Improving Efficiency of a Multistart with Interrupted Hooke-and-Jeeves Filter Search for Solving MINLP Problems

Florbela P. Fernandes[1], M. Fernanda P. Costa[2,3], Ana Maria A.C. Rocha[4,5], and Edite M.G.P. Fernandes[5]

[1] ESTiG, Polytechnic Institute of Bragança, 5301-857 Bragança, Portugal,
fflor@ipb.pt
[2] Department of Mathematics and Applications, University of Minho,
4800-058 Guimarães, Portugal,
[3] Centre of Mathematics,
mfc@math.uminho.pt
[4] Department of Production and Systems, University of Minho,
4710-057 Braga, Portugal,
[5] Algoritmi Research Centre,
University of Minho, 4710-057 Braga, Portugal
arocha@dps.uminho.pt, emgpf@dps.uminho.pt

Abstract. This paper addresses the problem of solving mixed-integer nonlinear programming (MINLP) problems by a multistart strategy that invokes a derivative-free local search procedure based on a filter set methodology to handle nonlinear constraints. A new concept of componentwise normalized distance aiming to discard randomly generated points that are sufficiently close to other points already used to invoke the local search is analyzed. A variant of the Hooke-and-Jeeves filter algorithm for MINLP is proposed with the goal of interrupting the iterative process if the accepted iterate falls inside an ϵ-neighborhood of an already computed minimizer. Preliminary numerical results are included.

Keywords: nonconvex MINLP, multistart, Hooke-and-Jeeves, filter method

1 Introduction

In this paper, we address the problem of solving mixed-integer nonlinear programming (MINLP) problems by a multistart strategy and a derivative-free local search procedure. Nonlinear inequality and equality constraints are handled by a filter set methodology. The MINLP problem has the form

$$\begin{aligned} \min\ & f(x,y) \\ \text{subject to}\ & g_j(x,y) \leq 0,\ j=1,\ldots,m_g \\ & h_i(x,y) = 0,\ i=1,\ldots,m_h \\ & x \in \Gamma_x,\ y \in \Gamma_y \end{aligned} \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function, $g : \mathbb{R}^n \to \mathbb{R}^{m_g}$ and $h : \mathbb{R}^n \to \mathbb{R}^{m_h}$ are the constraint functions. The continuous variables are represented by the vector x, with $\Gamma_x \subset \mathbb{R}^{n_c}$ being the set of simple bounds on x:

$$\Gamma_x = \{x \in \mathbb{R}^{n_c} : \ l_x \leq x \leq u_x\},$$

with $l_x, u_x \in \mathbb{R}^{n_c}$. The integer variables are represented by the vector y, where $\Gamma_y \subset \mathbb{Z}^{n_i}$ is the set of simple bounds on y:

$$\Gamma_y = \{y \in \mathbb{Z}^{n_i} : \ l_y \leq y \leq u_y\},$$

with $l_y, u_y \in \mathbb{Z}^{n_i}$, and the parameter $n = n_c + n_i$ represents the total number of variables.

MINLP problems combine the combinatorial difficulty of optimizing over discrete variable sets with the challenge of handling nonlinear functions. When all functions involved in the problem are convex, the problem is a convex MINLP problem; otherwise it is a nonconvex one. The feasible region is in general nonconvex and multiple minima may appear. Furthermore, if some of the involved functions are nonsmooth then finding a solution to the MINLP problem is a challenging issue. For example, when the objective and constraints are provided as black-boxes, the MINLP is a nonsmooth and nonconvex problem. Convex MINLP problems are much easier to solve than nonconvex ones and therefore likely to be tractable, at least in theory. New techniques for MINLP can be found in [1,2,3,4,5]. The recent work presented in [6] is a detailed literature review and contains theoretical contributions, algorithmic developments, software implementations and applications for MINLP. The area of nonconvex MINLP has deserved a lot of attention from researchers working with heuristics, like genetic algorithm [7], ant colony [8], evolutionary algorithms [9], pattern search algorithms [10], multistart Hooke-and-Jeeves algorithm [11,12] and differential evolution [13].

The two major issues to be addressed when solving nonsmooth and nonconvex MINLP problems are: i) the integrality of some variables, ii) lack of smoothness and convexity of the involved functions. The issue of locating multiple solutions of MINLP problems by using a multistart method [14], the Hooke-and-Jeeves (HJ) algorithm [15,16] for the local search, and a filter set methodology [17] to handle inequality and equality constraints has been addressed in the past (see [11]). The HJ algorithm has been extended to define a special pattern of points spread over the search space for the integer variables and continuous variables separately. Furthermore, a filter methodology [10,17,18] to assess objective function and constraint violation function separately in order to promote convergence to feasible and optimal solutions has been incorporated.

