Noname manuscript No.
(will be inserted by the editor)

An algorithmic framework based on primitive directions and nonmonotone line searches for black box problems with integer variables

Giampaolo Liuzzi · Stefano Lucidi · Francesco Rinaldi

Received: date / Accepted: date

Abstract In this paper, we develop a new algorithmic framework that handles black box problems with integer variables. The strategy included in the framework makes use of specific search directions (so called primitive directions) and a suitably developed nonmonotone line search, thus guaranteeing a high level of freedom when exploring the integer lattice. We first describe and analyze a version of the algorithm that tackles problems with bound constraints. Then, we combine it with a penalty approach in order to solve problems with black box constraints. In both cases we prove finite convergence to a suitably defined local minimum of the problem (convergence to a global minimum can eventually be obtained by properly modifing a few instructions in the scheme). We report extensive numerical experiments based on a testbed of both bound constrained and generally constrained problems. We show the efficiency and robustness of the method when compared to two state-of-the-art solvers for black box optimization.

Keywords Derivative Free Optimization · Black Box Problems · Integer variables · Nonmonotone line search

Mathematics Subject Classification (2010) 90C56 · 90C10 · 90C30

Giampaolo Liuzzi
Consiglio Nazionale delle Ricerche, Istituto di Analisi dei Sistemi ed Informatica "A. Ruberti"
Via dei Taurini 19, 00185 Rome, Italy
E-mail: giampaolo.liuzzi@iasi.cnr.it

Stefano Lucidi
"Sapienza" Università di Roma, Dipartimento di Ingegneria Informatica Automatica e Gestionale "A. Ruberti"
Via Ariosto 25, 00185 Rome, Italy
E-mail: lucidi@dis.uniroma1.it

Francesco Rinaldi
Dipartimento di Matematica, Università di Padova
Via Trieste, 63, 35121 Padua, Italy
E-mail: rinaldi@math.unipd.it

1 Introduction

Many real-world problems coming from different fields (e.g., Engineering, Economics, Biology) can be modeled using black box functions. Those functions represent the experimentally obtained behavior of a system and in practice are given by means of specific simulation tools, hence no internal or analytical knowledge for the functions is provided. Furthermore, evaluating the function at a given point is usually an expensive task in terms of computational resources, and only a limited budget of evaluations is available for the optimization. We also need to keep in mind that noise and discontinuities are pretty common in this context and make the problems very hard to solve. On top of that, real systems are often described by means of non-relaxable integer variables (e.g., number of nurses in a ward, number of coils in a magnet, and so on) that need to be properly handled. Indeed, when dealing with this kind of problems, there is no easy way to get lower bounds for evaluating the quality of the generated solutions. Thus, it is neither possible to cut away part of the feasible set nor to get an optimality gap for the solutions (like exact algorithmic frameworks in integer programming usually do).
In general, when exact methods cannot be applied, heuristic ones are considered. *Explorative search methods* (like, e.g., variable neighborhood search, iterated local search), which look for a new solution in some neighborhood of a given point and perturb point and/or neighborhood when specific conditions are met, and *population based methods* (like, e.g., genetic algorithms), which use a set (i.e., a population) of points at each iteration to generate new solutions, represent classic heuristic approaches for problems with integer variables (see, e.g., [6] for further details). However, those two classes of approaches may not be suitable for optimization problems with computationally expensive objective and constraint functions because a large number of evaluations is commonly needed to find good solutions.
So, the goal here is basically coming up with a method that explores the integer lattice in the best possible way with the information gathered around (i.e., a method that reduces as much as possible the objective function value without wasting the budget of function evaluations).

In the paper we then consider the following optimization problem:

$$\begin{aligned} &\min f(x) \\ &s.t.\ x \in \mathcal{C} \cap \mathbb{Z}^n, \end{aligned} \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a black box function (i.e., the mathematical representation of $f(x)$ is *not available*) and $\mathcal{C} \subset \mathbb{R}^n$, $\mathcal{C} \neq \emptyset$ is a non-empty compact set (whose description usually includes black box constraints). We further assume that black box functions are computationally expensive in this framework. Methods for dealing with this kind of problems can be essentially divided into two classes:

1. **Direct search methods:** at each iteration, the objective function is sampled over a specific set of points and the best point (in terms of objective function value) is chosen as the new iterate;
2. **Model based methods:** at each iteration, a model of the objective function is built and a new point that minimizes the model is eventually picked;

the interested reader can refer to [7] for further details. Methods of the direct search type can be further divided into different groups depending on the way the function is sampled over the integer lattice. Hence we can either consider methods that use a *stencil* (i.e., a predetermined set of search directions with a common fixed stepsize, see, e.g., [3–5, 14]) or methods that use a *skewed stencil* (i.e., a predetermined set of search directions with dynamically changing stepsizes, see, e.g., [12, 13]) whose shape is modified depending on a line search procedure. Recently, a more sophisticated way to handle the integer variables, which combines fixed stencils with tree-based searches, was proposed in [19]. Methods belonging to the second class can be further classified depending on the model used. We thus have algorithms that embed surrogate (see, e.g., [16, 17]) or quadratic models (see, e.g., [18]).
If we focus on direct search methods, we can easily notice that the search of points that monotonically reduce the objective function, together with the use of a predetermined set of directions for exploring the integer lattice can cause the algorithm to get stuck in a local minimum pretty soon in the optimization process. Even the use of a line search that dynamically changes the skewed stencil might not be enough in order to escape those points. It is actually easy to build examples where both strategies cannot move away from the starting point.
In this paper we then define a new algorithmic framework of the direct search type that tries to overcome these issues. The three main features that characterize our strategy are the following:

- stencil enrichment by means of a suitable set of search directions (the so called *primitive directions*) as soon as the algorithm gets stuck;
- a nonmonotone version of the line search defined in [12, 13] for getting more freedom when changing the shape of the stencil;
- a simple penalty approach, similar to the one described in [13], for handling the case when set $\mathcal{C}$ is described by means of black box functions.

The use of a nonmonotone acceptance rule is very important in this context since it improves robustness and efficiency of our framework. Indeed, trying to get a new point that strictly reduces the objective function, like we do when using a monotone line search, might either get small movements along the search direction (especially when dealing with an objective function that is noisy or with steep sided valleys) or require the generation of many primitive directions in order to escape a point (this usually happens when the objective function is locally "flat ").
Nonmonotone acceptance rules have already been used in derivative-free continuous optimization (see, e.g., [8–10]). Anyway, to the best of our knowledge,

this is the first time that a nonmonotone discrete line search is embedded into an algorithmic framework that handles black-box IP problems.

About the theoretical results reported in the paper, we prove finite convergence of the algorithmic framework to a suitably defined local minimum of Problem (1) when $\mathcal{C}$ is:

- a structured set (i.e., simple bounds);
- a more general set described by black box functions (in this case we need to assume that some sort of constraint qualification condition holds for the problem).

An extensive numerical analysis on a large testbed of both bound constrained and generally constrained problems is also reported. More specifically, we first investigate the effects of using enriched stencils and a nonmonotone acceptance rule in our algorithmic framework. Then, we show the effectiveness of our integer lattice exploration strategy when compared to the strategies embedded into two state-of-the-art direct search methods, namely NOMAD (v.3.8.1) [1] and BFO [19]. In order to understand if our simple direct search method is competitive with algorithms that use models, we further include in the analysis the comparison with the model based version of NOMAD.

The paper is organized as follows. In Section 2, we describe the algorithmic framework for black box problems with bound constraints. Then, in Section 3, we explain how to handle problems with black box general constraints. We hence report a numerical experience both for problems with bound constraints and black box constraints, respectively in Section 4 and 5. Finally, we draw some conclusions in Section 6.

1.1 Definitions

We report some definitions that will be useful in the next sections.

Definition 1 (Stencil) Given a point $x \in \mathbb{R}^n$, a stepsize $\alpha \in \mathbb{R}_+$ and p directions $d_i \in \mathbb{R}^n$, $i = 1, \ldots, p$, a stencil is the following set of points:

$$\mathcal{S}(x, \alpha, d_1, \ldots, d_p) = \{x \pm \alpha\, d_i,\ i = 1, \ldots, p\}.$$

A particular stencil is, for instance, the *coordinate* stencil, that is

$$\mathcal{S}(x, \alpha, e_1, \ldots, e_n) = \{x \pm \alpha e_i,\ i = 1, \ldots, n\},$$

where $e_i \in \mathbb{R}^n$ is the i-th coordinate vector. Next, we introduce the definition of *skewed stencil* which is obtained from Def. 1 by allowing different stepsizes for each direction.

