Branch and Bound Based Coordinate Search Filter Algorithm for Nonsmooth Nonconvex Mixed-Integer Nonlinear Programming Problems

Florbela P. Fernandes[1,3], M. Fernanda P. Costa[2,3], and Edite M.G.P. Fernandes[4]

[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,
mfc@math.uminho.pt
[3] Centre of Mathematics,
[4] Algoritmi Research Centre,
University of Minho, 4710-057 Braga, Portugal
emgpf@dps.uminho.pt

Abstract. A mixed-integer nonlinear programming problem (MINLP) is a problem with continuous and integer variables and at least, one nonlinear function. This kind of problem appears in a wide range of real applications and is very difficult to solve. The difficulties are due to the nonlinearities of the functions in the problem and the integrality restrictions on some variables. When they are nonconvex then they are the most difficult to solve above all. We present a methodology to solve nonsmooth nonconvex MINLP problems based on a branch and bound paradigm and a stochastic strategy. To solve the relaxed subproblems at each node of the branch and bound tree search, an algorithm based on a multistart strategy with a coordinate search filter methodology is implemented. The produced numerical results show the robustness of the proposed methodology.

Keywords: nonconvex MINLP, branch and bound, multistart, coordinate search, filter method

1 Introduction

A wide range of problems arising in practical applications, which involve both discrete decisions and nonlinear real-world phenomena, are modeled as mixed-integer nonlinear programming (MINLP) problems. Examples of practical applications modeled as MINLP appear in various areas, such as, process engineering [14], water, gas, energy and transportation networks [5]. For a review of a wide range of MINLP applications, see [3].

MINLP problems combine the combinatorial difficulty of optimizing over discrete variable sets with the challenge of handling nonlinear functions. They are the most general optimization problems, containing as special cases the mixed-integer linear programming (MILP) problem and the nonlinear programming (NLP) problem. This generality allows the modeling of a wide range of practical applications. When all functions involved in the problem are convex, the problem is a convex MINLP problem; otherwise it is a nonconvex one. Although MINLP problems are in general NP-hard, convex MINLPs are much easier to solve than nonconvex ones. This is due to the fact that the continuous relaxation of a convex MINLP problem, which is obtained by considering all the variables continuous, is itself convex and the computed solution being a global one provides a lower bound for the optimal solution of the MINLP. A variety of effective exact solution methods for convex MINLPs have been devised based on this fact. Unfortunately, when the MINLP is nonconvex, the continuous relaxation of a nonconvex MINLP is itself a global optimization problem, and therefore likely to be NP-hard. There is no guarantee that the computed solution of the continuous relaxation is a global optimum. The situation gets even worse when the MINLP model involves nonsmooth functions. For example, when the objective and constraints are provided as black-boxes. In this case, the MINLP is a nonsmooth and nonconvex problem.

There are different strategies to solve MINLP problems and a great majority rely on a branch and bound paradigm. One of the most well-known solvers for MINLP problems is the BARON (Branch And Reduce Optimization Navigator) solver [23]. For a review of the available solvers, we refer the reader to [3]. Recently, the nonconvex MINLP research area became more interesting due to its range of applications and the new techniques for globally solving NLP problems [8]. In the last 10–15 years, innovative heuristic type algorithms have also appeared: genetic algorithm [15], ant colony [22], pattern search algorithms [1], multistart Hooke and Jeeves algorithm [12], and differential evolution [19].

In this work we develop a derivative-free methodology to solve nonsmooth nonconvex MINLP problems. The proposed method is based on a branch and bound (BB) scheme, and the NLP problems that appear in the BB tree search are solved to optimality by a derivative-free global method that is based on a multistart algorithm coupled with a coordinate search filter method, initially developed in [11]. The therein called MCSFilter method is able to find multiple minima of a nonconvex NLP problem, and consequently the global one. The MCSFilter method is appropriate to solve nonsmooth problems since neither analytical nor numerical derivatives are required.

Our BB paradigm for nonsmooth nonconvex MINLP problems, henceforth denoted by BBMCSFilter, has been implemented in MatLab and the numerical experiments with a benchmark set of problems show that the method is competitive with others of the same type.