The present study is an extension of the work presented in [11] in two aspects. First, the multistart algorithm is modified in a way that randomly generated points in the search space that are sufficiently close to other already generated and used points are discarded. This strategy aims to improve the diversification attribute of the multistart strategy, since points too close to other already used

points would be starting points (to the local search) that would converge to already computed minimizers. To check closeness, the concept of componentwise normalized distance is presented. A new stopping condition for the new multistart algorithm is also proposed. The condition joins the probability of a point being generated in the area of the search space not yet covered and the probability of finding a new minimizer after the decision of invoking the local search. Second, the HJ filter based algorithm [11] is further explored by incorporating a methodology that goals the interruption of the HJ iterations when an accepted trial iterate enters an ϵ-neighborhood of an already computed minimizer.

The remaining part of the paper is organized as follows. In Section 2, the new multistart algorithm is presented and discussed and Section 3 describes the interrupted HJ filter local search methodology. Section 4 reports on the preliminary numerical experiments and the paper is concluded in Section 5.

2 An Efficient Multistart Method

The multistart algorithm is a stochastic strategy that repeatedly applies a local search procedure starting from sampled points, i.e., points randomly generated inside the search space, aiming to converge to a solution of the optimization problem. The goal of the local search procedure is to deeply exploit around the starting sampled point. When a problem has a multimodal objective function, the multistart algorithm is capable of converging to the multiple solutions, although some or all of them may be found over and over again.

In this section, we present a multistart algorithm that incorporates two important concepts. One is concerned with the componentwise normalized distance of the current sampled point to other previously used sampled point, and the other is related with the region of attraction of a minimizer X^*, herein represented by $A(X^*)$.

When the componentwise normalized distance of the current sampled point to other previously used sampled points is used, the algorithm avoids using a sampled point that is sufficiently close to other sampled points already used to start the local search, by discarding it and generating another one. This strategy aims to improve efficiency and increase the diversification capacity of the algorithm when points are randomly generated in the search space $\Gamma_x \times \Gamma_y$.

Using regions of attraction, the algorithm avoids invoking a local search procedure starting from a sampled point when the probability of converging to an already detected minimizer is high. This way the algorithm avoids convergence to minimizers over and over again. A region of attraction of a minimizer is a popular concept to avoid convergence to previously computed solutions by defining prohibited regions around them [14,19]. Each time a sampled point falls inside that prohibited region, it will be discarded since the implementation of a local search procedure will produce one of the previously computed solutions. Thus, the efficiency of the algorithm improves since the number of calls to the local search procedure is reduced. Clustering techniques are sophisticated variants of the multistart method that use a typical distance concept and a gradient cri-

terion to decide which point is used to start a local search. They define a set of points that are believed to belong to the region of attraction of a particular minimizer, also denoted as a cluster [20].

Hereafter, the following notation is used: $X = (x, y)^T$ represents the vector of the n variables, $L = (l_x, l_y)^T$ and $U = (u_x, u_y)^T$ represent the lower and upper bound vectors on the variables. In a multistart context, to randomly generate a point with n_x continuous variables and n_y integer variables, in $[L, U]$, we proceed as follows:

$$\begin{array}{ll} x_i = (l_x)_i + \lambda_i((u_x)_i - (l_x)_i) & \text{for } i = 1, \ldots, n_x \\ y_j = (l_y)_j + \tau_j & \text{for } j = 1, \ldots, n_y \end{array} \tag{2}$$

where the notation $(l_x)_i$ represents the component i of the vector l_x (similarly for $(u_x)_i, (l_y)_j, (u_y)_j$), λ_i is a number uniformly distributed in $[0, 1]$ and τ_j is a number randomly selected from the set $\{0, 1, \ldots, ((u_y)_j - (l_y)_j)\}$.

The componentwise normalized distance of the current sampled point (x, y) to another already used sampled point $\bar{X} = (\bar{x}, \bar{y})^T$ depends on the (componentwise) size of the search space, the number of already used sampled points, t_{rand}, and is computed for the continuous and integer variables separately as follows:

$$D_x(x, \bar{x}) = \sum_{i=1}^{n_x} \frac{(x_i - \bar{x}_i)^2}{(d^i_{avg})^2} \quad \text{and} \quad D_y(y, \bar{y}) = \sum_{i=1}^{n_y} \frac{(y_i - \bar{y}_i)^2}{(d^{n_x+i}_{avg})^2} \tag{3}$$

where each component of the vector $d_{avg} = (d^1_{avg}, \ldots, d^n_{avg})^T$ represents the average distance between two different points and is defined by

$$d^i_{avg} = \frac{(u_x)_i - (l_x)_i}{(t_{rand} + 1)}, i = 1, \ldots, n_x \quad \text{and} \quad d^{n_x+i}_{avg} = \frac{(u_y)_i - (l_y)_i}{(t_{rand} + 1)}, i = 1, \ldots, n_y.$$

The use of the normalized distance aims to take into account different variable and search domain scaling. Furthermore, separate D_x and D_y values allow the use of different continuous and integer distance tolerances. The proposed multistart algorithm uses both distances $D_x(x, \bar{x})$ and $D_y(y, \bar{y})$ to check if the current sampled point X is sufficiently close to an already used sampled point $\bar{X}$, in which case the point is discarded since that region has been already explored and the local search procedure applied to X is likely to converge to an already detected minimizer. The herein implementation of the normalized distances aiming to discard a sampled point X that is sufficiently close to $\bar{X}$ uses the following criteria and tolerances:

$$D_x(x, \bar{x}) \leq 1 \text{ and } D_y(y, \bar{y}) \leq 1. \tag{4}$$

This strategy also goals the diversification of the sampled points by effectively using points that are far apart.