Definition 2 (Divisor) For integers a and b, we say that a divides b, or that a is a divisor (or factor) of b, or that b is a multiple of a, if there exists an integer c such that $b = ca$, and we denote this by $a|b$.

Definition 3 (Skewed stencil) Given a point $x \in \mathbb{R}^n$, p stepsizes $\alpha_i \in \mathbb{R}_+$ and p directions $d_i \in \mathbb{R}^n$, $i = 1, \ldots, p$, a skewed stencil is the following set of points:

$$\mathcal{S}(x, \alpha_1, \ldots, \alpha_p, d_1, \ldots, d_p) = \{x \pm \alpha_i \, d_i, \ i = 1, \ldots, p\}.$$

Let $v \in \mathbb{Z}^n$. We call $d \in \mathbb{Z}$ a common divisor of $v_1, \ldots, v_n$ if $d | v_i$, with $i = 1, \ldots, n$. Then, the *greatest common divisor* of $v_1, \ldots, v_n$, denoted as $GCD(v_1, \ldots, v_n)$, is a non-negative common divisor such that all other common divisors of $v_1, \ldots, v_n$ divide d.
Now, we give a few definitions that will be useful when describing in depth our algorithm. We start by introducing the concept of primitive vector:

Definition 4 (Primitive vector) A vector $v \in \mathbb{Z}^n$ is called primitive if $GCD(v_1, \ldots, v_n) = 1$.

Remark 1 Given any two points $x, y \in \mathbb{Z}^n$, we have $x - y = \alpha d$, with $d \in \mathbb{Z}^n$ a primitive vector and $\alpha \in \mathbb{N}$. Hence, starting from a point $x \in \mathbb{Z}^n$ any other point $y \in \mathbb{Z}^n$ can be reached by choosing a suitable stepsize $\alpha \in \mathbb{N}$ along a specific primitive direction $d \in \mathbb{Z}^n$.

Then, we formally define the concept of feasible primitive direction, which represents an important feature in our framework:

Definition 5 (Feasible primitive direction) Given a point $\bar{x} \in \mathcal{C} \cap \mathbb{Z}^n$, a primitive direction d is feasible at $\bar{x}$ for $\mathcal{C}$ when $\beta \in \mathbb{N}$ exists such that

$$\bar{x} + \alpha d \in \mathcal{C} \cap \mathbb{Z}^n, \quad \text{for all } \alpha \leq \beta, \ \alpha \in \mathbb{N}.$$

We further denote with $D(\bar{x})$ the set of all feasible primitive directions at a given point $\bar{x}$. By taking into account Remark 1, it is easy to understand that starting from $\bar{x}$, by suitably picking a direction in the set $D(\bar{x})$ and by properly moving along the chosen direction, we can reach any other feasible point of problem (1).

Definition 6 (Discrete neighborhood) Given a point $\bar{x} \in \mathcal{C} \cap \mathbb{Z}^n$ and a parameter $\beta \in \mathbb{N}$, the discrete neighborhood of $\bar{x}$ is

$$\mathcal{N}(\bar{x}, \beta) = \{x \in \mathcal{C} \cap \mathbb{Z}^n : x = \bar{x} + \alpha d, \text{ with } \alpha \leq \beta, \ \alpha \in \mathbb{N} \text{ and } d \in D(\bar{x})\}.$$

Remark 2 Note that, the discrete neighborhood $\mathcal{N}(\bar{x}, \beta)$ can coincide with the whole feasible set of Problem (1), provided that the parameter β is chosen sufficiently large.

An example of discrete neighborhood is given in Figure 1. Obviously, the concept of discrete neighborhood is only ideal. Indeed, building up a such a neighborhood is an expensive task that cannot be efficiently done in practice. This is the reason why we need to replace $D(\bar{x})$ in the definition above with a suitably chosen subset of directions, thus getting the following definition.

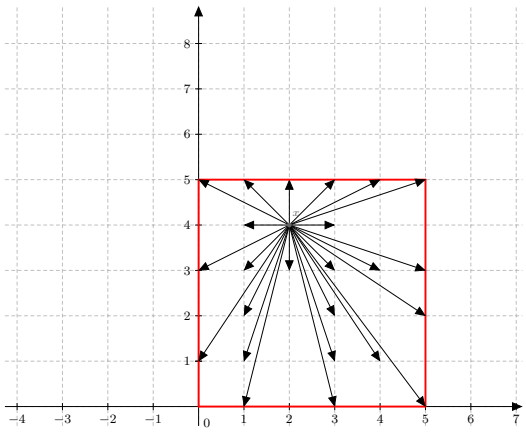

Fig. 1 Discrete neighborhood: example with $\beta = 1$, $X = \{x \in \mathbb{R}^2 : \ 0 \leq x_1 \leq 5, \ 0 \leq x_2 \leq 5\}$

Definition 7 (Weak discrete neighborhood) Given a point $\bar{x} \in \mathcal{C} \cap \mathbb{Z}^n$, a parameter $\beta \in \mathbb{N}$ and a subset $D \subset D(\bar{x})$, the weak discrete neighborhood of $\bar{x}$ is

$$\mathcal{N}_w(\bar{x}, \beta, D) = \{x \in X \cap \mathbb{Z}^n : x = \bar{x} + \alpha d, \text{ with } \alpha \leq \beta, \ \alpha \in \mathbb{N} \text{ and } d \in D\}.$$

Now, we can formally give the definition of a local minimum for Problem (1).

Definition 8 (Local minimum point) A point $x^* \in \mathcal{C} \cap \mathbb{Z}^n$ is a local minimum of Problem (1), if $\beta \in \mathbb{N}$ exists such that

$$f(x^*) \leq f(x), \quad \forall x \in \mathcal{N}(x^*, \beta). \tag{2}$$

Following the same path as before, we get another definition that makes more sense from a practical point of view.

Definition 9 (Weak local minimum point) A point $x^* \in \mathcal{C} \cap \mathbb{Z}^n$ is a weak local minimum of Problem (1), if a parameter $\beta \in \mathbb{N}$ and a subset $D \subset D(x^*)$ exist such that

$$f(x^*) \leq f(x), \quad \forall x \in \mathcal{N}_w(x^*, \beta, D). \tag{3}$$

2 A nonmonotone algorithm for black box optimization problems with bound constraints

In the first part of the paper, we focus on bound-constrained integer programming problems of the following form:

$$\begin{aligned} &\min f(x) \\ &s.t. \ l_i \leq x_i \leq u_i, \qquad i = 1, \ldots, n \\ &\quad x \in \mathbb{Z}^n \end{aligned} \tag{4}$$

namely, Problem (1) where $\mathcal{C} = \{x \in \mathbb{R}^n : \ l_i \leq x_i \leq u_i, \ i = 1, \ldots, n\}$ with $l_i, u_i \in \mathbb{Z}$ and $-\infty < l_i < u_i < \infty$, for all $i = 1, \ldots, n$. Note that, in this case, $\mathcal{C}$ is obviously a compact subset of $\mathbb{R}^n$. From now on, let us denote the set of bound constraints as follows

$$X = \{x \in \mathbb{R}^n : l_i \leq x_i \leq u_i, \ i = 1, \ldots, n\}.$$

In this section, we describe a new algorithm for black box optimization problems with bound constraints. It basically combines the use of primitive feasible directions with a suitable nonmonotone line search that only explores points in the integer lattice. The detailed scheme of the algorithm is reported below (see Algorithm 1). As we can easily see, there are three different phases at each iteration. In the first phase, we generate points around the iterate x_k. More specifically, we pick a direction from a specific set $D \subseteq D(x_k)$ and explore this direction by means of a nonmonotone line search in order to find a new point that both guarantees a sufficiently large movement along the search direction and a reduction with respect to a specific reference value f^{ref} (see Step 12 in Algorithm 1). Phase 1 will then end once we either find such a point or D becomes an empty set (i.e., all directions in D have been explored, but no reduction is obtained with respect to the reference value). We notice that the iterate x_k, the directions $d \in D_k$ and starting stepsizes $\tilde{\alpha}_k^{(d)}$ might be seen as a dynamically changing skewed stencil suitably modified at each iteration of the algorithm depending on the outcomes of the line search. In case we get a point that guarantees a reduction with respect to the reference value, the algorithm updates

- the starting stepsize related to the last direction explored (i.e., it enlarges the stencil with respect to the direction);
- the reference value;
- the queue needed to build up the reference values in the next iterations.

