.3e-01*)	1.0e+01	0	(*2.6e+03*)	2.9e-01	0	8.213e+04	5.961e+04	4.5e+01	30	6.352e+04	2.893e+04	3.5e+01	40
RG_5	4.773e+04	2.952e+04	8.7e+00	60	5.597e+03	1.2e+00	1	1.300e+01	4.0e-03	1	(*1.6e-02*)	(*9.1e-03*)	7.4e+01	0	1.130e+04	–	6.5e+00	10
RG_10	(*2.7e+00*)	(*9.4e-01*)	7.9e+01	0	5.135e+03	1.7e+00	1	2.300e+01	7.8e-03	1	(*5.0e+00*)	(*1.0e+00*)	1.6e+02	0	(*5.0e+00*)	(*1.0e+00*)	1.8e+02	0
RG_20	(*1.1e+01*)	(*9.8e-01*)	1.4e+02	0	7.097e+03	3.7e+00	1	4.300e+01	2.3e-02	1	(*2.8e+01*)	(*3.0e+00*)	2.1e+02	0	(*3.4e+01*)	(*2.0e+00*)	1.5e+03	0
S10	3.805e+04	4.809e+04	6.8e+00	100	(*-1.1e+01*)	2.7e+00	0	(*-9.3e-01*)	2.8e-01	0	3.746e+04	2.020e+04	2.1e+01	70	4.729e+04	–	2.6e+01	10
SS_5	1.429e+04	4.050e+03	2.6e+00	100	(*1.2e+00*)	5.1e+01	0	(*9.4e+01*)	2.2e-01	0	(*1.3e-04*)	(*6.5e-03*)	7.8e+01	0	(*4.4e-05*)	(*3.2e-03*)	7.6e+01	0
SS_10	7.321e+04	1.583e+04	2.2e+01	100	(*2.9e+02*)	1.4e+02	0	(*3.4e+02*)	3.9e-01	0	(*3.7e-02*)	(*6.3e-02*)	1.6e+02	0	(*1.7e-02*)	(*3.9e-02*)	1.8e+02	0
Total time			6.33e+02			4.26e+02			4.57e+00				1.73e+03				3.92e+03	

[(a)] corresponds to $npenf_{\max}$ =5000;
[†] $npenf_{\max} = 20$ and maximum number of iterations set to 10;
[‡] $npenf_{\max} = 3000n$ and other criteria like the average change in the function value, running time limit, objective function limit (among six criteria);
[♯] average time (in seconds) over the successful runs, or the time of the reported solution if none of the runs is successful;
[§] execution time (in seconds) to reach the solution if Suc='1', or the time of the reported solution if Suc='0'.

The total time required by 'DIRECT' (with setting $npenf_{\max}$=5000) to reach the reported solutions for all the problems is 4.26e+02. The default 'DIRECT' version takes much less time (4.57e+00 seconds) since it terminates mostly before a good solution is reached, due to the default parameter values ($npenf_{\max} = 20$ and maximum number of iterations set to 10).

When the penalty approach invokes the 'SA', we conclude that both tested versions, one based on setting $npenf_{\max}$ =5000 and the other based on default parameters, successfully solve (with 90-100% of successful runs) a small percentage of instances, 27% and 32% respectively. Overall, the first version produces a solution satisfying the conditions (19) in at least one run for 45% of the instances, against 55% of the second version. The total times to achieve the reported solutions are of the same order of magnitude: 1.73e+03 for 'SA' (with $npenf_{\max}$ =5000) and 3.92e+03 for 'SA' (with stopping default values).

From the comparison between the 'adaptive FA' and 'DIRECT' (with the setting $npenf_{\max}$ =5000) we conclude that the 'adaptive FA' performs better in terms of robustness (73% of successful runs against 55%), although it requires much more function evaluations. In terms of total time to achieve the reported solutions of all instances, 'adaptive FA' uses only 1.5 times more seconds than 'DIRECT'.

When a comparison is made between algorithms that converge successfully to the required solutions, the function evaluations required by the 'adaptive FA' are in general slightly larger than those of 'SA' (with the setting $npenf_{\max}$ =5000). However, the average times with 'SA' are mostly superior to those of the 'adaptive FA'. Furthermore, the 'adaptive FA' successfully solves almost three times more instances than 'SA'.