The remaining part of the paper is organized as follows. In Section 2 the MINLP problem is described and in Section 3 the proposed BBMCSFilter algorithm is presented and discussed. Section 4 reports on the computational ex-

periments carried out using a benchmark set of problems (from the engineering field) from the *MINLPLib* library available online [6] and we conclude the paper with Section 5.

2 The MINLP Problem

The problem to be solved is of the form

$$\begin{array}{c} \min f(x,y) \\ \text{subject to } g_j(x,y) \leq 0,\ j \in J \\ x \in X,\ y \in Y \end{array} \tag{1}$$

where f is the objective function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$, g_j, $j \in\ J = \{1, \cdots, m\}$, are the constraint functions and J is the index set of the g functions. The continuous variables are represented by the vector x, with $X \subset \mathbb{R}^{n_c}$ being the set of simple bounds on x:

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

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

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

with $l_y, u_y \in \mathbb{Z}^{n_i}$. The parameter $n = n_c + n_i$ represents the total number of variables. We will assume that at least one of the functions is nonlinear since this study aims to focus on MINLP problems. In the present study we are particularly interested in nonsmooth and nonconvex MINLPs. With this kind of problems two major issues have to be addressed:

- one is related with the integrality of some variables;
- the other is concerned with the lack of smoothness and convexity of the functions.

To settle the first issue, a BB paradigm is used. A brief description of the main ideas behind this classical technique is presented in the next section. The second issue is crucial since the nonconvex NLP relaxed problems that arise in the nodes of the BB tree search have multiple minima, some global and others local. When a minimum is found it is not possible to know if it is a global or a local one, until all the minima are computed and compared. Unless, a global search exact method with guaranteed convergence to a global minimum is used when solving the nonconvex NLP problems. Some heuristic methods that do not use derivative information by methodically searching the space can also guarantee convergence to global solutions with probability one. Both real analysis pointwise convergence and stochastic convergence are appropriate when convergence to a global minimum is required.

3 The BBMCSFilter Method

As previously stated, the BB paradigm and the MCSFilter method constitute the two major parts of the method. First, a summary of the BB paradigm is presented.

3.1 BB Method

The BB method was initially devised for MILP problems but has been applied ever since to MINLPs too. The first reference to nonlinear MINLP problems appears in 1965 [7]. The extension to convex MINLP problems is an easy task since the minimum of the convex relaxed NLP problem is also a global one [18]. BB methodology can be explained accordingly with a tree search [3,4,20]. We first define a continuous relaxation.

Definition 1. *Consider the (convex or nonconvex) MINLP problem* (1)*. A continuous relaxation of problem* (1) *is:*

$$\begin{aligned} &\min f(x,y) \\ &\textit{subject to } g_j(x,y) \leq 0,\ j \in J \\ &\qquad x \in X \subset \mathbb{R}^{n_c},\ y \in Y_{\mathbb{R}} \subset \mathbb{R}^{n_i} \end{aligned} \tag{2}$$

with $Y_{\mathbb{R}} = \{y \in \mathbb{R}^{n_i} : l_y \leq y \leq u_y\}$*, meaning that all variables are real numbers.*

At the beginning of the process all the integer variables are relaxed and the relaxed NLP problem (2) is solved. This is the first node (also known as root) of the tree search. After solving this problem, if all integer variables take integer values at the solution then this solution also is the solution of the MINLP problem. However, in general, at this stage, some integer variables take non-integer values. Then, a tree search is performed in the space of the integer variables. Branching generates new NLP subproblems by adding new simple bounds to the new relaxed NLP subproblems. Next, a new subproblem is selected and solved. The solution of each relaxed subproblem provides a lower bound, $\underline{f}$, for the descent nodes of the tree (or the child nodes). The integer solutions (at some nodes of the tree) provide upper bounds, $\overline{f}$, on the optimal integer solution. This process of branching at each node continues until:

- the lower bound exceeds the best known upper bound;
- the NLP subproblem is infeasible;
- the solution provides integer values for the integer variables, thus providing an upper bound.

The BB algorithm stops when there are no more nodes to explore. In a BB technique there are two crucial components. One is related with choosing a good branching variable. The main goal is to choose the branching variable that minimizes the size of the tree that needs to be searched. However, this is not practical and the selection of a branching variable that maximizes the increase in the lower bound at a node is a good alternative. The other component is

related with the choice of which relaxed problem (node) should be solved next. The main goal here is to find a good feasible solution as quickly as possible in order to reduce the upper bound, and to prove optimality of the current best integer solution by increasing the lower bound as quickly as possible.