The region of attraction of a minimizer, $X^* = (x^*, y^*)^T$, associated with a local search procedure $\mathbf{L}$, is defined as

$$A(X^*) \equiv \{X \in [L, U] : \mathbf{L}(X) = X^*\}, \tag{5}$$

which means that if a point X is randomly selected from the set $[L, U]$ and belongs to the region of attraction $A(X^*)$, then X^* is obtained when $\mathbf{L}$ is invoked starting from X [14]. The regions of attraction are used to check if the sampled point does not belong to any of the regions of attraction of already computed minimizers or, equivalently, to the union of those regions of attraction (since they do not overlap). In this case, the local search procedure may be invoked to converge to a minimizer not yet detected. Since the region of attraction $A(X^*)$ is difficult to compute in practice, the probability, p, that the sampled point X will not belong to the union of the regions of attraction of the previously computed minimizers is used instead. This probability is easily estimated by the probability that X will not belong to the region of attraction of the nearest to X minimizer, X_o^*,

$$p = Prob[X \notin \cup_{i=1}^{s} A(X_i^*)] = \prod_{i=1}^{s} Prob[X \notin A(X_i^*)] \approx Prob[X \notin A(X_o^*)]$$

where o is the index of the nearest to X minimizer and s is the number of the already computed minimizers. Furthermore, $Prob[X \notin A(X_o^*)]$ is approximated by $Prob[X \notin B(X_o^*, R_o)]$ where $B(X_o^*, R_o)$ is the closed n-ball of radius R_o and center X_o^*. For any i, the maximum attractive radius of the minimizer X_i^* [14], denoted by R_i, is defined by

$$R_i = \max_j \left\{ \left\| \bar{X}_j - X_i^* \right\|_2 \right\}, \tag{6}$$

where $\bar{X}_j$ is the sampled point j that has already converged to the minimizer X_i^* after invoking the local search procedure, being the distance between X (the current sampled point) and the minimizer X_i^* given by $d_i = \|X - X_i^*\|_2$. Then if $d_i < R_i$, the point X is likely to be inside the region of attraction of X_i^* and the local search procedure ought not to be invoked since the probability of converging to the minimizer X_i^* is high. However, if the direction from X to X_i^* is ascent, i.e, the objective function increases if one goes from X to X_i^*, then X is likely to be outside the region of attraction of X_i^* and the local search procedure ought to be invoked, starting from X, since a new minimizer could be detected with high probability [14]. Thus, p is set to 1 if $d_i \geq R_i$ or if $d_i < R_i$ but the direction from X to X_i^* is ascent; otherwise $0 < p < 1$ and we set simply as a fraction of d_i/R_i.

Algorithm 1 displays the steps of the proposed multistart algorithm. The set Δ^*, empty at the beginning of the iterative process, contains all different computed minimizers. The condition $X^* \in \Delta^*$ is represented by the following conditions

$$|f(X^*) - f(X_l^*)| \leq \gamma^* \text{ and } \|x^* - x_l^*\|_2 \leq \gamma^* \text{ and } \|y^* - y_l^*\|_2 = 0 \tag{7}$$

for any $l \in \{1, \ldots, s\}$ and a small $\gamma^* > 0$, and means that the minimizer X^* has been previously detected.

Although the basic multistart algorithm is easily perceived and simple to code, it requires an adequate stopping rule to be effective. The goal of the stopping rule is to make the algorithm to stop when all minimizers have been located

and it should not require a large number of calls to the local search procedure to decide that all minimizers have been found.