5. Conclusions

This article describes a penalty approach for solving MINLP problems that relies on a continuous reformulation of the MINLP problem by converting it to a finite sequence of nonlinear penalty problems with only continuous variables. For the two newly proposed penalty terms, 'tanh'-based and 'asinh'-based, we prove that the penalty problem and the MINLP problem are equivalent in the sense that they have the same minimizers. In the penalty-based algorithm, the global minimizer of the nonlinear penalty function is computed by a firefly algorithm. A variant of FA, coined 'adaptive FA', is proposed aiming to reduce control parameter dependence.

The numerical experiments carried out to compare the quality of the produced solutions, as well as the convergence speed of the algorithm, show that the hyperbolic tangent penalty produces slightly better results than the other two penalties in comparison, in terms of solution quality and average percentage of successful runs. Furthermore, the proposed 'adaptive FA'-penalty algorithm compares favorably with the penalty algorithm when the DIRECT deterministic solver or the stochastic simulated annealing solver are invoked.

Acknowledgments

The authors wish to thank two anonymous referees for their careful reading of the manuscript and their valuable comments and suggestions to improve the paper.

This work has been supported by FCT (Fundação para a Ciência e Tecnologia, Portugal) in the scope of the projects: PEst-UID/CEC/00319/2013 and PEst-OE/MAT/UI0013/2014.

Appendix A.

Proof of Property 2.5: According to Theorem 2.4, to prove that penalty P^t can be used in problem (3), we have to show that the penalty term (10) satisfies Assumptions 2.2 and 2.3. We assume that f satisfies Assumption 2.1.

Proof: For all $x, y \in W$, $P^t(x;\varepsilon) = \dfrac{1}{\varepsilon}\sum_{j\in I_d}\tanh(\varepsilon) = |I_d|\dfrac{1}{\varepsilon}\tanh(\varepsilon) = P^t(y;\varepsilon)$ and Assumption 2.2 is satisfied.

To prove that Assumption 2.3 is also satisfied, the reasoning used in [32] is extended to our penalty term. Thus, the behavior of the term related to x_j (for $j \in I_d$)

$$P_j^t(x_j;\varepsilon) = \frac{1}{\varepsilon}\min_{\substack{l_j\le d_i\le u_j\\ d_i\in\mathbb{Z}}}\tanh\left(|x_j - d_i| + \varepsilon\right),$$

in a neighborhood of z_j ($z \in W$) is analyzed in the following three cases, where ρ is a sufficiently small positive constant:

(i) $z_j = d_i$ and $d_i < x_j < d_i + \rho$;
(ii) $z_j = d_i$ and $d_i - \rho < x_j < d_i$;
(iii) $z_j = x_j = d_i$.

Case (i): Using the mean theorem

$$P_j^t(x_j;\varepsilon) - P_j^t(z_j;\varepsilon) = \frac{1}{\varepsilon\cosh^2\left((\tilde{x}_j - d_i) + \varepsilon\right)}|x_j - z_j| \ge \frac{1}{\varepsilon\cosh^2(\rho+\varepsilon)}|x_j - z_j|$$

where $\tilde{x}_j \in (d_i, x_j)$. We may choose ρ and ε such that

$$\frac{1}{\varepsilon\cosh^2(\rho+\varepsilon)} \ge \bar{L} \tag{A1}$$

and obtain

$$P_j^t(x_j;\varepsilon) - P_j^t(z_j;\varepsilon) \ge \bar{L}|x_j - z_j|. \tag{A2}$$

For Case (ii) and using the mean theorem, we get

$$P_j^t(x_j;\varepsilon) - P_j^t(z_j;\varepsilon) = \frac{1}{\varepsilon\cosh^2\left((d_i - \tilde{x}_j) + \varepsilon\right)}|x_j - z_j| \ge \frac{1}{\varepsilon\cosh^2(\rho+\varepsilon)}|x_j - z_j|$$

where $\tilde{x}_j \in (x_j, d_i)$. Relation (A2) is also obtained in this case.

When considering Case (iii), we get $P_j^t(x_j;\varepsilon) - P_j^t(z_j;\varepsilon) = 0$ by Assumption 2.2.