Our main contribution is related with solving the relaxed NLP subproblems that appear in the multiple nodes of the tree search by a derivative-free multistart method so that a global solution of the relaxed subproblem (2) is obtained. This is a challenging task that is accomplished by the MCSFilter method.

3.2 MCSFilter Method for the Relaxed NLP Subproblems

In each node of the tree search a nonsmooth nonconvex NLP subproblem is required to be globally solved. To obtain the global optimal solution of these relaxed problems is crucial since fathoming /eliminating nodes of the BB tree search can no longer be made if the solution is not a global one. The process of eliminating nodes is very important because if all subproblems need to be solved the BB scheme will be very time consuming. So, it is necessary to eliminate some nodes as the result of a bounding scheme with $\underline{f}$ and $\overline{f}$.

The MCSFilter method which is used to solve the relaxed nonsmooth nonconvex NLP subproblems at each node of the tree has nice features:

- does not make use of any derivative information;
- solves nonconvex NLP problems;
- finds multiple solutions, global as well as the local ones;
- is based on a multistart strategy with regions of attraction coupled with a coordinate search filter methodology;
- is, relatively, simple to implement.

Since the variables of the relaxed problem are all real, hereafter we use z to denote the vector of all the n variables, the vectors l and u of the set $\mathbb{R}^n$ to denote the lower and upper bounds respectively, of all the variables. Thus, problem (2) is now formulated as:

$$\begin{array}{c} \min f(z) \\ \text{subject to } g_j(z) \leq 0,\ j \in J\,. \\ z \in [l,u] \subset \mathbb{R}^n \end{array} \tag{3}$$

This method is based on a multistart strategy. To explore the search space more effectively, the multistart algorithm sequentially generates points

$$z_i = l_i + \lambda\,(u_i - l_i)\,,\ i = 1,\ldots,n,$$

where λ is a random number uniformly distributed in $[0,1]$, and applies a local search aiming to converge to the optimal solutions of the problem (3). Since this simple strategy may converge to some or all the minimizers over and over again, the implemented multistart strategy incorporates the concept of regions of attraction of minimizers to avoid convergence to the already computed solutions [17,24]. The region of attraction of a minimizer, z_*^i, associated with a local search procedure $\mathbf{L}$, is defined as:

$$A_i \equiv \left\{ z \in [l,u] : \mathbf{L}(z) = z_*^i \right\}, \tag{4}$$

where $\mathbf{L}(z)$ is the minimizer obtained when the local search procedure $\mathbf{L}$ starts at point z. Computing the region of attraction A_i of a minimizer z_*^i is not an easy task. Alternatively, a stochastic procedure may be used to estimate the probability, p, that a sampled point will not belong to the union of the regions of attraction of already computed minimizers. This probability is estimated similarly to [24] but using instead forward differences to estimate the gradient of f [10]. Different stopping rules have been tested in a multistart algorithm context, see [17]. The algorithm should stop when all minima have been identified with certainty, and it should not require a large number of calls to the local search procedure to decide that all minima have been found. The rule used in our implementation gives an estimate of the fraction of uncovered space [17]. A formal description of the multistart algorithm for solving the nonsmooth nonconvex NLP subproblem (2), based on the CSFilter method as the procedure for the local search (4), is presented in Algorithm 1.

Algorithm 1 Multistart algorithm

Set $Z^* = \emptyset$ (contains the computed minimizers), $k = 1$, $t = 1$;
Randomly generate $z \in [l, u]$;
Compute $z_*^1 = \mathbf{L}(z)$ using Algorithm 2, set $Z^* = Z^* \cup z_*^1$, define A_1;
repeat
  Randomly generate $z \in [l, u]$;
  if z has a high probability of being outside $\bigcup_{i=1}^{k} A_i$ **then**
    Compute $z_* = \mathbf{L}(z)$ using Algorithm 2, set $t = t + 1$;
    if $z_* \notin Z^*$ **then**
      Set $k = k + 1$, $z_*^k = z_*$, $Z^* = Z^* \cup z_*^k$, compute A_k;
    else
      Update A_l (region of attraction of the nearest to z_* minimizer)
    end if
  end if
until the stopping rule is satisfied