Once Phase 1 is done, we switch to Phase 2 where the sets of search directions D (set of search directions to be used in Phase 1) and D_k (set of search directions generated so far) are eventually updated. More specifically, if in Phase 1 the algorithm succeded in finding a new point, D_k stays the same and D is set equal to D_k. Otherwise, the algorithm checks if *Nonmonotone search* failed along all the directions in D_k and the starting stepsize is 1 for all of those directions (i.e., the dynamically changing skewed stencil defined by directions $d \in D_k$ and stepsizes $\tilde{\alpha}_k^{(d)}$ shrank to its minimum size in Phase 1). If this is not the case, D_k stays the same and D includes only the directions that failed with stepsize greater than one. Otherwise, the algorithm tryes to enrich D_k. If D_k equals the set $D(x_k)$ of primitive feasible directions in x_k, then the algorithm checks whether x_k is the point with the best objective function. If so, it stops, otherwise it moves to the best point found so far (see Step 19 in Algorithm 1), thus D and D_k are set equal to $D(x_k)$. If the set of generated search directions is still smaller than $D(x_k)$, it is enriched ($D_{k+1} \supset D_k$) and the set D includes only the new directions generated to enrich D_k. Finally, in Phase 3 the iterates are updated.

Algorithm 1 NonMonotone Black Box Optimization Algorithm (NM-BBOA)

Data. $x_0 \in X \cap \mathbb{Z}^n$, $D = D_0 \subset D(x_0)$ a set of initial directions, $\tilde{\alpha}_0^{(d)} = 1$, for each $d \in D_0$. $W = \{f(x_0)\}$, $f^{ref} = \bar{f}_0 = f(x_0)$, $x_{min} = x_0$, $M \geq 1$, $M \in \mathbb{N}$.

For $k = 0, 1, \ldots$

PHASE 1 - Explore points around x_k

Set $y = x_k$

While $D \neq \emptyset$ and $y = x_k$ **do**

Pick and explore direction

Choose $d \in D$ set $D = D \setminus \{d\}$

Compute α by the *Nonmonotone Search*$(\tilde{\alpha}_k^{(d)}, x_k, d, f^{ref}; \alpha)$

Update starting stepsize, reference value and queue

If $\alpha = 0$ **then** set $\tilde{\alpha}_{k+1}^{(d)} = \max\{1, \lfloor \tilde{\alpha}_k^{(d)}/2 \rfloor\}$

else

Set $y = x_k + \alpha d$, $\alpha_k^{(d)} = \alpha$ and $\tilde{\alpha}_{k+1}^{(d)} = \alpha$

If $f(y) < f(x_{min})$ **then** $x_{min} = y$

If $|W| = M$ **then** pop(W)

push$(f(y), W)$, $f^{ref} = \max_{f \in W}\{f\}$

End If

End While

PHASE 2 - Update set of search directions

If $y = x_k$ **then**

If *Nonmonotone Search* failed with $\tilde{\alpha}_k^{(d)} = 1$ for all $d \in D_k$ **then**

If $D_k = D(x_k)$ **then**

If $f(x_k) = f(x_{min})$ **then** STOP

else Set $y = x_{min}$ and $D = D_{k+1} = D(x_k)$

else

Generate $D_{k+1} \supset D_k$, set $\tilde{\alpha}_{k+1}^{(d)} = 1$, for all $d \in D_{k+1}$ and $D = D_{k+1} \setminus D_k$

End If

else

Set $D_{k+1} = D_k$ and $D = \{d \in D_k : \ \textit{Nonmonotone Search}$ failed with $\tilde{\alpha}_k^{(d)} > 1\}$

End If

else

Set $D_{k+1} = D_k$ and $D = D_k$

End If

PHASE 3 - Update iterates

Set $x_{k+1} = y$, $\bar{f}_{k+1} = f^{ref}$

End For

The detailed scheme of the Nonmonotone Search is reported in Algorithm 2. It first calculates the starting stepsize for the search. This is chosen as the minimum between $\tilde{\alpha}_k^{(d)}$ (defining the dynamically changing skewed stencil) and the largest stepsize $\bar{\alpha}$ that can be taken along the search direction d (See Initialization). If the starting stepsize is greater than zero and the function reduces along the search direction with respect to the reference vaule f^{ref}, the search starts expanding the stepsize and keeps doing it until either the maximum stepsize is reached or a reduction with respect to the reference

value cannot be guaranteed anymore (See Step 3). We would like to notice that the line search moves along the direction by guaranteeing feasibility (the points chosen are on the integer lattice).

Algorithm 2 Nonmonotone Search

Input. $\tilde{\alpha}_k^{(d)}$, x_k, d, f^{ref}

Initialization. Compute the largest $\bar{\alpha}$ such that $x_k + \bar{\alpha} d \in X \cap \mathbb{Z}^n$. Set $\alpha = \min\{\bar{\alpha}, \tilde{\alpha}_k^{(d)}\}$.
Step 1. **If** $\alpha > 0$ and $f(x_k + \alpha d) < f^{ref}$ **then** go to Step 2
else Set $\alpha = 0$ and go to Step 5
Step 2. Let $\beta = \min\{\bar{\alpha}, 2\alpha\}$
Step 3. **If** $\alpha = \bar{\alpha}$ or $f(x_k + \beta d) \geq f^{ref}$ **then** set $\tilde{\alpha}_{k+1}^{(d)} = \alpha$ and go to Step 5
Step 4. Set $\alpha = \beta$ and go to Step 2
Step 5. Return α

In the following Theorem, we prove convergence of the method to a local minimum. We would like to notice that the minimum obtained is related to a discrete neighborhood with $\beta = 1$.

Theorem 1 *Let $\{x_k\}$ and $\{\bar{f}_k\}$ be the sequences of solutions and of reference values, respectively, generated by NM-BBOA. Then, the algorithm cannot cycle and produces a local minimum point.*

Proof. First, we observe that the sequence $\{\bar{f}_k\}$ is bounded from below, since the number of solutions in the feasible set is finite. Then, let us define

$$K = \{k : x_{k+1} \neq x_k\},$$

and

$$H(k, \bar{k}) = \{h \in K : k < h \leq \bar{k}\}, \text{ for } \bar{k} \geq k.$$

Then, let $\bar{k}(M)$ be the index such that

$$|H(k, \bar{k}(M))| = M.$$

For each $k \in K$, we have that

$$f(x^{k+1}) < \bar{f}_k. \tag{5}$$

Moreover, from the updating rules of f^{ref} and the definition of $\bar{f}_k$, we have that

$$\bar{f}_{k+1} \leq \bar{f}_k. \tag{6}$$

Then, remembering that $|X \cap \mathbb{Z}^n| < \infty$, we can define

$$0 < \delta = \min_{x,y \in X \cap \mathbb{Z}^n} \Big\{ |f(x) - f(y)| : f(x) \neq f(y) \Big\}.$$

so that we have

$$\bar{f}_{\bar{k}(M)} < \bar{f}_k - \delta. \tag{7}$$

We prove now that NM-BBOA does not cycle. By contradiction, we have that $\{x_k\}$ is an infinite sequence. If this is the case, it is easy to see that the set K is also infinite. Since the procedure does not terminate, a solution $\tilde{x}$ (which is not a local minimum) is generated an infinite number of times. By (6) and (7), there exists an iteration $\tilde{k}$ such that

$$\bar{f}_{\tilde{k}} \leq f(\tilde{x}).$$

Furthermore, as $\tilde{x}$ is generated an infinite number of times, there exists an iteration $\hat{k} \geq \tilde{k}$ such that

$$f(\tilde{x}) < \bar{f}_{\hat{k}}.$$

Hence, we have

$$f(\tilde{x}) < \bar{f}_{\hat{k}} \leq \bar{f}_{\tilde{k}} \leq f(\tilde{x}),$$

which shows that the local search procedure cannot cycle.
Finally, we prove that the point produced x^* is a local minimum of the problem. When NM-BBOA stops, let $\bar{k}$ be the last iteration index, so that $x^* = x_{\bar{k}}$ and is such that $x_{\bar{k}} = x_{min}$. Furthermore, $D_{\bar{k}} = D(x^*)$ is the set of all the feasible and relatively prime directions at x^*. We thus have

$$f(x^*) \leq \bar{f}_{\bar{k}} \tag{8}$$

and, by the instructions of the algorithm,

$$\bar{f}_{\bar{k}} \leq f(x^* + d) \quad \forall\, d \in D(x^*). \tag{9}$$

Then, combining inequalities (8) and (9), we get that x^* is a local minimum. □

Obviously, it is possible to develop a procedure that explores a larger discrete neighborhood (i.e., a neighborhood with $\beta > 1$). In order to do that, we just need to suitably modify the way we set the starting stepsize when generating a new direction ($\tilde{\alpha}^{(d)}_{k+1} = \beta$ at Step 21) and we need to set to β the stepsizes that are equal to one when we get a successful iteration (if $\tilde{\alpha}^{(d)}_{k} = 1$ then set $\tilde{\alpha}^{(d)}_{k+1} = \beta$ at Step 27). Taking into account Remark 2, those points we obtain in the end might actually be global minima when β is sufficiently large. This parameter should anyway be wisely chosen in order to keep the exploration computationally cheap. It is easy to understand that such a choice becomes even more critical when getting a small budget of evaluations.