Thus, for ρ and ε satisfying (A1), we have

$$P^t(x;\varepsilon) - P^t(z;\varepsilon) \ge \bar{L}\sum_{j\in I_d}|x_j - z_j| = \bar{L}\|x - z\|_1 \ge \bar{L}\|x - z\|_\infty$$

for all $z \in C$ with $z_j \in \mathbb{Z}$ for $j \in I_d$, and $z_i = x_i$ for $i \in I\backslash I_d$, and for all x such that $\|x - z\|_\infty < \rho$.

Condition (6) in Assumption 2.3 holds if we define S as the union of the neighborhoods $S(z) = \{x \in \mathbb{R}^n : \|x - z\|_\infty < \rho\}$ of all $z \in W$.

To prove (7) and (8), let $\{\varepsilon^{(k)}\}$ be an infinite sequence that converges to 0 as $k \to \infty$ (where k represents the iteration counter of the algorithm) and let $\bar{x}$ be defined by

$$\begin{cases} \bar{x}_l = d_l \pm \rho, \text{ where } l_l \le d_l \le u_l, d_l \in \mathbb{Z}, \\ \bar{x}_i = d_i \text{ where } l_i \le d_i \le u_i, d_i \in \mathbb{Z} \text{ for all } i \neq l, \text{ where } i, l \in I_d \\ l_i \le \bar{x}_i \le u_i, \bar{x}_i \in \mathbb{R} \text{ for } i \in I \backslash I_d \end{cases} \tag{A3}$$

and for any $z \in W$, we have

$$\begin{aligned} \lim_{k\to\infty} \left(P^t(\bar{x}; \varepsilon^{(k)}) - P^t(z; \varepsilon^{(k)}) \right) &= \lim_{k\to\infty} \left(\frac{1}{\varepsilon^{(k)}} \tanh(\rho + \varepsilon^{(k)}) \right. \\ &\quad \left. + \frac{1}{\varepsilon^{(k)}} (|I_d| - 1) \tanh(\varepsilon^{(k)}) - \frac{1}{\varepsilon^{(k)}} |I_d| \tanh(\varepsilon^{(k)}) \right) \\ &= \lim_{k\to\infty} \frac{1}{\varepsilon^{(k)}} \left(\tanh(\rho + \varepsilon^{(k)}) - \tanh(\varepsilon^{(k)}) \right) \\ &= +\infty. \end{aligned}$$

Furthermore, for all $x \in C \backslash S$ and all $\varepsilon > 0$ we have

$$\begin{aligned} P^t(x; \varepsilon) - P^t(\bar{x}; \varepsilon) &= \frac{1}{\varepsilon} \sum_{j \in I_d} \min_{\substack{l_j \le d_i \le u_j \\ d_i \in \mathbb{Z}}} \tanh\left(|x_j - d_i| + \varepsilon\right) - \frac{1}{\varepsilon} \sum_{j \in I_d} \min_{\substack{l_j \le d_i \le u_j \\ d_i \in \mathbb{Z}}} \tanh\left(|\bar{x}_j - d_i| + \varepsilon\right) \\ &= \frac{1}{\varepsilon} \sum_{j \in I_d} \min_{\substack{l_j \le d_i \le u_j \\ d_i \in \mathbb{Z}}} \tanh\left(|x_j - d_i| + \varepsilon\right) - \frac{1}{\varepsilon} \tanh(\rho + \varepsilon) - \frac{1}{\varepsilon} (|I_d| - 1) \tanh(\varepsilon) \\ &= \frac{1}{\varepsilon} \sum_{j \neq \bar{l} \in I_d} \min_{\substack{l_j \le d_i \le u_j \\ d_i \in \mathbb{Z}}} \tanh\left(|x_j - d_i| + \varepsilon\right) - \frac{1}{\varepsilon} (|I_d| - 1) \tanh(\varepsilon) \\ &\quad + \frac{1}{\varepsilon} \tanh\left(|x_{\bar{l}} - \bar{d}| + \varepsilon\right) - \frac{1}{\varepsilon} \tanh(\rho + \varepsilon) \\ &\ge 0 \end{aligned}$$

since the hyperbolic tangent is strictly increasing, $|x_{\bar{l}} - \bar{d}| \ge \rho$ where $\bar{d}$ is the integer feasible value nearest to $x_{\bar{l}}$. □

Proof of Property 2.6: According to Theorem 2.4, to prove that penalty P^a can be used in problem (3) we show that the penalty term (11) satisfies Assumptions 2.2 and 2.3. We assume that f satisfies Assumption 2.1.