Local CSFilter Method. The CSFilter method that combines a derivative-free local search technique with a filter methodology [13] is used as the search procedure $\mathbf{L}$, to compute a minimizer z_* starting from a sampled point $z \in [l, u]$ [11]. The classical coordinate search, which is a direct search method [16], and the filter methodology are combined to construct a local search procedure that does not require any derivative information. The filter methodology is implemented to handle the constraints by forcing the local search towards the feasible region. The main idea behind the filter approach is to interpret problem (3) as a bi-objective optimization problem aiming to minimize both the objective function $f(z)$ and a nonnegative continuous aggregate constraint violation function $\theta(z)$ defined by

$$\theta(z) = \|g(z)_+\|^2 + \|(l - z)_+\|^2 + \|(z - u)_+\|^2 \tag{5}$$

where $v_+ = \max\{0, v\}$. Therefore, the problem is rewritten as a bi-objective optimization problem of the form

$$\min_z \left(\theta(z), f(z)\right). \tag{6}$$

The filter technique incorporates the concept of nondominance, present in the field of multi-objective optimization, to build a filter that is able to accept trial approximations if they improve the constraint violation or objective function value [1,2,13]. A filter $\mathcal{F}$ is a finite set of points z, corresponding to pairs $(\theta(z), f(z))$, none of which is dominated by any of the others. A point z is said to dominate a point z' if and only if $\theta(z) \leq \theta(z')$ and $f(z) \leq f(z')$.

We now discuss the implemented CSFilter algorithm for the local procedure [11]. The pseudo-code for the local filter-based coordinate search algorithm is presented below in Algorithm 2. At the beginning, and every time the procedure

Algorithm 2 CSFilter algorithm

Require: z (sampled in the Multistart algorithm) and parameter values; set $\tilde{z} = z$, $z_{\mathcal{F}}^{inf} = z$, $t = \tilde{z}$;
Initialize the filter;
Set $\alpha = \min\{1, 0.05\frac{\sum_{i=1}^{n} u_i - l_i}{n}\}$;
repeat
  Compute the trial approximations $t_c^i = \tilde{z} + \alpha d_i$, for all $d_i \in \mathcal{D}_\oplus$;
  repeat
    Check acceptability of trial points t_c^i using (7) and (8);
    if there are some t_c^i acceptable by the filter **then**
      Update the filter;
      Choose t_c^{best};
      Set $t = \tilde{z}$, $\tilde{z} = t_c^{best}$; update $z_{\mathcal{F}}^{inf}$ if appropriate;
    else
      Compute the trial approximations $t_c^i = z_{\mathcal{F}}^{inf} + \alpha d_i$, for all $d_i \in \mathcal{D}_\oplus$;
      Check acceptability of trial points t_c^i using (7) and (8);
      if there are some t_c^i acceptable by the filter **then**
        Update the filter;
        Choose t_c^{best};
        Set $t = \tilde{z}$, $\tilde{z} = t_c^{best}$; update $z_{\mathcal{F}}^{inf}$ if appropriate;
      else
        Set $\alpha = \alpha/2$;
      end if
    end if
  until new trial t_c^{best} is acceptable
until the stopping condition is satisfied

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.

Let $\mathcal{D}_\oplus$ denote the set of $2n$ coordinate directions, defined as the positive and negative unit coordinate vectors, $\mathcal{D}_\oplus = \{e_1, e_2, \ldots, e_n, -e_1, -e_2, \ldots, -e_n\}$. The search begins with a central point, the current approximation $\tilde{z}$, as well as $2n$ trial approximations $t_c^i = \tilde{z} + \alpha d_i$, for $d_i \in \mathcal{D}_\oplus$, where $\alpha > 0$ is a step size. The constraint violation value and the objective function value of all $2n+1$ points are computed. If some trial approximations improve over $\tilde{z}$, reducing θ or f by a certain amount (see equations (7) and (8)), and are acceptable by the filter, then the best of these non-dominated trial approximations, t_c^{best}, is selected and the filter is updated (adding the corresponding entries to the filter and removing any dominated entries). Then, this best approximation becomes the new central point in the next iteration, $\tilde{z} \leftarrow t_c^{best}$. If, on the other hand, all trial approximations t_c^i are dominated by the current filter, then all t_c^i are rejected, and a restoration phase is invoked.