3 Handling of black box general constraints

We now consider problem

$$\begin{aligned} &\min\ f(x) \\ &s.t.\ g(x) \leq 0 \\ &\quad x \in X \cap \mathbb{Z}^n, \end{aligned} \tag{10}$$

where $g : \mathbb{R}^n \to \mathbb{R}^m$ are $m \geq 1$ black box constraint functions. The above problem is Problem (1) where $\mathcal{C} = \{x \in \mathbb{R}^n : \ g(x) \leq 0, \ l \leq x \leq u\} = X \cap \mathcal{F}$ where $\mathcal{F} = \{x \in \mathbb{R}^n : \ g(x) \leq 0\}$. In order to handle the nonlinear constraints, we use a simple penalty approach (see, e.g., [13]). Specifically, given Problem (10) and a penalty parameter $\epsilon \in \mathbb{R}_+$, we introduce the following penalty function:

$$P(x; \epsilon) = f(x) + \frac{1}{\epsilon} s(x),$$

where $s(x) = \sum_{i=1}^{m} \max\{0, g_i(x)\}$ and $\epsilon > 0$. Then, we consider the following bound-constrained problem

$$\begin{aligned} &\min \ P(x; \epsilon) \\ &s.t. \ x \in X \cap \mathbb{Z}^n \end{aligned} \tag{11}$$

Now, we prove equivalence between the original problem (10) and the penalized problem (11). In particular, we will prove that there exists a threshold value $\bar{\epsilon}$ for the penalty parameter such that, for any $\epsilon \in (0, \bar{\epsilon})$, any minimum of the penalized problem is also a minimum of the original problem and viceversa. To ease the notation, we again indicate the set of box constraints with X.

Theorem 2 *Given problem* (10) *and considering problem* (11)*, a threshold value $\bar{\epsilon} > 0$ exists such that for every $\epsilon \in (0, \bar{\epsilon})$, any global minimum point $\bar{x}$ of* (11) *is also a global minimum of* (10) *and viceversa.*

Proof. We first prove that any minimum point $\bar{x}$ of (11) is also a minimum of (10). Let us define the following penalty parameter

$$\bar{\epsilon} = \frac{a}{b}, \tag{12}$$

with

$$a = \min\{s(x), x \in X, g(x) > 0\} \tag{13}$$

and

$$b = \max\{f(y) - f(x), x, y \in X, g(x) > 0, g(y) \leq 0\}.$$

By contradiction, we assume that there exists a minimum point $\bar{x}$ of Problem (11) that is not feasible for Problem (10). For any feasible point y of Problem (10), we have:

$$f(y) - f(\bar{x}) \leq b = \frac{1}{\bar{\epsilon}} a < \frac{1}{\epsilon} s(\bar{x}),$$

with $\epsilon \in (0, \bar{\epsilon})$. Hence, we can write

$$f(y) < f(\bar{x}) + \frac{1}{\epsilon} s(\bar{x}),$$

thus contradicting the fact that $\bar{x}$ is minimum for Problem (11).

We now prove that any minimum point $\bar{x}$ of (10) is a minimum of (11). For any point $x \in X$ not feasible for Problem (10) we have

$$f(\bar{x}) - f(x) \leq b = \frac{1}{\bar{\epsilon}} a < \frac{1}{\epsilon} s(x),$$

with $\epsilon \in (0, \bar{\epsilon})$. Hence, we can write

$$f(\bar{x}) < f(x) + \frac{1}{\epsilon} s(x),$$

and $\bar{x}$ is also minimum for Problem (11). □

In order to prove that every local minimum of the penalized problem is also a local minimum of the original problem, we introduce the following assumption.

Assumption 1 *For every $x \in X \cap \mathbb{Z}^n$ not feasible for the original Problem (10), there exists a direction $\bar{d} \in D(x)$ such that*

$$s(x + \bar{d}) = \sum_{i=1}^{m} \max\{0, g_i(x + \bar{d})\} < \sum_{i=1}^{m} \max\{0, g_i(x)\} = s(x).$$

The assumption, which will also be considered when studying the convergence of the method, is basically a kind of Mangasarian-Fromowitz constraint qualification condition for integer problems. The condition simply says that, when we get a point that is not feasible for the original problem, we can always find a primitive direction that guarantees a reduction of the violation. This sounds pretty reasonable when dealing with the class of problems considered in here. Now, we can prove that there exists a threshold value $\bar{\epsilon}$ for the penalty parameter such that, for any $\epsilon \in (0, \bar{\epsilon})$, any local minimum of the penalized problem is also a local minimum of the original problem.

Theorem 3 *Let Assumption 1 hold. Given problem* (10) *and considering problem* (11)*, a threshold value $\bar{\epsilon} > 0$ exists such that for every $\epsilon \in (0, \bar{\epsilon})$, any local minimum point $\bar{x}$ of* (11) *is also a local minimum of* (10)*.*

Proof. Let us define the following penalty parameter

$$\bar{\epsilon} = \frac{a}{b}, \tag{14}$$

with

$$a = \min\{s(x) - s(x + d), x \in X, d \in D(x), s(x) - s(x + d) > 0\}$$

and

$$b = \max\{f(x + d) - f(x), x \in X, d \in D(x), f(x + d) - f(x) > 0\}.$$

By contradiction, we assume that there exists a local minimum point $\bar{x}$ of Problem (11) that is not feasible for Problem (10). Taking into account Assumption 1, we can find a direction $\bar{d} \in D(\bar{x})$, such that:

$$f(\bar{x} + \bar{d}) - f(\bar{x}) \leq b = \frac{1}{\bar{\epsilon}} a < \frac{1}{\epsilon} \left[s(\bar{x}) - s(\bar{x} + \bar{d}) \right],$$

with $\epsilon \in (0, \bar{\epsilon})$. Hence, we can write

$$f(\bar{x} + \bar{d}) + \frac{1}{\epsilon} s(\bar{x} + \bar{d}) < f(\bar{x}) + \frac{1}{\epsilon} s(\bar{x}),$$

thus contradicting the fact that $\bar{x}$ is local minimum for Problem (11). □

The algorithm we use in the constrained case, called NM-BBOA_CP, has the same structure as the NM-BBOA algorithm for bound-constrained integer programs. The detailed scheme is reported in Algorithm 3. It is easy to see that the main differences between the two are:

1. the function f is replaced by the penalty function P;
2. a specific rule is used for the update of the penalty parameter ϵ.

As we can easily notice, the algorithm checks, in Phase 2, if the update is timely. More specifically, if *Nonmonotone search* failed along all the directions in D_k and the starting stepsize is 1 for all of those directions (i.e., the dynamically changing skewed stencil defined by directions $d \in D_k$ and stepsizes $\tilde{\alpha}_k^{(d)}$ shrank to its minimum size in Phase 1), the algorithm checks if either violation of the constraints is larger than a reference value μ_k (which goes to zero as k goes to infinity) or all of the directions in $D(x_k)$ have been generated. If this is the case, the penalty parameter decreases, otherwise it stays the same.
We finally prove finite convergence of the method to a local minimum. Again it is appropriate to notice that the local minimum obtained is related to a discrete neighborhood with $\beta = 1$. We further point out that, in order to prove the result, Assumption 1 is used.