Proof: For all $x, y \in W$, $P^a(x; \varepsilon) = \sum_{j \in I_d} \operatorname{asinh}(\varepsilon) = |I_d| \operatorname{asinh}(\varepsilon) = P^a(y; \varepsilon)$ and Assumption 2.2 is satisfied.

To prove that Assumption 2.3 is also satisfied, the behavior of

$$P^a_j(x_j; \varepsilon) = \min_{\substack{l_j \le d_i \le u_j \\ d_i \in \mathbb{Z}}} \operatorname{asinh}\left(\frac{1}{\varepsilon} |x_j - d_i| + \varepsilon \right)$$

(for $j \in I_d$) in a neighborhood of a feasible z_j is analyzed using the three previously referred cases (see the proof of Property 2.5). Let $\rho > 0$ be a sufficiently small value.

Case (i): Using the mean theorem

$$P^a_j(x_j; \varepsilon) - P^a_j(z_j; \varepsilon) = \frac{1}{\varepsilon \sqrt{1 + \left(\frac{1}{\varepsilon} (\tilde{x}_j - d_i) + \varepsilon \right)^2}} |x_j - z_j| \ge \frac{1}{\varepsilon \sqrt{1 + \left(\frac{1}{\varepsilon} \rho + \varepsilon \right)^2}} |x_j - z_j|$$

where $\tilde{x}_j \in (d_i, x_j)$. Choosing ρ and ε such that

$$\frac{1}{\varepsilon\sqrt{1+\left(\frac{1}{\varepsilon}\rho+\varepsilon\right)^2}} \geq \bar{L} \tag{A4}$$

we obtain

$$P^a_j(x_j;\varepsilon) - P^a_j(z_j;\varepsilon) \geq \bar{L}|x_j - z_j|. \tag{A5}$$

Similarly for Case (ii), we have

$$\begin{aligned} P^a_j(x_j;\varepsilon) - P^a_j(z_j;\varepsilon) &= \frac{1}{\varepsilon\sqrt{1+\left(\frac{1}{\varepsilon}(d_i-\tilde{x}_j)+\varepsilon\right)^2}}|x_j - z_j| \\ &\geq \frac{1}{\varepsilon\sqrt{1+\left(\frac{1}{\varepsilon}\rho+\varepsilon\right)^2}}|x_j - z_j| \\ &\geq \bar{L}|x_j - z_j| \end{aligned}$$

where $\tilde{x}_j \in (x_j, d_i)$. For Case (iii), we obtain $P^a_j(x_j;\varepsilon) - P^a_j(z_j;\varepsilon) = 0$, and we may conclude that for ρ and ε satisfying (A4), we get

$$P^a(x;\varepsilon) - P^a(z;\varepsilon) \geq \bar{L}\sum_{j\in I_d}|x_j - z_j| = \bar{L}\|x-z\|_1 \geq \bar{L}\|x-z\|_\infty$$

for all $z \in C$ (where $z_j \in \mathbb{Z}$ for $j \in I_d$, and $z_i = x_i$ for $i \in I\backslash I_d$) and for all x such that $\|x - z\|_\infty < \rho$.

If S is defined as the union of the neighborhoods $S(z) = \{x \in \mathbb{R}^n : \|x-z\|_\infty < \rho\}$ of all $z \in W$, then condition (6) in Assumption 2.3 is satisfied.

To prove (7) and (8), we consider the infinite sequence $\{\varepsilon^{(k)}\}$ converging to 0 (for $k \to \infty$), the point $\bar{x}$ defined by (A3) and $z \in W$. Then,

$$\begin{aligned} \lim_{k\to\infty}\left(P^a(\bar{x};\varepsilon^{(k)}) - P^a(z;\varepsilon^{(k)})\right) &= \lim_{k\to\infty}\Bigg(\operatorname{asinh}\left(\frac{1}{\varepsilon^{(k)}}\rho+\varepsilon^{(k)}\right) \\ &\quad +(|I_d|-1)\operatorname{asinh}(\varepsilon^{(k)}) - |I_d|\operatorname{asinh}(\varepsilon^{(k)})\Bigg) \\ &= \lim_{k\to\infty}\left(\operatorname{asinh}\left(\frac{1}{\varepsilon^{(k)}}\rho+\varepsilon^{(k)}\right) - \operatorname{asinh}(\varepsilon^{(k)})\right) \\ &= +\infty \end{aligned}$$