Algorithm 1 Efficient multistart algorithm

Require: L, U, $\xi > 0$, $0 < \delta < 1$, $K_{\max}$, set $\Delta^* = \emptyset$, $s = 1$, $t_{local} = 1$, $t_{rand} = 1$, $k = 1$;
Randomly generate $X \in [L, U]$ (sampled point), set $\bar{X}_1 \leftarrow X$;
Compute $X_1^* = \mathbf{L}(X)$, $R_1 = \|X - X_1^*\|_2$, set $r_1 = 1$, $\Delta^* = \Delta^* \cup X_1^*$;
repeat
  Randomly generate $X \in [L, U]$ (sampled point);
  if $D_x(x - \bar{x}_i) > 1 \vee D_y(y - \bar{y}_i) > 1$ for all $i \in \{1, \ldots, t_{rand}\}$ **then**
    Set $t_{rand} = t_{rand} + 1$, set $\bar{X}_{t_{rand}} = X$;
    Set $o = \arg\min_{j=1,\ldots,s} d_j \equiv \|X - X_j^*\|_2$;
    if $d_o < R_o$ **then**
      if the direction from X to X_o^* is ascent **then**
        Set $p = 1$;
      else
        Set $p = \delta(d_o / R_o)$;
      end if
    else
      Set $p = 1$;
    end if
    if $rand(0, 1) < p$ **then**
      Compute $X^* = \mathbf{L}(X)$, set $t_{local} = t_{local} + 1$;
      if $X^* \in \Delta^*$ **then**
        Update $R_l = \max\{R_l, \|X - X_l^*\|_2\}$, $r_l = r_l + 1$;
      else
        Set $s = s+1$, $X_s^* = X^*$, $r_s = 1$, $\Delta^* = \Delta^* \cup X_s^*$, compute $R_s = \|X - X_s^*\|_2$;
      end if
    else
      Update $R_o = \max\{R_o, \|X - X_o^*\|_2\}$, $r_o = r_o + 1$;
    end if
  end if
  Set $k = k + 1$;
until $\frac{t_{rand}}{k} \frac{s}{t_{local}} \leq \xi$ or $t_{local} > K_{\max}$

A simple stopping rule [21] uses an estimate of the fraction of uncovered space, $s(s+1)/(t_{local}(t_{local}-1)) \leq \xi$ where s represents the number of (different) computed minimizers, t_{local} represents the number of local search calls and ξ is a small positive constant. Previous experiments show that this condition might not be efficient since a large number of sampled points may be required to start a local search procedure (making t_{local} to increase) although the search ends up by locating a previously identified minimizer (and s is not increased).

In this paper, we propose a different and yet simple rule to reflect a reasonable coverage of the search space. To check if the diversification procedure was

able to cover all the search space and detect all minimizers, the product of the probability that a good sampled point is found (in the area of the search space not yet covered), by the probability that a different minimizer is found when the local search is invoked (noting that invocation itself follows the region of attraction concept) is required to be small:

$$\frac{t_{rand}}{k}\frac{s}{t_{local}} \leq \xi,$$

for a small $\xi > 0$. The parameter t_{rand} represents the number of effectively used sampled points and k is the total number of generated points; s and t_{local} have the same meaning as above. Alternatively, the multistart algorithm stops when the number of local search calls exceeds a maximum value $K_{\max}$.

3 The 'Interrupted HJ-filter' Method for MINLP

In this section, we address the issue related with the local search procedure invoked inside the multistart algorithm, represented by **L** in Algorithm 1. A derivative-free pattern search method that is prepared to handle inequality and equality constraints by means of a filter methodology is used. Furthermore, the algorithm has been extended to be able to handle continuous and integer variables simultaneously. Thus, the proposed algorithm relies on the HJ approach, as outlined in [15], for solving a MINLP problem. This is an extension of the work presented in [11,12] in the sense that multiple solutions of problems like (1) are computed using a multistart algorithm that invokes a modified HJ filter based local search.

The herein proposed 'Interrupted HJ-filter' local search algorithm is an iterative process that is applied to a sampled point $X = (x, y)^T$ in order to provide a trial point $X^+ = (x^+, y^+)^T$ that is a global or a local minimizer of the problem (1). At each iteration, an acceptable trial point is obtained by searching around a central point. The term acceptable point means that the point is not dominated by any other point that belongs to a filter set.

Two stopping conditions are used to terminate the 'Interrupted HJ-filter' algorithm. When it is not possible to find an acceptable trial point in a very small neighborhood of the central point of the search, the iterative process can be terminated with the central point as final solution. When the acceptable trial point falls inside an ϵ-neighborhood of a previously computed minimizer, the iterative process is interrupted since the likelihood that the process will converge to that minimizer is high. This strategy aims to improve efficiency by avoiding convergence to previously detected solutions.

Thus, the most relevant issues addressed by the 'Interrupted HJ-filter' algorithm are the following. The algorithm

- accommodates two different pattern of points to be able to handle simultaneously continuous and integer variables, when generating trial points;
- uses a projection onto the search space $[L, U]$ when generated trial points violate the bound constraints during exploratory and pattern moves;

– incorporates the filter set methodology to handle equality and inequality constraints by accepting trial points that improve feasibility or optimality;
– interrupts the iterative search process when a trial point is accepted as the new iterate and is inside an ϵ-neighborhood of a previously computed minimizer;
– terminates the iterations when the step size α_x for the continuous variables falls below a pre-specified small tolerance $\alpha_{\min}$.