Theorem 4 *Let Assumption 1 hold. Let $\{x_k\}$ and $\{\bar{P}_k\}$ be the sequences of solutions and of reference values, respectively, generated by NM-BBOA_CP. Then, the algorithm terminates after a finite number of iterations $\bar{k}$ and the produced point $x_{\bar{k}}$ is a local minimum of the original Problem (10).*

Proof. We assume, by contradiction, that the sequence $\{x_k\}$ is infinite. We have two different cases:

Case 1. $\epsilon_k = \tilde{\epsilon}$ when k sufficiently large. In this case, taking into account (13), we have that $\mu_k < a$ for k sufficiently large. Thus points generated are all feasible. Hence, the proof is a verbatim repetition of proof given for Theorem 1.

Case 2. $\epsilon_k \to 0$. We first define

$$K = \{k : \epsilon_{k+1} \neq \epsilon_k\}.$$

Taking into account the fact that X is compact, we can now consider a further subsequence $K_1 \subset K$ such that $x_k = \tilde{x}$ for all $k \in K_1$. When k is sufficiently large, we have $D_k = D(\tilde{x})$, and, from the instructions of the algorithm, we can write

$$P(\tilde{x} + d; \epsilon_k) \geq \bar{P}_k \geq P(\tilde{x}; \epsilon_k), \quad \forall\, d \in D(\tilde{x}). \tag{15}$$

Algorithm 3 NonMonotone Black Box Optimization Algorithm for Constrained Problems (NM-BBOA_CP)

Data. $x_0 \in X \cap \mathbb{Z}^n$, $D = D_0 \subset D(x_0)$ a set of initial directions, $\tilde{\alpha}_0^{(d)} = 1$, for each $d \in D_0$. $\epsilon_0 > 0$, $\theta \in (0,1)$ and a sequence $\{\mu_k\} \downarrow 0$. $W = \{P(x_0;\epsilon_0)\}$, $P^{ref} = \bar{P}_0 = P(x_0;\epsilon_0)$, $x_{min} = x_0$, $M \geq 1$, $M \in \mathbb{N}$.

For $k = 0, 1, \ldots$

PHASE 1 - Explore points around x_k

Set $y = x_k$

While $D \neq \emptyset$ and $y = x_k$ **do**

Pick and explore direction

Choose $d \in D$ set $D = D \setminus \{d\}$

Compute α by the *Nonmonotone Search*$(\tilde{\alpha}_k^{(d)}, x_k, d, P^{ref}; \alpha)$

Update starting stepsize, reference value and queue

If $\alpha = 0$ **then** set $\tilde{\alpha}_{k+1}^{(d)} = \max\{1, \lfloor \tilde{\alpha}_k^{(d)}/2 \rfloor\}$

else

Set $y = x_k + \alpha d$, $\alpha_k^{(d)} = \alpha$ and $\tilde{\alpha}_{k+1}^{(d)} = \alpha$

If $P(y;\epsilon_k) < P(x_{min};\epsilon_k)$ **then** $x_{min} = y$

If $|W| = M$ **then** pop(W)

push$(P(y;\epsilon_k), W)$, $P^{ref} = \max_{P \in W}\{P\}$

End If

End While

PHASE 2 - Update set of search directions and penalty parameter

If $y = x_k$ **then**

If *Nonmonotone Search* failed with $\tilde{\alpha}_k^{(d)} = 1$ for all $d \in D_k$ **then**

Set `upd=FALSE`

If $D_k = D(x_k)$ **then**

If $(\|g^+(x_k)\| = 0)$ **then**

If $f(x_k) = f(x_{min})$ **then** STOP

else Set $y = x_{min}$ and $D = D_{k+1} = D(x_k)$

else

Set `upd=TRUE` and $D = D_{k+1} = D(x_k)$

End If

else

Generate $D_{k+1} \supset D_k$, set $\tilde{\alpha}_{k+1}^{(d)} = 1$, for all $d \in D_{k+1}$ and $D = D_{k+1} \setminus D_k$

End If

If $(\|g^+(x_k)\| > \mu_k)$ or (`upd`) **then** Set $\epsilon_{k+1} = \theta\epsilon_k$

else Set $\epsilon_{k+1} = \epsilon_k$

else

Set $\epsilon_{k+1} = \epsilon_k$

Set $D_{k+1} = D_k$ and $D = \{d \in D_k : \ \textit{Nonmonotone Search}$ failed with $\tilde{\alpha}_k^{(d)} > 1\}$

End If

else

Set $\epsilon_{k+1} = \epsilon_k$

Set $D_{k+1} = D_k$ and $D = D_k$

End If

PHASE 3 - Update iterates

Set $x_{k+1} = y$, $\bar{P}_{k+1} = P^{ref}$

End For

Hence, dividing by ϵ_k and considering the limit for $k \to \infty$, we have

$$\sum_{i=1}^{m} \max\{0, g_i(\tilde{x}+d)\} \geq \sum_{i=1}^{m} \max\{0, g_i(\tilde{x})\}, \quad \forall\, d \in D(\tilde{x}).$$

Thus we get, by recalling Assumption 1, that $\tilde{x}$ is feasible and, thanks to inequality (15), it is a local minimum for Problem (10). This contradicts the fact that the algorithm does not terminate.

Then, the theorem is proved. □

4 Numerical experience on bound constrained problems

In this section we report the results of the numerical experience and comparison of our nonmonotone algorithm (NM-BBOA) with other solvers from the literature, on problems with only bound constraints on the variables. We also report comparison between algorithms obtained from NM-BBOA by disabling phase 2 (thus obtaining an algorithm using dynamically changing skewed stencils) or disabling both the nonmonotone search procedure and phase 2 (thus coming up with an algorithm using fixed stencil). Furthermore, we also investigate to what extent the nonmonotonicity is useful in this context.

In order to evaluate the relative performances of the algorithms on bound constrained problems, we use the set of 48 unconstrained nonsmooth problems from [20]. More precisely, we use the problems from sections 2 and 3 of [20], i.e. unconstrained minimax problems and general nonsmooth unconstrained problems. Then, since the selected problems are indeed unconstrained, we add bound constraints on the variables as

$$\ell_i = (\tilde{x}_0)_i - 10 \leq \tilde{x}_i \leq (\tilde{x}_0)_i + 10 = u_i, \quad i = 1, \ldots, n,$$

where $\tilde{x}_0$ is the provided starting point for the problem. Furthermore, given the *continuous* bound constrained optimization problem

$$\begin{array}{l} \min\ \tilde{f}(\tilde{x}) \\ s.t.\ \ell_i \leq \tilde{x}_i \leq u_i, \qquad i = 1, \ldots, n \\ \quad \tilde{x} \in \mathbb{R}^n, \end{array}$$

we consider the discretized problem

$$\begin{array}{l} \min\ f(x) \\ s.t.\ 0 \leq x_i \leq 100, \qquad i = 1, \ldots, n \\ \quad x \in \mathbb{Z}^n \end{array} \tag{16}$$

where $f(x) = \tilde{f}(y)$ with

$$y_i = \ell_i + x_i(u_i - \ell_i)/100, \quad i = 1, \ldots, n.$$

As concerns the starting point x_0 for Problem (16), we set

$$(x_0)_i = 50, \qquad i = 1, \ldots, n,$$

and note that x_0 is nothing but the point

$$(x_0)_i = \lfloor 100((\tilde{x}_0)_i - \ell)/(u - \ell) \rceil, \quad i = 1, \ldots, n,$$

where $\lfloor \cdot \rceil$ denotes the nearest integer operator.

4.1 The algorithms

In this section we briefly describe the algorithms that we used in the comparison. Apart from our proposed solver NM-BBOA (with $D_0 = \{e_1, \ldots, e_n\}$ and $M = 4$), we consider:

- M-BBOA, the monotone version of NM-BBOA, i.e. the one obtained by setting $M = 1$ in algorithm NM-BBOA;
- NM-DCS (M-DCS), nonmonotone (respectively, monotone) dynamically changing (skewed) stencil algorithm, i.e. the nonmonotone (monotone) version of the algorithm obtained from NM-BBOA (M-BBOA) where phase 2 is disabled so that $D_{k+1} = D_k = D$ for all k;
- NM-FS (M-FS), nonmonotone (respectively, monotone) fixed stencil algorithm, i.e. the algorithm obtained from NM-DCS (M-DCS) where we inhibit the step expansion within the nonmonotone search procedure;
- NOMAD (v.3.8.1) [1];
- BFO [19].