Furthermore, for all $x \in C\backslash S$ and all $\varepsilon > 0$ we have

$$\begin{aligned}
P^a(x;\varepsilon) - P^a(\bar{x};\varepsilon) &= \sum_{j\in I_d} \min_{\substack{l_j \le d_i \le u_j \\ d_i \in \mathbb{Z}}} \operatorname{asinh}\left(\frac{1}{\varepsilon}|x_j - d_i| + \varepsilon\right) \\
&\quad - \sum_{j\in I_d} \min_{\substack{l_j \le d_i \le u_j \\ d_i \in \mathbb{Z}}} \operatorname{asinh}\left(\frac{1}{\varepsilon}|\bar{x}_j - d_i| + \varepsilon\right) \\
&= \sum_{j\in I_d} \min_{\substack{l_j \le d_i \le u_j \\ d_i \in \mathbb{Z}}} \operatorname{asinh}\left(\frac{1}{\varepsilon}|x_j - d_i| + \varepsilon\right) - \operatorname{asinh}\left(\frac{1}{\varepsilon}\rho + \varepsilon\right) \\
&\quad -(|I_d| - 1)\operatorname{asinh}(\varepsilon) \\
&= \sum_{j\neq \bar{l}\in I_d} \min_{\substack{l_j \le d_i \le u_j \\ d_i \in \mathbb{Z}}} \operatorname{asinh}\left(\frac{1}{\varepsilon}|x_j - d_i| + \varepsilon\right) - (|I_d| - 1)\operatorname{asinh}(\varepsilon) \\
&\quad + \operatorname{asinh}\left(\frac{1}{\varepsilon}|x_{\bar{l}} - \bar{d}| + \varepsilon\right) - \operatorname{asinh}\left(\frac{1}{\varepsilon}\rho + \varepsilon\right) \\
&\ge 0
\end{aligned}$$

for the monotonically increasing inverse hyperbolic sine function, where $|x_{\bar{l}} - \bar{d}| \ge \rho$ and $\bar{d}$ is the integer feasible value nearest to $x_{\bar{l}}$. □

Appendix B.

The collection of problems used in this study is listed below. They come from well-known test problems in continuous optimization and have been suitably modified by considering that at least one of the variables assumes only integer values [6, 30]:

Ackley (ACK_n) (with $x^* = (0, \dots, 0)$ and $f^* = 0$):

$$\min -20\exp\left(-0.2\sqrt{\tfrac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\tfrac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$$
$$x_i \in \{-30, \dots, 30\},\ i = 1, \dots, n$$

a) $n = 5$; **b)** $n = 10$; **c)** $n = 20$

Aluffi-Pentini (AP) (with $x^* = (-1.0465, 0)$ and $f^* \approx -0.3523$):

$$\min 0.25x_1^4 - 0.5x_1^2 + 0.1x_1 + 0.5x_2^2$$
$$x_1 \in [-10, 10] \text{ and } x_2 \in \{-10, \dots, 10\}$$

Beale (Bea) (with $x^* = (3, 0.5)$ and $f^* = 0$):

$$\min (1.5 - x_1(1 - x_2))^2 + (2.25 - x_1(1 - x_2^2))^2 + (2.625 - x_1(1 - x_2^3))^2$$
$$x_1 \in \{-5, \dots, 5\} \text{ and } x_2 \in [-4.5, 4.5]$$

Becker-Lago (BL) (with $x^* = (\mp 5, \mp 5)$ and $f^* = 0$):

$$\min (|x_1| - 5)^2 + (|x_2| - 5)^2$$
$$x_i \in \{-10, \dots, 10\},\ i = 1, 2$$

Bohachevsky 1 (BF1) (with $x^* = (0, 0)$ and $f^* = 0$):

$$\min x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$$
$$x_i \in \{-50, \dots, 50\},\ i = 1, 2$$

Bukin (Buk) (with $x^* = (-10, 1)$ and $f^* = 0$):

$$\min 100\sqrt{|x_2 - 0.01x_1^2|} + 0.01\,|x_1 + 10|$$
$$x_1 \in \{-15, -5\} \text{ and } x_2 \in \{-3, \dots, 3\}$$