Using a filter methodology [7,10,17,18], problem (1) is reformulated as a bi-objective optimization problem

$$\begin{array}{c} \min\ [f(x,y), \theta(x,y)] \\ \text{subject to } x \in \Gamma_x,\ y \in \Gamma_y \end{array} \tag{8}$$

where $\theta(x,y) = \|g(x,y)_+\|_2^2 + \|h(x,y)\|_2^2$ is the nonnegative constraint violation function and $v_+ = \max\{0, v\}$. The concept of nondominance, borrowed from the multi-objective optimization, aims to build a filter set that is able to accept trial points if they improve the constraint violation or the objective function value. The filter $\mathcal{F}$ is defined as a finite set of points (x,y), corresponding to pairs $(\theta(x,y), f(x,y))$, none of which is dominated by any of the others. A point (x,y) is said to dominate a point (x',y') if and only if $\theta(x,y) \leq \theta(x',y')$ and $f(x,y) \leq f(x',y')$.

Every time the local search procedure is invoked inside the multistart algorithm, the filter is initialized to $\mathcal{F} = \{(\theta, f) : \theta \geq \theta_{\max}\}$, where $\theta_{\max} > 0$ is an upper bound on the acceptable constraint violation.

We now discuss the implemented 'Interrupted HJ-filter' local search algorithm with pseudo-code shown in Algorithm 2. Like the classical HJ, the HJ-filter search procedure comprises both the exploratory and the pattern moves. The exploratory move starts by generating a set of n possible trial points along the unit coordinate vectors $e_i \in \mathbb{R}^n$ with a fixed step size $\alpha_x \in (0,1]$ for the continuous variables and a unitary step for the integer ones:

$$x^+ = \bar{x} \pm \alpha_x D e_i,\ i = 1, \ldots, n_x, \quad \text{and} \quad y^+ = \bar{y} \pm e_i,\ \ i = n_x + 1, \ldots, n, \tag{9}$$

where (x^+, y^+) is a trial point, $(\bar{x}, \bar{y})$ is the central point of the search and $D \in \mathbb{R}^{n_x \times n_x}$ is a weighting diagonal matrix. The initial central point of the current iteration is the sampled point (x,y). If one of the following conditions

$$\theta(x^+, y^+) < (1 - \gamma_\theta)\ \theta(\bar{x}, \bar{y}) \quad \text{or} \quad f(x^+, y^+) \leq (1 - \gamma_f)\ f(\bar{x}, \bar{y}) \tag{10}$$

holds, for fixed constants $\gamma_\theta, \gamma_f \in (0,1)$, and (x^+, y^+) is acceptable by the filter, then the point (x^+, y^+) is accepted and replaces $(\bar{x}, \bar{y})$. We note that if a sequence of trial points is feasible, the condition (10) guarantees that the trial approximation (x^+, y^+) must satisfy the second condition in order to be acceptable. This way the optimal solution is guaranteed. Whenever a point is acceptable, the point is added to the filter $\mathcal{F}$, and all dominated points are removed from the filter. The search then selects the most nearly feasible point, (x^{inf}, y^{inf}), among all the accepted trial points.

If $(x^{inf}, y^{inf}) \neq (x, y)$, the iteration is successful and a pattern move is carried out through the direction $(x^{inf}, y^{inf}) - (x, y)$. A new set of n possible trial points along the unit coordinate vectors is generated, as shown in (9), replacing $(\bar{x}, \bar{y})$ by $(x^{inf}, y^{inf}) + ((x^{inf}, y^{inf}) - (x, y))$ (as central point). If the most nearly feasible point, selected among the trial points, is acceptable, (x^{inf}, y^{inf}) is accepted as the new iterate, replaces $(\bar{x}, \bar{y})$, and the pattern move is repeated.

Algorithm 2 Interrupted HJ-filter algorithm

Require: X (sampled point), $\alpha_{\min}$ and computed minimizers $X_i^*, i \in \{1, \ldots, s\}$;
Initialize the filter;
Set central point $\bar{X} = X$, set $k = 0$;
repeat
  Generate n possible trial points around central point;
  Check feasibility and optimality conditions of trial points;
  if trial points are acceptable by the filter **then**
    Update the filter;
    Choose the most nearly feasible point (x^{inf}, y^{inf});
    Set $\bar{X} = (x^{inf}, y^{inf})^T$;
    while (x^{inf}, y^{inf}) is a non-dominated point **do**
      (pattern move) Make a pattern move and define new central point;
      Generate n possible trial points around the central point;
      Check feasibility and optimality conditions of trial points;
      if trial points are acceptable by the filter **then**
        Update the filter;
        Choose the most nearly feasible point (x^{inf}, y^{inf});
        Set $\bar{X} = (x^{inf}, y^{inf})^T$;
      else
        Recuperate $\bar{X}$;
      end if
    end while
  else
    (restoration phase) Recuperate the most nearly feasible point in the filter as central point;
    Generate n possible trial points around the central point;
    Check feasibility and optimality conditions of trial points;
    if trial points are acceptable by the filter **then**
      Update the filter;
      Choose the most nearly feasible point (x^{inf}, y^{inf});
      Set $\bar{X} = (x^{inf}, y^{inf})^T$;
    else
      Recuperate $\bar{X}$;
      Reduce α_x;
    end if
  end if
  Set $k = k + 1$;
until $\alpha_x \leq \alpha_{\min}$ or $(\mathrm{mod}(k, 5) = 0 \wedge \exists i \in \{1, \ldots, s\} : \|\bar{x} - x_i^*\|_2 \leq \epsilon \wedge \|\bar{y} - y_i^*\|_2 \leq 1)$