All the algorithms have been run specifying a maximum of 5000 function evaluations. As concerns step 21 of algorithms NM-BBOA and M-BBOA, namely the procedures that given set D_k returns $D_{k+1} \supset D_k$, it is implemented according to [2]. Specifically, procedure described in Algorithm 4 below is used. NOMAD has been run by using the default parameter settings except for

`initial_mesh_size = 5;  direction_type = ortho 2n.`

In order to better understand the effectiveness of our exploration strategy when compared with the pure MADS strategy used by NOMAD for integer problems, we further run NOMAD with the option

`disable = models`,

and report in the comparisons both versions of NOMAD.

4.2 Results

First, we investigated the efficiency and robustness of our codes NM-BBOA, M-BBOA, NM-DCS, M-DCS, NM-FS, and M-FS. Such a comparison, in terms of data and performance profiles [15], is reported in Figure 2 for values of the

Algorithm 4 Procedure to generate new set of search directions

Input. $t > 0$, $\eta > 0$, D_k
If $\eta < 50\sqrt{n}/2$
  For $h = 1, \ldots, 1000$
    Set $t \leftarrow t + 1$
    Let u_t be the t-th vector in the n dimensional Halton sequence [11]
    Compute

$$q_t(\eta) = \left\lfloor \eta \frac{2u_t - e}{\|2u_t - e\|} \right\rceil \in \mathbb{Z}^n \cap \left[-\eta - \frac{1}{2}, \eta + \frac{1}{2} \right]^n .$$

    If $q_t(\eta)$ is a prime vector and $q_t(\eta) \notin D_k$
      Set $D_{k+1} = D_k \cup \{q_t(\eta)\}$
      return (`success`, t, η, D_{k+1})
    Endif
  End For
Endif
return (`failure`, t, η, D_k)

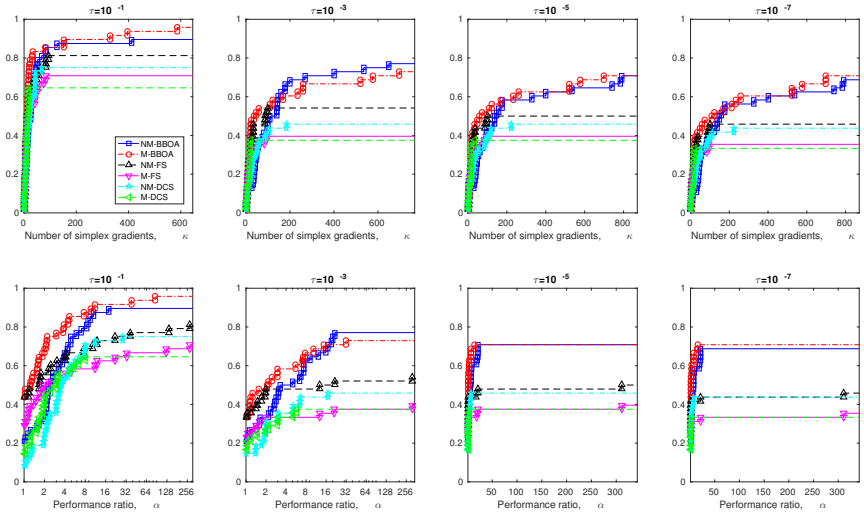

Fig. 2 Comparison between BBOA, DCS and FS, both monotone and nonmonotone, on the 48 bound constrained problems

gate parameter τ in $\{10^{-1}, 10^{-3}, 10^{-5}, 10^{-7}\}$. It can be noted that both NM-BBOA and M-BBOA are quite efficient and robust over the other methods. In Figure 3, we hence compare NM-BBOA and M-BBOA against NOMAD (with and without models) and BFO. From Figures 2 and 3, we can conclude that M-BBOA is slightly better than NM-BBOA in terms of efficiency. This can be explained by considering that the nonmonotone algorithm can take uphill steps especially at the beginning of the optimization process. This could in turn

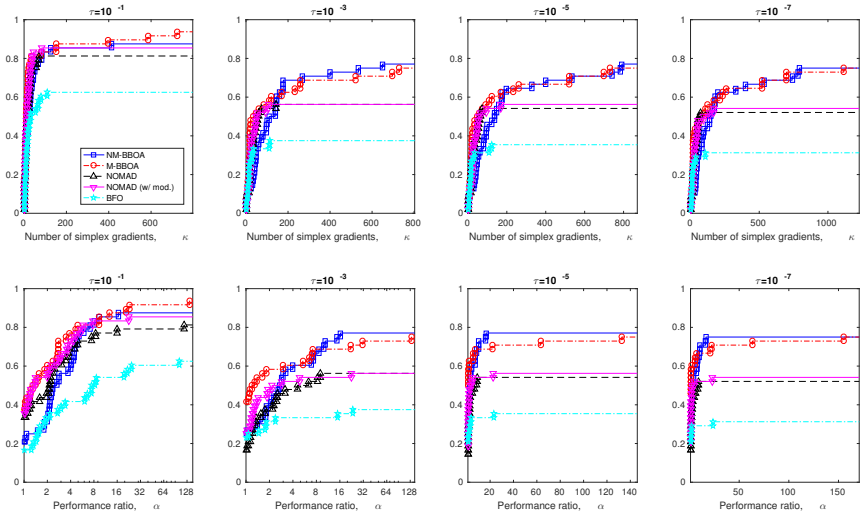

Fig. 3 Comparison between NM-BBOA, M-BBOA, NOMAD (3.8.1) with/without models, and BFO on the 48 bound constrained problems

justify a less steeper descent of the nonmonotone algorithm with respect to the monotone one. However, the nonmonotone algorithm should have a greater ability to escape from local minima, thus converging to better minimum points. Unfortunately, this does not seem to emerge from the figures, that are instead telling us that NM-BBOA and M-BBOA are almost equivalent in terms of robustness. We suspect that the considered problems are too easy for the advantages of the nonmonotonicity to clearly emerge. We better investigate this aspect in the next subsection where noise is considered the problems more difficult.

4.3 Deterministic additive noise

In order to better investigate the usefulness of the nonmonotonicity in our codes, we decided to run some tests on noisy problems. To this aim, given Problem (16) and according to [15], we define a noisy function

$$f_N(x) = (1 + \epsilon\phi(x))f(x),$$

with the relative noise level $\epsilon = 10^{-3}$ and the noise function ϕ defined as

$$\phi(x) = \phi_0(x)(4\phi_0(x)^2 - 3)$$

where

$$\phi_0(x) = 0.9\sin(100\|x\|_1)\cos(100\|x\|\infty) + 0.1\cos(\|x\|_2).$$

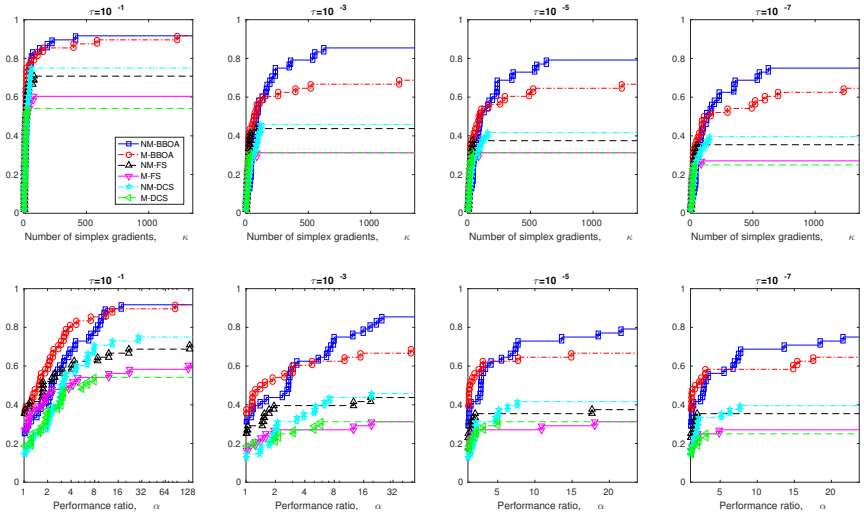

Fig. 4 Comparison between BBOA, DCS and FS, both monotone and nonmonotone, on the 48 bound constrained noisy problems

Comparisons of our codes (NM-BBOA, M-BBOA, NM-DCS, M-DCS, NM-FS, and M-FS) on the 48 bound constrained noisy problems is reported in Figure 4. Figure 5 reports the comparison between NM-BBOA, M-BBOA, NOMAD and BFO on the noisy problems. These confirm the conclusions drawn when noise is not present. On top of this, now it clearly emerges the superiority of the nonmonotone algorithm NM-BBOA over the monotone version M-BBOA in terms of robustness, which is precisely what we would expect.