Dekkers-Aarts (DA) (with $x^* = (0, \mp 15)$ and $f^* = -24777$):

$$\min 10^5 x_1^2 + x_2^2 - (x_1^2 + x_2^2)^2 + 10^{-5}(x_1^2 + x_2^2)^4$$
$$x_i \in \{-20, \dots, 20\},\ i = 1, 2$$

Table B1.: Data for Problem 14

(i)	a_{ij}				c_i
	(j) 1	2	3	4	
1	4	4	4	4	0.1
2	1	1	1	1	0.2
3	8	8	8	8	0.2
4	6	6	6	6	0.4
5	3	7	3	7	0.4
6	2	9	2	9	0.6
7	5	5	3	3	0.3
8	8	1	8	1	0.7
9	6	2	6	2	0.5
10	7	3.6	7	3.6	0.5

Dixon-Price (DP_n) (with $x^* = (0, \dots, 0)$ and $f^* = 0$):

$$\min\ (x_1 - 1)^2 + \sum_{i=2}^{n} i(2x_i^2 - x_{i-1})^2$$
$$x_1 \in \{-10, \dots, 10\} \text{ and } x_i \in [-10, 10],\ i = 2, \dots, n$$

a) $n = 2$; **b)** $n = 4$

Himmelblau (Him) (with $x^* = (3, 2)$ and $f^* = 0$):

$$\min\ (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$$
$$x_i \in \{-5, \dots, 5\},\ i = 1, 2$$

Levy-Montalvo 2 (LM2_n) (with the global $(1, 1, \dots, 1)$ and $f^* = 0$):

$$\min\ 0.1\left(\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\right)$$
$$x_i \in \{-5, \dots, 5\},\ i = 1, \dots, n$$

a) $n = 5$; **b)** $n = 10$; **c)** $n = 20$

Neumaier 2 (NF2) (with $x^* = (1, 2, 2, 3)$ and $f^* = 0$):

$$\min \sum_{j=1}^{4}\left(b_j - \sum_{i=1}^{4} x_i^j\right)^2$$
$$x_i \in \{0, \dots, 4\},\ i = 1, 2, 3, 4$$

where $b = (8, 18, 44, 114)$

Rastrigin (RG_n) (with $x^* = (0, \dots, 0)$ and $f^* = 0$):

$$\min \sum_{i=1}^{n}\left(x_i^2 - 10\cos(2\pi x_i)\right) + 10n$$
$$x_i \in \{-5, \dots, 5\},\ i = 1, \dots, n$$

a) $n = 5$; **b)** $n = 10$; **c)** $n = 20$

Shekel 10 (S10) (the global is located at $(4, 4, 4, 4)$ with $f^* \approx -10.5319$)

$$\min -\sum_{j=1}^{10} \frac{1}{\sum_{i=1}^{4}(x_j - a_{ij})^2 + c_i}$$
$$x_i \in \{0, \dots, 10\},\ i = 1, 2, 3, 4$$

Sum Squares (SS_n) (with the global $(0, 0, \dots, 0)$ and $f^* = 0$):

$$\min \sum_{i=1}^{n} i x_i^2$$
$$x_i \in \{-5, \dots, 10\},\ i = 1, \dots, n$$

a) $n = 5$; **b)** $n = 10$

References

[1] K. Abhishek, S. Leyffer, J.T. Linderoth, *FilMINT: An outer approximation-based solver for convex mixed-integer nonlinear programs*, INFORMS Journal on Computing, 22(4) (2010), pp. 555-567.

[2] M.A. Abramson, C. Audet, J.W. Chrissis, J.G. Walston, *Mesh adaptive direct search algorithms for mixed variable optimization*, Optimization Letter, 3(1) (2009), pp. 35–47.

[3] M.A. Abramson, C. Audet, J.E. Dennis Jr., *Filter pattern search algorithms for mixed variable constrained optimization problems*, Pacific Journal on Optimization, 3(3) (2007), pp. 477–500.

[4] C.S. Adjiman, I.P. Androulakis, C.A. Floudas, *A global optimization method, αBB, for general twice-differentiable constrained NLPs – II. Implementation and computational results*, Computers & Chemical Engineering, 22(9) (1998), pp. 1159–1179.