To avoid the acceptance of a point t_c^i, or the corresponding pair $\big(\theta(t_c^i), f(t_c^i)\big)$, that is arbitrary close to the boundary of $\mathcal{F}$, the trial t_c^i is considered to improve over $\tilde{z}$ if one of the conditions

$$\theta(t_c^i) < (1 - \gamma_\theta)\ \theta(\tilde{z}) \ \text{ or } \ f(t_c^i) \leq f(\tilde{z}) - \gamma_f\, \theta(\tilde{z}) \tag{7}$$

holds, for fixed constants $\gamma_\theta, \gamma_f \in (0, 1)$. However, the filter alone cannot ensure convergence to optimal points. For example, if a sequence of trial points satisfies $\theta(t_c^i) < (1 - \gamma_\theta)\ \theta(\tilde{z})$ then it could converge to an arbitrary feasible point. Therefore, when $\tilde{z}$ is nearly feasible, $\theta(\tilde{z}) \leq \theta_{\min}$, for a small positive $\theta_{\min}$, the trial approximation t_c^i has to satisfy only the condition

$$f(t_c^i) \leq f(\tilde{z}) - \gamma_f\, \theta(\tilde{z}) \tag{8}$$

instead of (7), in order to be acceptable.

The best non-dominated trial approximation is selected as follows. The best point t_c^{best} of a set $T = \{t_c^i : t_c^i = \tilde{z} + \alpha d_i, d_i \in \mathcal{D}_\oplus\}$ is the point that satisfies one of two following conditions:

i) if there are some feasible points in T, t_c^{best} is the point that has the smallest objective function value among the feasible points;
ii) if there are no feasible points in T, t_c^{best} is the point that has the smallest constraint violation among the non-dominated infeasible points.

We remark that the filter is updated whenever the trial approximations t_c^i verify conditions (7) or (8) and are non-dominated.

When it is not possible to find a non-dominated best trial approximation, and before declaring the iteration unsuccessful, a restoration phase is invoked. In this phase, the most nearly feasible point in the filter, $z_{\mathcal{F}}^{inf}$, is recuperated and the search along the $2n$ coordinate directions is carried out about it to find the set $T = \{t_c^i : t_c^i = z_{\mathcal{F}}^{inf} + \alpha d_i, d_i \in \mathcal{D}_\oplus\}$. If a non-dominated best trial approximation is found, this point becomes the central point of the next iteration and the iteration is successful. Otherwise, the iteration is unsuccessful, the search returns back to the current $\tilde{z}$, the step size α is halved, and new

$2n$ trial approximations t_c^i are generated around $\tilde{z}$. If a best non-dominated trial approximation is still not found, the step size reduction is repeated since another unsuccessful iteration has occurred. When α falls below $\alpha_{\min}$ [16], the algorithm stops, where $\alpha_{\min}$ is a small positive tolerance.

4 Numerical Results

To analyze the performance of the BBMCSFilter algorithm, a set of 23 test problems is used (see Table 1). The set contains inequality and equality constrained problems. Almost all problems are selected from published literature in several different engineering fields [12,19,21]. The BBMCSFilter method was coded in MatLab. Crucial BB algorithm specifications are:

- the branching variable is chosen by a simple heuristic that picks up the variable which maximizes the increase in the lower bound at that node, i.e.,

$$\arg \max_{i=1,\ldots,n_i} \{|\underline{f}_i - \underline{f}|\},$$

  where $\underline{f} = f(x', y')$ is the solution of the NLP subproblem ($x' \in X, y' \in Y_{\mathbb{R}}$) and $\underline{f}_i = f(x', y_i)$, for $i = 1, \ldots, n_i$, being $y_i = [y'_i]_{\mathbb{R}}$ the scalar rounding of y'_i to the nearest integer;
- the next subproblem to be solved is selected by a depth-first strategy, also known as *last-in-first-out*, so that the upper bounds are found as early as possible.