However, when $(x^{inf}, y^{inf}) = (x, y)$, the iteration is unsuccessful and a restoration phase is invoked. In this phase, the most nearly feasible point in the filter, $(x_{\mathcal{F}}^{inf}, y_{\mathcal{F}}^{inf})$, is recuperated and a set of n possible trial points along the unit coordinate vectors is generated, as shown in (9), replacing $(\bar{x}, \bar{y})$ by $(x_{\mathcal{F}}^{inf}, y_{\mathcal{F}}^{inf})$ as the central point. If a non-dominated trial point is found, then it will become the central point for the next iteration. Otherwise, the iteration remains unsuccessful, the search returns back to the current $(\bar{x}, \bar{y})$, the step size α_x is reduced, and a new search consisting of exploratory moves and pattern moves is repeated taking $(\bar{x}, \bar{y})$ as the central point [11,12].

When α_x is reduced within an unsuccessful iteration, it may fall below a sufficiently small positive tolerance, $\alpha_{\min}$, thus indicating that no further improvement is possible and a solution is found. For the interruption issue of the HJ-filter algorithm, an acceptable trial point X is considered to be in the ϵ-neighborhood of an already computed minimizer X_i^*, for some $i \in \{1, \ldots, s\}$, if the distance to the minimizer verifies

$$||x - x_i^*||_2 < \epsilon \text{ and } ||y - y_i^*||_2 \leq 1, \tag{11}$$

where ϵ is a small positive tolerance. In this case, the likelihood that the search will converge to the minimizer X_i^* is high and the local search is interrupted. We remark that conditions (11) are tested only every five iterations.

4 Numerical Results

In this section, we aim to analyze the performance of the proposed multistart method based on the 'Interrupted HJ-filter' local search procedure. Four small problems are selected for these preliminary experiments. The results were obtained in a PC with an Intel Core i7-2600 CPU (3.4GHz) and 8 GB of memory. The parameters of the multistart and 'Interrupted HJ-filter' algorithms are set after an empirical study as follows: $K_{\max} = 20$, $\xi = 0.1$, $\delta = 0.5$, $\gamma^* = 0.005$, $\gamma_\theta = \gamma_f = 10^{-8}$, $\theta_{\max} = 10^2 \max\{1, \theta(\bar{x}, \bar{y})\}$, $\alpha_{\min} = 10^{-4}$ and $\epsilon = 0.05$. A solution is considered feasible when $\theta(x, y) \leq 10^{-8}$. Due to the stochastic nature of the multistart algorithm, each problem was solved 30 independent times and the results correspond to average values.

This section presents the full description of the problems, the results obtained by the proposed algorithm and a comparison with the work presented in [11].

Problem 1. (Example 1 in [22]) with 2 known solutions (1 global and 1 local)

$$\begin{aligned} &\min -x - y \\ &\text{subject to } xy - 4 \leq 0, \\ &\qquad 0 \leq x \leq 4, \;\; y \in \{0, \ldots, 6\} \end{aligned}$$

In [11] two solutions were detected. The global solution was found in all the 30 runs and the local in 24. The new Algorithm 1 based on the 'Interrupted HJ-filter' local search produces the global solution in all runs and the local in 18 out of 30. An average of 34 iterations were carried out and the average number of local solver calls was 9.1.

Problem 2. (Example 11 in [22]) with 2 known solutions (1 global and 1 local)

$$\begin{aligned} \min\ & 35x_1^{0.6} + 35x_2^{0.6} \\ \text{subject to}\ & 600x_1 - 50y - x_1 y + 5000 = 0 \\ & 600x_2 + 50y - 15000 = 0 \\ & 0 \le x_1 \le 34, 0 \le x_2 \le 17,\ y \in \{100, \ldots, 300\} \end{aligned}$$

When solving this problem, [11] detected two optimal solutions, one global and one local. The global was located in all runs and the local only in two out of the 30 runs. The herein presented Algorithm 1 detected the global in all runs and the local solution in four out of 30 runs, after an average of 21 iterations and 12.5 local search calls.