[5] M.M. Ali, M. Golalikhani, *An electromagnetism-like method for nonlinearly constrained global optimization*, Computers & Mathematics with Applications, 60(8) (2010), pp. 2279–2285.

[6] M.M. Ali, C. Khompatraporn, Z.B. Zabinsky, *A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems*, Journal of Global Optimization, 31 (2005), pp. 635–672.

[7] M.M. Ali, W.X. Zhu, *A penalty function-based differential evolution algorithm for constrained global optimization*, Computational Optimization and Applications, 54(3) (2013), pp. 707–739.

[8] S. Arora, S. Singh, *The firefly optimization algorithm: convergence analysis and parameter selection*, International Journal of Computer Applications, 69(3) (2013), pp. 48–52.

[9] C. Audet, J.E. Dennis Jr., *Pattern search algorithms for mixed variable programming*, SIAM Journal on Optimization, 11(3) (2001), pp. 573–594.

[10] P. Belotti, J. Lee, L. Liberti, F. Margot, A. Wächter, *Branching and bounds tightening techniques for non-convex MINLP*, Optimization Methods and Software, 24(4) (2009), pp. 597–634.

[11] P. Bonami, L.T. Biegler, A.R. Conn, G. Cornuéjols, I.E. Grossmann, C.D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, A. Wächter, *An algorithmic framework for convex mixed integer nonlinear programs*, Discrete Optimization, 5(2) (2008), pp. 186–204.

[12] P. Bonami, G. Cornuéjols, A. Lodi, F. Margot, *A feasibility pump for mixed integer nonlinear programs*, Mathematical Programming, 119(2) (2009), pp. 331–352.

[13] P. Bonami, J. Lee, S. Leyffer, A. Wächter, *More branch-and-bound experiments in convex nonlinear integer programming*, Preprint ANL/MCS-P1949-0911, Argonne National Laboratory, Mathematics and Computer Science Division, 2011.

[14] S. Burer, A.N. Letchford, *Non-convex mixed-integer nonlinear programming: a survey*, Surveys in Operations Research and Management Science, 17 (2012), pp. 97–106.

[15] Y. Changjun, K.L. Teo, Y. Bai, *An exact penalty function method for nonlinear mixed discrete programming problems*, Optimization Letters, 7(1) (2013), pp. 23–38.

[16] L. Costa, I.A.C.P. Espírito Santo, E.M.G.P. Fernandes, *A hybrid genetic pattern search augmented Lagrangian method for constrained global optimization*, Applied Mathematics and Computation, 218(18) (2012), pp. 9415–9426.

[17] M.F.P. Costa, A.M.A.C. Rocha, E.M.G.P. Fernandes, *An artificial fish swarm algorithm based hyperbolic augmented Lagrangian method*, Journal of Computational and Applied Mathematics, 259(Part B) (2014), pp. 868–876.

[18] M.F.P. Costa, A.M.A.C. Rocha, R.B. Francisco, E.M.G.P. Fernandes, *Heuristic-based firefly algorithm for bound constrained nonlinear binary optimization*, Advances in Operations Research 2014, Article ID 215182, (2014) 12 pages.

[19] C. D'Ambrosio, A. Frangioni, L. Liberti, A. Lodi, *A storm of feasibility pumps for nonconvex MINLP*, Mathematical Programming, Series B, 136(2) (2012), pp. 375–402.

[20] K. Deb, S. Srivastava, *A genetic algorithm based augmented Lagrangian method for constrained optimization*, Computational Optimization and Applications, 53(3) (2012), pp. 869–902.

[21] G. Di Pillo, S. Lucidi, F. Rinaldi, *An approach to constrained global optimization based on exact penalty functions*, Journal of Global Optimization, 54(2) (2012). pp. 251–260.

[22] Sh.M. Farahani, A.A. Abshouri, B. Nasiri, M.R. Meybodi, *A Gaussian firefly algorithm*, International Journal of Machine Learning and Computing, 1(5) (2011), pp. 448–453.

[23] I. Fister, I. Fister Jr., X.-S. Yang, J. Brest, *A comprehensive review of firefly algorithms*, Swarm and Evolutionary Computation, 13 (2013), pp. 34–46.