The results were obtained in a PC with an Intel Core i7-2600 CPU (3.4GHz) and 8 GB of memory. In the CSFilter method, we set after an empirical study: $\gamma_\theta = \gamma_f = 10^{-5}$, $\alpha_{\min} = 10^{-3}$, $\theta_{\min} = 10^{-6}$ and $\theta_{\max} = 10^2 \max\{1, 1.25\theta(z)\}$, where z is the point on entry in the local search. A maximum of 10 points were generated in the multistart algorithm and each problem was solved 30 times.

Table 1 shows the numerical results produced by the proposed BBMCSFilter method. The columns in the table show:

- the known global optimum, f^*;
- the average value of the obtained objective function values (over the 30 runs), 'f_{avg}';
- the standard deviation of obtained function values, 'SD';
- the average number of function evaluations, 'nfe_{avg}';
- the average CPU time (in seconds), 'T_{avg}';
- the average number of nodes, '$Nodes_{avg}$';
- the successful rate (percentage of runs that found a solution within an error of 10^{-2} of the known global optimal solution), 'SR' (%).

We may conclude from Table 1 that our algorithm found the global optimal solution in all the 30 runs when solving 17 of the 23 problems. The computed solutions are of high quality and the f_{avg} for all problems are very close to the known minimum. We remark that our values of f_{avg} under f^* (on problems f4,

Table 1. Numerical results produced by BBMCSFilter

Problem	f^*	f_{avg}	SD	T_{avg}	nfe_{avg}	$Nodes_{avg}$	SR
f1	2	2.00082	3.6E-04	10.0	3530	1.1	100
f2	2.124	2.124590	1.4E-06	1.5	1259	1.0	100
f3	1.07654	1.081640	8.1E-03	3.8	5274	3.0	87
f4	99.245209	99.239635	1.0E-07	0.2	670	1.0	100
f5	3.557463	3.560848	2.0E-03	59.3	76775	11.0	97
f6	4.579582	4.582322	9.3E-04	54.9	75413	10.7	100
f7	-17	-16.998054	2.3E-03	9.5	4296	1.8	100
f8	-32217.4	-32217.42778	0.0E00	3.2	18051	5.7	0
f9	7.6671801	7.667583	9.5E-04	30.3	28090	3.9	100
f10	-2.4444	-2.444444	0.0E00	2.4	2736	5.0	100
f11	3.2361	3.236121	8.7E-05	18.8	41635	10.0	100
f12	1.125	1.125115	2.9E-04	42.3	7770	2.8	100
f13	87.5	87.507043	1.7E-02	252.5	41852	3.0	90
f14	-6.666667	-6.666131	1.8E-04	0.5	1122	1.0	100
f15	-5.6848	-5.651952	2.6E-02	3567.4	393345	59.8	30
f16	2.000	2.000000	0.0E00	52.7	29847	4.9	100
f17	3.4455	3.445808	2.1E-04	18.2	5469	6.0	100
f18	2.2000	2.200000	0.0E00	8.6	11182	3.7	100
f19	6.0098	6.010714	6.6E-04	21.1	37132	5.3	100
f20	-17.0000	-16.994605	5.5E-03	52.5	27149	1.1	80
f21	-4.514202	-4.513448	6.8E-04	84.4	50146	4.6	100
f22	-13.401904	-13.401930	3.6E-04	57.8	84790	14.0	100
f23	-1.08333	-1.083245	5.4E-05	2.6	2458	1.0	100

Table 2. Numerical results obtained with BBGA

Problem	f_{avg}	SD	T_{avg}	nfe_{avg}	SR
f1	2.00	6.8E-05	2.5	10481	100
f2	2.1246	2.1E-04	2.4	11527	100
f3	1.078992	2.4E-03	3.3	13635	20
f5	3.564265	7.6E-03	13.5	47282	23
f6	4.5987	2.9E-02	13.7	46678	53
f7	-17	1.8E-04	3.4	14292	100
f8	-32217.4	1.5E-11	1.3	5220	100
f15	-5.684	1.9E-03	65.0	247055	87
f16	2.000	8.9E-07	3.2	12808	67
f17	3.446	2.0E-05	7.8	23489	100
f18	2.200	5.8E-05	4.5	15290	87

f8 and f22) are due to the slight constraint violation allowed by the algorithm when it stops. The values of SD are equal to zero on problems f8, f10, f16 and f18, and are moderate, ranging from 2.6E-02 to 1.0E-07, on the remaining problems, showing the consistency of the algorithm. The number of function evaluations and the time required by the algorithm are higher than one could expect.