Problem 3. (Example 21 in [22]) with 2 known solutions (1 global and 1 local)

$$\begin{aligned} \min\ & x_1^{0.6} + y_1^{0.6} + y_2^{0.4} - 4y_2 + 2x_2 + 5y_3 - y_4 \\ \text{subject to}\ & x_1 + 2x_2 - 4 \le 0 \\ & y_1 + y_3 - 4 \le 0 \\ & y_2 + y_4 - 6 \le 0 \\ & -3x_1 + y_1 - 3x_2 = 0 \\ & -2y_1 + y_2 - 2y_3 = 0 \\ & 4x_2 - y_4 = 0 \\ & 0 \le x_1 \le 3,\ 0 \le x_2 \le 2,\ y_1, y_2 \in \{0, \ldots, 4\},\ y_3 \in \{0, 1, 2\},\ y_4 \in \{0, \ldots, 6\} \end{aligned}$$

Two optimal solutions, one global and one local, were found in [11]. The global was located in all runs and the local in 22 out of 30 runs. The new Algorithm 1 located the global in all runs and the local in four, after an average of 32 iterations and 20 local search calls.

Problem 4. (Example 13 in [22], f1 in [7]) with 2 known solutions (1 global and 1 local)

$$\begin{aligned} \min\ & 2x + y \\ \text{subject to}\ & 1.25 - x^2 - y \le 0 \\ & x + y - 1.6 \le 0 \\ & 0 \le x \le 1.6,\ y \in \{0, 1\} \end{aligned}$$

In [11] two solutions were detected, one global and one local. The global was located in all runs and the local was located only in two runs. In the present study, and after an average of 12 iterations and 2.1 local search calls, Algorithm 1 produces the global solution in all runs and the local in one out of the 30 runs.

The results presented in Table 1 aim to compare the efficiency of the Algorithm 1 based on the 'Interrupted HJ-filter' strategy with the multistart (MS) HJ-filter based algorithm in [11]. For each problem, the table shows averaged values over the 30 runs – the average overall number of function evaluations of a run, 'Nfe^{avg}'; the average overall time of a run (in seconds), 'T^{avg}'; the average number of local search calls, t_{local}^{avg}; and for each identified global and local solution, the average of the best f values obtained during the runs where the solution was located, 'f^{avg}'; the average number of function evaluations required by the HJ local search while converging to those best solutions, 'Nfe_{local}^{avg}'; and

the success rate (the percentage of runs that found that particular solution at least during one local search), 'SR'.

Table 1. MS 'HJ-filter' [11] *vs* Algorithm 1 based on the 'Interrupted HJ-filter'

Method	Solution		Problem 1	2	3	4
MS with 'HJ-filter' (in [11])		Nfe^{avg}	11513	13109	79892	4199
		T^{avg}	33	113	477	36
		t_{local}^{avg}	16.5	9.2	20.5	8.1
	global	f^{avg}	-6.666394	189.29356	-13.401916	2.000250
		Nfe_{local}^{avg}	590	1495	4929	458
		SR (%)	100	100	100	100
	local	f^{avg}	-5.000000	291.75820	-4.258899	2.236262
		Nfe_{local}^{avg}	519	2122	2986	527
		SR (%)	80	6.7	73.3	6.7
Algorithm 1 with 'Interrupted HJ-filter' ($\epsilon = 0.05$)		Nfe^{avg}	3110	9193	14596	628
		T^{avg}	6	42	36	4
		t_{local}^{avg}	9.1	12.5	20.0	2.1
	global	f^{avg}	-6.662584	189.29714	-13.401973	2.003090
		Nfe_{local}^{avg}	318	712	762	297
		SR (%)	100	100	100	100
	local	f^{avg}	-5.000000	291.02098	-4.258890	2.236161
		Nfe_{local}^{avg}	450	1327	480	372
		SR (%)	60	13.3	13.3	3.3
Algorithm 1 with 'Interrupted HJ-filter' ($\epsilon = 0.01$)		Nfe^{avg}	3759	12894	17479	628
		T^{avg}	8	59	44	4
		t_{local}^{avg}	9.6	15.5	21	1.9
	global	f^{avg}	-6.655440	189.29836	-13.401970	2.002982
		Nfe_{local}^{avg}	372	793	781	325
		SR (%)	100	100	100	100
	local	f^{avg}	-5.000000	291.69833	-4.258890	2.236149
		Nfe_{local}^{avg}	427	990	559	464
		SR (%)	60	10	23.3	6.7

Based on the results of the table, it can be concluded that the herein proposed multistart algorithm with the 'Interrupted HJ-filter' local search wins on efficiency when compared with the MS HJ-filter algorithm, in the sense that reductions on Nfe^{avg}, T^{avg} and t_{local}^{avg} are notorious. The algorithm was also tested with the smaller value $\epsilon = 0.01$ (the neighborhood radius on the continuous variables for the HJ interruption strategy, see (11)). From the corresponding results shown in Table 1, we may conclude that the algorithm is able to locate the local solutions more often but at a cost of more function evaluations, time and HJ local search calls.

5 Conclusions

The strategy of discarding randomly generated points that are sufficiently close to other generated and already used points, coupled with the interruption of the HJ-filter local search, for solving MINLP problems, have shown to be effective. The proposed stopping rule of the new multistart algorithm is able to detect multiple solutions without requiring a large number of local search calls. On the other hand, interrupting the local search procedure when an iterate that falls inside an ϵ-neighborhood of an already detected solution is found has also improved the overall efficiency of the algorithm.