[24] L. Guo, G.-G. Wang, H. Wang, D. Wang, *An effective hybrid firefly algorithm with harmony search for global numerical optimization*, The Scientific World Journal, Volume 2013, Article ID 125625, 9 pages.

[25] R. Hemmecke, M. Koppe, J. Lee, R. Weismantel, *Nonlinear Integer Programming*, in M. Jünger, T. Liebling, D. Naddef, G. Nemhauser, W. Pulleyblank, G. Reinelt, G. Rinaldi, L. Wolsey (Eds.), *50 Years of Integer Programming 1958–2008: The Early Years and State-of-the-Art Surveys*, Springer-Verlag (2009) ISBN 3540682740.

[26] L. Ingber, *Very fast simulated re-annealing*, Mathematical and Computer Modelling, 12 (1989), pp. 967–973.

[27] D.R. Jones, *The DIRECT global optimization algorithm*, In: C. Floudas, P. Pardalos, (eds.) Encyclopedia of Optimization, pp. 431–440. Kluwer, Dordrecht (2001)

[28] X. Lin, Y. Zhong, H. Zhang, *An enhanced firefly algorithm for function optimisation problems*, International Journal of Modelling, Identification and Control, 18(2) (2013), pp. 166–173.

[29] G. Liuzzi, S. Lucidi, V. Piccialli, *A partition-based global optimization algorithm*, Journal of Global Optimization, 48(1) (2010), pp. 113–128.

[30] G. Liuzzi, S. Lucidi, F. Rinaldi, *Derivative-free methods for bound constrained mixed-integer optimization*, Computational Optimization and Applications, 53(2) (2012), pp. 505–526.

[31] G. Liuzzi, S. Lucidi, F. Rinaldi, *Derivative-free methods for mixed-integer constrained optimization problems*, Journal of Optimization Theory and Applications, 164(3) (2015), pp. 933–965.

[32] S. Lucidi, F. Rinaldi, *Exact penalty functions for nonlinear integer programming problems*, Journal

of Optimization Theory and Applications, 145(3) (2010), pp. 479–488.

[33] S. Lucidi, F. Rinaldi, *An exact penalty global optimization approach for mixed-integer programming problems*, Optimization Letters, 7(2) (2013), pp. 297–307.

[34] A. Manju, M.J. Nigam, *Firefly algorihtm with fireflies having quantum behavior*, in Institute of Electrical and Electronics Engineers, 2012 ICRCC, India, December 2012, pp. 117–119.

[35] W. Murray, K.-M. Ng, *An algorithm for nonlinear optimization problems with binary variables*, Computational Optimization and Applications, 47(2) (2010), pp. 257–288.

[36] H.S. Ryoo, N.V. Sahinidis, *Global optimization of nonconvex NLPs and MINLPs with applications in process design*, Computers and Chemical Engineering, 19(5) (1995), pp. 551–566.

[37] R.A. Shandiz, N. Mahdavi-Amiri, *An exact penalty approach for mixed integer nonlinear programming problems*, American Journal of Operations Research, 1(3) (2011), pp. 185–189.

[38] M. Tawarmalani, N.V. Sahinidis, *A polyhedral branch-and-cut approach to global optimization*, Mathematical Programming, Series B, 103(2) (2005), pp. 225–249.

[39] S.L. Tilahun and H.C. Ong, *Modified firefly algorithm*, Journal of Applied Mathematics, Volume 2012 (2012), Article ID 467631, 12 pages.

[40] X.-S. Yang, *Firefly algorithms for multimodal optimization*, in O. Watanabe, T. Zeugmann (Eds.) Stochastic Algorithms: Foundations and Applications (SAGA 2009) Lecture Notes in Computer Sciences, Vol. 5792, (2009), pp. 169–178.

[41] X.-S. Yang, *Firefly algorithm, stochastic test functions and design optimization*, International Journal of Bio-Inspired Computation, 2(2) (2010), pp. 78–84.

[42] X.-S. Yang, X. He, *Firefly algorithm: recent advances and applications*, International Journal of Swarm Intelligence, 1(1) (2013), pp. 36–50.

[43] S. Yu, S. Yang, S. Su, *Self-adaptive step firefly algorithm*, Journal of Applied Mathematics, Volume 2013 (2013), Article ID 832718, 8 pages.