4.4 Results on a class of hard global optimization problems

In this section we study the impact of a parameter $\beta \geq 1$ on the ability of the proposed algorithm to find the global minimum of Problem (4). To this aim, we define a class of hard bound constrained problems. More specifically, we consider problem

$$\begin{aligned} &\min \ \varphi(x) \\ &s.t. \ 0 \leq x_i \leq 100, \ i = 1, 2 \\ &\quad x \in \mathbb{Z}^2 \end{aligned} \tag{17}$$

where

$$\varphi(x) = \min_{j=1,\ldots,20} \ln\left(\|c_j - x\| + \sigma_j\right)$$

with c_j, $j \in \{1, \ldots, 20\}$, random feasible points for problem (17), $\sigma_j = 10^{-2}$, $j \in \{1, \ldots, 20\} \setminus \bar{J}$, and $\sigma_j = 10^{-6}$, $j \in \bar{J}$, where $\bar{J}$ is a random subset of

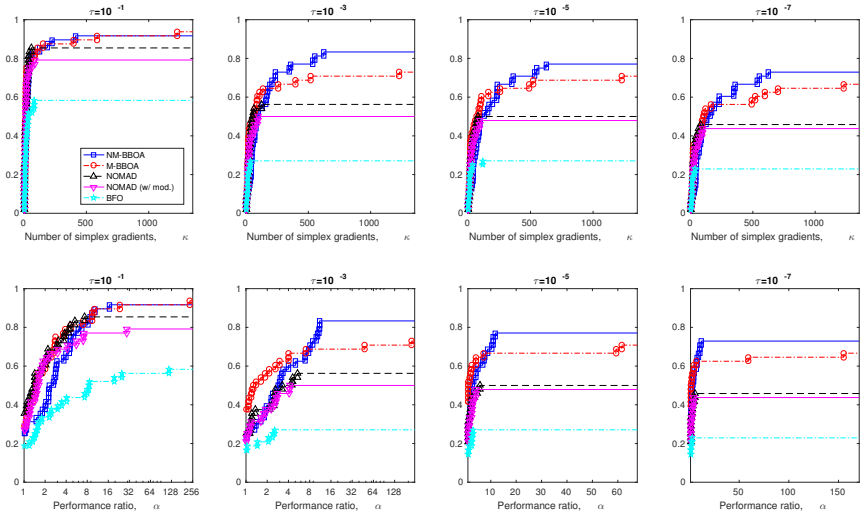

Fig. 5 Comparison between NM-BBOA, M-BBOA, NOMAD (3.8.1) with/without models, and BFO on the 48 bound constrained noisy problems

$\{1, \ldots, 20\}$ with $|\bar{J}| = 3$. Note that, by definition of $\varphi(x)$, it results

$$\varphi(c_j) = \begin{cases} \ln(10^{-6}) & j \in \bar{J}, \\ \ln(10^{-2}) & j \notin \bar{J}. \end{cases}$$

We compare performance of NM-BBOA with a modified version of NM-BBOA that tries to explore larger neighborhoods. Taking into account Remark 2 and comments at the end of Section 2, we used $\beta = 50$ in the modified version of the algorithm (note that NM-BBOA uses a value $\beta = 1$). Both algorithms have been run on a set of 100 randomly generated problems of the form (17), allowing a maximum of 5000 function evaluations. In Figure 6, we plot the percentage of problems solved to global optimality (y axis) with the given number of function evaluations (x axis). As we can easily see, the modified version of NM-BBOA gets a larger percentage of global minima with respect to the standard version, once the number of evaluations is large enough (i.e., larger than 1000). These results seem to indicate that better performances can evenutally be obtained when exploring larger neighborhoods in NM-BBOA (that is, when suitably setting the β parameter in the algorithm).

5 Numerical experience on general constrained problems

This section is devoted to the analysis of the results obtained by the proposed algorithms on general constrained test problems. Furthermore, comparison with NOMAD (version 3.8.1) [1] is reported.

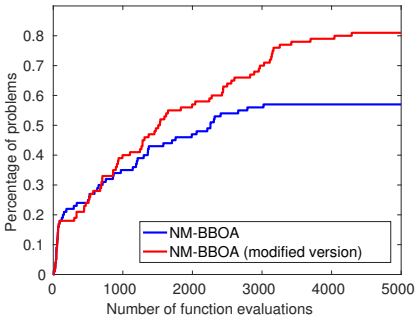

Fig. 6 Comparison between NM-BBOA and modified NM-BBOA on hard global optimization problems

We defined a collection of constrained problems by adding to every bound constrained problem (among the 48 problems from [20] previously defined) the following families of nonlinear constraints.

$$
\begin{array}{ll}
\tilde{g}_j(\tilde{x}) = (3 - 2x_{j+1})x_{j+1} - x_j - 2x_{j+2} + 1 \leq 0, & j = 1, \ldots, n-2 \ (n \geq 3); \\
\tilde{g}_j(\tilde{x}) = (3 - 2x_{j+1})x_{j+1} - x_j - 2x_{j+2} + 2.5 \leq 0, & j = 1, \ldots, n-2 \ (n \geq 3); \\
\tilde{g}_j(\tilde{x}) = x_j^2 + x_{j+1}^2 + x_j x_{j+1} - 2x_j - 2x_{j+1} + 1 \leq 0, & j = 1, \ldots, n-1 \ (n \geq 2); \\
\tilde{g}_j(\tilde{x}) = x_j^2 + x_{j+1}^2 + x_j x_{j+1} - 1 \leq 0, & j = 1, \ldots, n-1, (n \geq 2); \\
\tilde{g}_j(\tilde{x}) = (3 - 0.5x_{j+1})x_{j+1} - x_j - 2x_{j+2} + 1 \leq 0, & j = 1, \ldots, n-2 \ (n \geq 3); \\
\tilde{g}_1(\tilde{x}) = \sum_{j=1}^{n-2}((3 - 0.5x_{j+1})x_{j+1} - x_j - 2x_{j+2} + 1) \leq 0, & (n > 3).
\end{array}
$$

In this way we obtain a set of 237 test problems with $n \in [2, 50]$ and $m \in [1, 49]$. Note that, as already discussed, given the *continuous* constraint function $\tilde{g}_j(\tilde{x})$, $j = 1, \ldots, m$, we consider the discretized constraint function $g_j(x)$ such that $g_j(x) = \tilde{g}_j(y)$ with $y_i = \ell_i + x_i(u_i - \ell_i)/100$, $i = 1, \ldots, n$.

5.1 Results

In Figure 7, we report comparison between algorithm NM-BBOA_CP and NOMAD (3.8.1) where we inhibit the use of models through the option `disable = models`. As we can see, NOMAD is slightly more efficient than NM-BBOA_CP but considerably less robust. As a further experiment, we compare our algorithm NM-BBOA_CP against NOMAD where use of models is allowed. The results are reported in Figure 8. As we expected, the use of models allows NOMAD to improve its performances. Indeed, NOMAD becomes considerably more efficient but it is still less robust (or at least less accurate) than NM-BBOA_CP. To better investigate the situation, we repeat the same comparisons but on a restricted test set obtained by selecting problems with $n \geq 10$ among the 237 constrained problems, thus obtaining a subset of 96 problems. Such comparisons are reported in Figures 9 and 10. As it can be noted, NM-BBOA_CP is now both more efficient and more robust than NOMAD not using models. Again, when we allow the use of models within NOMAD, the