To compare our results with those of a BB scheme that uses the genetic algorithm solver from MatLab^{TM} Optimization Toolbox, to solve the nonsmooth nonconvex NLP relaxed subproblems, we include Table 2 with the results available in [9]. The columns in the table list the values of f_{avg}, SD, T_{avg}, nfe_{avg} and SR. The comparison with the therein called BBGA method involves 11 problems from the previous set of 23 problems. We conclude from the tables that BBMCSFilter method outperforms BBGA in terms of criteria SD and SR, but is outweighted by BBGA in criterion T_{avg} on all tested problems but one. As far as nfe_{avg} is concerned, BBMCSFilter is better than BBGA in 6 of the tested problems.

Finally, we compare our results with those reported in [19]. This paper presents two hybrid differential evolution (DE) algorithms. One, enhances a modified DE (MDE) algorithm with a local search operator, therein denoted by `MDE-LS`; the other, adds a second metaheuristic – the harmony search algorithm – to cooperate with the MDE algorithm, and is denoted by `MDE-IHS`. The results obtained from the `MDE` algorithm from which the other two hybrids were created are also reported. These results are shown in Table 3. When a comparison is made between BBMCSFilter and `MDE` it is possible to state that our method performs better than `MDE`, relatively to f_{avg} and SD. The BBMCSFilter method requires in general more function evaluations than `MDE` but on the other hand the quality of the solutions is higher. When comparing with the results of `MDE-LS` algorithm, we observe that BBMCSFilter produces better values of f_{avg} and SD in 7 of the 9 common problems (see Table 3). However, the values of nfe_{avg} are larger with BBMCSFilter on 6 problems. Observing the results obtained by `MDE-IHS` algorithm in Table 3, we may conclude that this method wins over BBMCSFilter in number of function evaluations, although the quality of the solutions in terms of f_{avg} and SD are lower than that of BBMCSFilter in 5 of the 9 common problems.

5 Conclusions

We have presented a method to solve nonsmooth nonconvex MINLP problems, the BBMCSFilter method, and showed that the BB paradigm coupled with a stochastic multistart method based on the classical coordinate search for a local exploitation of a minimizer is effective and worthy of further research. The MCSFilter method is used to solve the nonsmooth nonconvex NLP relaxed subproblems that appear in the multiple nodes of the BB tree search. The filter methodology aims to promote convergence to feasible and optimal solutions.

A set of benchmark problems was used to test the algorithm performance and the results are very promising. We observed that the proposed method consistently located the global optimum of all problems. The quality of the solutions is good and one can state that the BBMCSFilter behaves generally better than the other methods in comparison.

Table 3. Numerical results obtained from [19]

	MDE			MDE-LS			MDE-IHS		
Problem	f_{avg}	SD	nfe_{avg}	f_{avg}	SD	nfe_{avg}	f_{avg}	SD	nfe_{avg}
f1	2.009348	4.4E-02	1075	2.00000	0.0E00	1430	2.000001	0.0E00	3297
f2	2.167894	1.3E-01	827	2.124574	7.1E-05	653	2.124604	7.6E-05	1409
f4	99.240933	1.4E-03	403	99.241271	1.8E-03	684	99.512250	1.5E00	449
f5	3.599903	5.9E-02	37739	3.564912	2.9E-02	20116	3.561157	8.4E-03	27116
f6	4.661414	3.1E-01	7688	4.579595	3.0E-06	13023	4.579599	5.0E-06	14518
f8	-32217.427262	2.8E-03	1240	-32217.427106	3.7E-03	1955	-32217.427780	0.0E00	493
f9	7.918619	4.8E-02	96718	7.883841	9.9E-02	93524	7.848896	1.2E-01	83442
f12	1.124453	7.6E-02	30986	1.099805	5.6E-02	25766	1.094994	5.3E-02	22146
f13	89.879034	2.8E00	7777	88.230145	1.9E00	4436	87.497550	2.1E-03	5359

One problematic issue of the proposed BBMCSFilter method is related to the required large number of function evaluations which is a consequence of the set of search directions $\mathcal{D}_{\oplus}$ inside the MCSFilter method. For large dimensional problems, the computational effort in terms of number of function evaluations and consequently CPU time greatly increases with n. This issue is to be deepen in the near future.