The preliminary numerical results presented in this paper show that the combination of the componentwise normalized distance and region of attraction concepts in the multistart algorithm, as well as the interruption of the HJ-filter iterative search, make the derivative-free based multistart framework more competitive.

Future developments will be directed to the implementation of other mixed-integer point distance concepts, as well as the most popular Euclidean distance. Large-dimensional benchmark problems in the context of engineering applications will also be used to test the effectiveness of the proposed algorithm.

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. Abramson, M.A., Audet, C., Chrissis, J.W., Walston, J.G.: Mesh adaptive direct search algorithms for mixed variable optimization. Optim. Lett. 3, 35–47 (2009)
2. Belotti, P., Kirches, C., Leyffer, S., Linderoth, J., Luedtke, J., Mahajan, A.: Mixed-Integer Nonlinear Optimization. Acta Numer. 22, 1–131 (2013)
3. Burer, S., Letchford, A.: Non-convex mixed-integer nonlinear programming: a survey. Surveys in Operations Research and Management Science. 17, 97–106 (2012)
4. Gueddar, T., Dua, V.: Approximate multi-parametric programming based B&B algorithm for MINLPs. Comput. Chem. Eng. 42, 288-297 (2012)
5. Liuzzi, G., Lucidi, S., Rinaldi, F.: Derivative-free methods for mixed-integer constrained optimization problems. J. Optimiz. Theory App. 164(3), 933–965 (2015)
6. Boukouvala, F., Misener, R., Floudas, C.A.: Global optimization advances in mixed-integer nonlinear programming, MINLP, and constrained derivative-free optimization, CDFO. Eur. J. Oper. Res. 252, 701–727 (2016)
7. Hedar, A., Fahim, A.: Filter-based genetic algorithm for mixed variable programming. Numerical Algebra, Control and Optimization 1(1), 99–116 (2011)
8. Schlüter, M., Egea, J.A., Banga, J.R.: Extended ant colony optimization for non-convex mixed integer nonlinear programming. Comput. Oper. Res. 36, 2217–2229 (2009)

9. Lin, Y.C., Hwang, K.S.: A mixed-coding scheme of evolutionary algorithms to solve mixed-integer nonlinear programming problems. Comput. Math. Appl. 47, 1295–1307 (2004)
10. Abramson, M.A., Audet, C., Dennis Jr., J.E.: Filter pattern search algorithms for mixed variable constrained optimization problems. Pac. J. Optim. 3(3), 477-500 (2007)
11. Costa, M.F.P., Fernandes, F.P., Fernandes, E.M.G.P., Rocha, A.M.A.C.: Multiple solutions of mixed variable optimization by multistart Hooke and Jeeves filter method. Applied Mathematical Sciences 8(44), 2163–2179 (2014)
12. Fernandes, F.P., Costa, M.F.P., Fernandes, E.M.G.P., Rocha, A.M.A.C.: Multistart Hooke and Jeeves filter method for mixed variable optimization. In: Simos, T.E., Psihoyios, G., Tsitouras, Ch. (eds.) ICNAAM 2013. AIP Conf. Proc. vol. 1558, pp. 614–617 (2013)
13. Liao, T.W.: Two hybrid differential evolution algorithms for engineering design optimization. Appl. Soft Comput. 10(4), 1188–1199 (2010)
14. Voglis, C., Lagaris, I.E.: Towards "Ideal Multistart". A stochastic approach for locating the minima of a continuous function inside a bounded domain. Appl. Math. Comput. 213, 1404–1415 (2009)
15. Hooke, R., Jeeves, T.A.: Direct search solution of numerical and statistical problems. Journal on Associated Computation. 8, 212–229 (1961)
16. Kolda, T.G., Lewis, R.M., Torczon, V.: Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods. SIAM Rev. 45(3), 385–482 (2003)
17. Fletcher, R., Leyffer, S.: Nonlinear programming without a penalty function. Math. Program. 91, 239–269 (2002)
18. Audet, C., Dennis Jr., J.E.: A pattern search filter method for nonlinear programming without derivatives. SIAM J. Optimiz. 14(4), 980–1010 (2004)
19. Tsoulos, I.G., Stavrakoudis, A.: On locating all roots of systems of nonlinear equations inside bounded domain using global optimization methods. Nonlinear Anal.: Real World Appl. 11, 2465–2471 (2010)
20. Tsoulos, I.G., Lagaris, I.E.: MinFinder: Locating all the local minima of a function. Comput. Phys. Commun. 174, 166–179 (2006)
21. Lagaris, I.E., Tsoulos, I.G.: Stopping rules for box-constrained stochastic global optimization. Appl. Math. Comput. 197, 622–632 (2008)
22. Ryoo, H.S., Sahinidis, N.V.: Global optimization of nonconvex NLPs and MINLPs with applications in process design. Comput. Chem. Eng. 19(5), 551–566 (1995)