Acknowledgments. The authors wish to thank two anonymous referees for 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-OE/MAT/UI0013/2014 and PEst-OE/EEI/UI0319/2014.

References

1. 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)
2. Audet, C., Dennis Jr., J.E.: A pattern search filter method for nonlinear programming without derivatives. SIAM J. Optimiz. 14(4), 980–1010 (2004)
3. Belotti, P., Kirches, C., Leyffer, S., Linderoth, J., Mahajan, A.: Mixed-Integer Nonlinear Optimization. Acta Numer. 22, 1–131 (2013)
4. Belotti, P., Lee, J., Liberti, L., Margot, F., Wächter, A.: Branching and bound tightening techniques for non-convex MINLP. Opt. Methods Softw. 24, 597-634 (2009)
5. Burer, S., Letchford, A.: Non-convex mixed-integer nonlinear programming: a survey. Surveys in Operations Research and Management Science. 17, 97–106 (2012)
6. Bussieck, M.R., Drud, A.S., Meeraus, A.: Minlplib – a collection of test models for mixed-integer nonlinear programming. INFORMS J. Comput. 15(1), 1-5 (2003) URL http://www.gamsworld.org/minlp/minlplib/minlpstat.htm
7. Dakin, R.J.: A tree search algorithm for mixed integer programming problems. Comput. J. 8, 250–255 (1965)
8. D'Ambrosio, C., Frangioni, A., Liberti, L., Lodi, A.: A storm of feasibility pumps for nonconvex MINLP. Math. Program., Ser. B, 136(2), 375–402 (2012)
9. Fernandes, F.P.; Costa, M.F.P., Fernandes, E.M.G.P.: Assessment of a hybrid approach for nonconvex constrained MINLP problems. In: Proceedings of the 2011 International Conference on Computational and Mathematical Methods in Science and Engineering, pp. 484–495 (2011)
10. Fernandes, F.P., Costa, M.F.P., Fernandes, E.M.G.P.: A derivative-free filter driven multistart technique for global optimization. In: ICCSA 2012 Part III, Lect. Notes Comput. Sc., Vol. 7335, Murgante, B. et al. (Eds.) 103–118 (2012)
11. Fernandes, F.P., Costa, M.F.P., Fernandes, E.M.G.P.: Multilocal programming: a derivative-free filter multistart algorithm. In: ICCSA 2013 Part I, Lect. Notes Comput. Sc., Vol. 7971, Murgante, B. et al. (Eds.) 333–346 (2013)
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. International Conference of Numerical Analysis and Applied Mathematics, T.E. Simos, G. Psihoyios and Ch. Tsitouras (Eds.), AIP Conf. Proc. Vol. 1558, 614–617 (2013)

13. Fletcher, R., Leyffer, S.: Nonlinear programming without a penalty function. Math. Program. 91, 239–269 (2002)
14. Gueddar, T., Dua, V.: Approximate multi-parametric programming based B&B algorithm for MINLPs. Comput. Chem. Eng. 42, 288-297 (2012)
15. Hedar, A., Fahim, A.: Filter-based genetic algorithm for mixed variable programming. Numerical Algebra, Control and Optimization. 1(1), 99–116 (2011)
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. Lagaris, I.E., Tsoulos, I.G.: Stopping rules for box-constrained stochastic global optimization. Appl. Math. Comput. 197, 622–632 (2008)
18. Leyffer, S.: Integrating SQP and branch-and-bound for mixed integer nonlinear programming. Comput. Optim. Appl. 18, 295–309 (2001) 2001.
19. Liao, T. W.: Two hybrid differential evolution algorithms for engineering design optimization. Appl. Soft Comput. 10, 1188–1199 (2010)
20. Liberti, L., Mladenović, N., Nannicini, G.: A recipe for finding good solutions to MINLPs. Math. Prog. Comp. 3, 349–390 (2011)
21. 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)
22. 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)
23. Tawarmalani, M., Sahinidis, N.V.: A polyhedral branch-and-cut approach to global optimization, Math. Program. 103(2), 225–249 (2005)
24. 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)