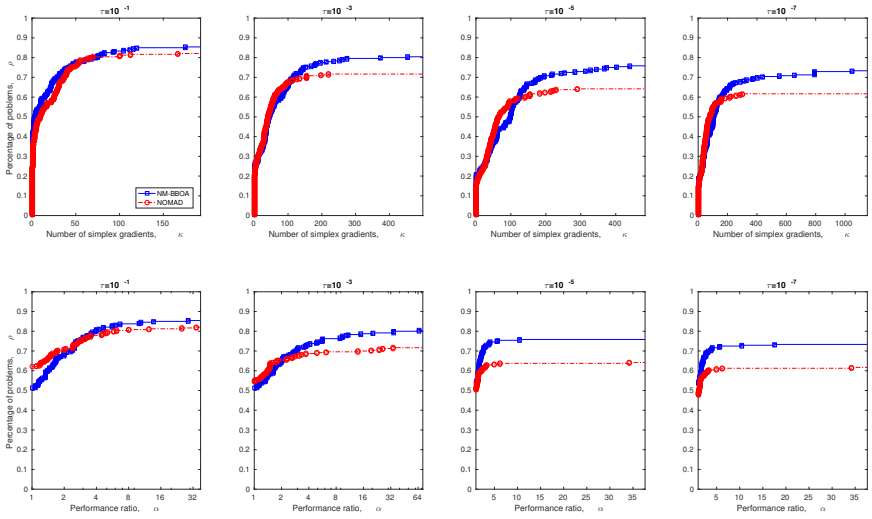

Fig. 7 Comparison between NM-BBOA_CP and NOMAD (3.8.1) without models on the 237 general constrained problems

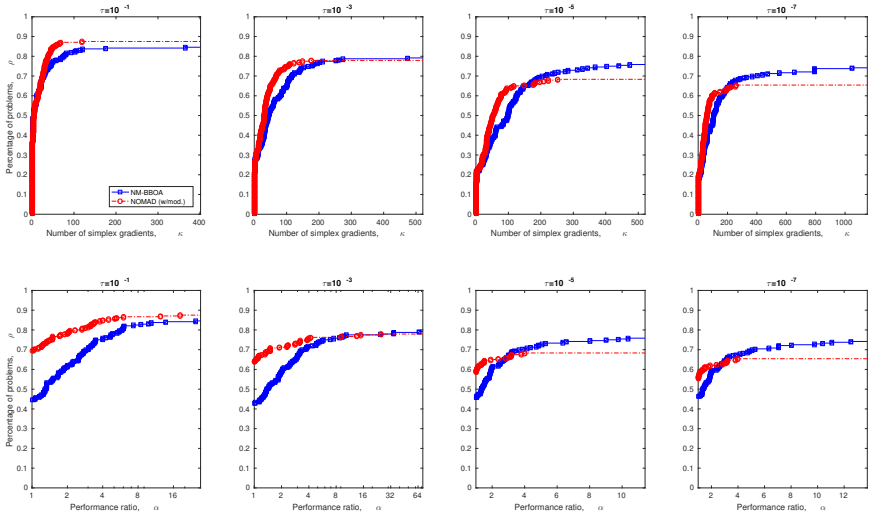

Fig. 8 Comparison between NM-BBOA_CP and NOMAD (3.8.1) using models on the 237 general constrained problems

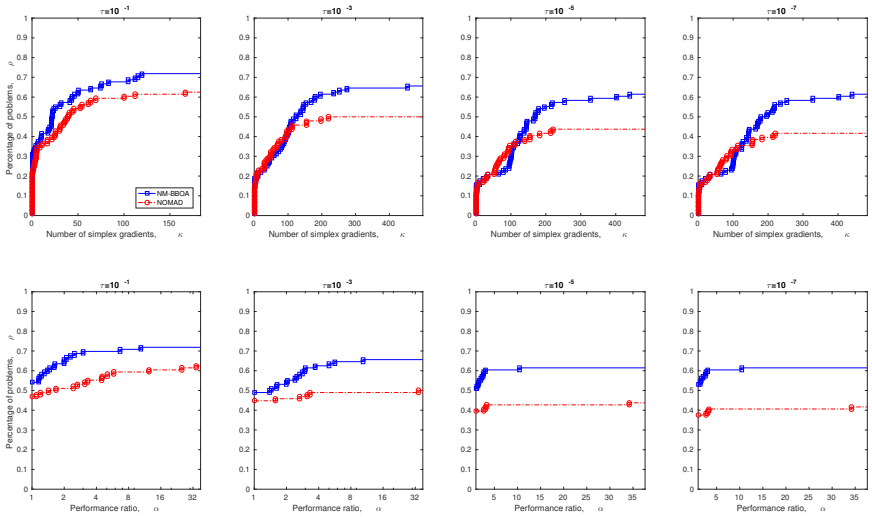

Fig. 9 Comparison between NM-BBOA_CP and NOMAD (3.8.1) without models on the 96 general constrained problems with $n \geq 10$

gap in terms of efficiency is considerably reduced but NM-BBOA_CP is still more robust than NOMAD.

6 Conclusions

In this paper, we developed a tailored strategy for solving black box problems with integer variables. The use of primitive directions combined with a suitably developed nonmonotone line search gives a high level of freedom when exploring the integer lattice and further guarantees a high level of robustness. We first described and analyzed in depth a version of the algorithm that handles bound constrained problems. Then, we tackled the generally constrained case by embedding a penalty approach in the algorithmic framework.
We also included an extensive numerical analysis on a large testbed of both bound constrained and generally constrained problems. As a first step, we both studied the effects of of using enriched stencils and compared monotone vs nonmonotone acceptance rules in our algorithmic framework. The results showed that

- sampling the function by such a dynamically changing set of search directions can significantly improve the performance of the algorithm;
- the use of a nonmonotone line search can get better results in terms of robustness especially when dealing with noisy problems.

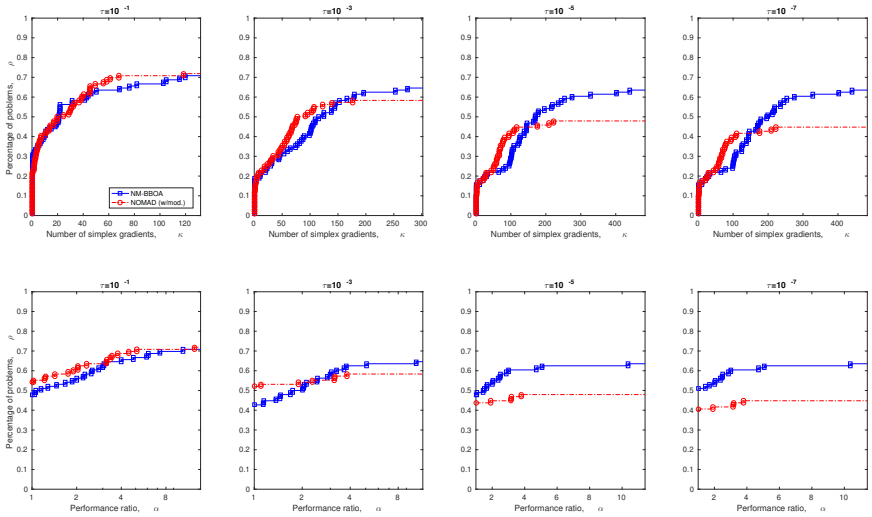

Fig. 10 Comparison between NM-BBOA_CP and NOMAD (3.8.1) using models on the 96 general constrained problems with $n \geq 10$

A comparison with NOMAD (with and without models) and BFO was carried out on bound constrained problems. The results we reported allow us to conclude that our strategy gives better performance both in terms of efficiency and robustness, and the gap significantly increases as we ask for higher precisions or include noise. The results on nonlinearly constrained problems showed that our strategy is again very competitive with the version of NOMAD that does not embed models, while it can only guarantee better performances in terms of robustness against the model based version of NOMAD. The results change if we focus on instances with a number of variables larger than 10. Indeed, models are not really effective in this case and our algorithm outperforms NOMAD both in terms of efficiency and robustness when precision is sufficiently high (and the gap between the two increases as we ask for higher precisions).
Some preliminary results on a class of hard global optimization problems (with bound constraints) further highlighted potential of the algorithm in finding global minima when larger neighborhoods are explored (i.e., when a $\beta > 1$ is properly chosen in the algorithm).

References

1. Abramson, M., Audet, C., Couture, G., Jr., J.D., Digabel, S.L.: The NOMAD project. URL `http://www.gerad.ca/nomad`

2. Abramson, M., Audet, C., Jr., J.D., Le Digabel, S.: Orthomads: A deterministic mads instance with orthogonal directions. SIAM Journal on Optimization **20**(2), 948–966 (2009)
3. Abramson, M.A., Audet, C., Chrissis, J.W., Walston, J.G.: Mesh adaptive direct search algorithms for mixed variable optimization. Optimization Letters **3**(1), 35–47 (2009)
4. Abramson, M.A., Audet, C., Dennis Jr, J.E.: Filter pattern search algorithms for mixed variable constrained optimization problems. Pacific Journal on Optimization **3**(3), 477–500 (2007)
5. Audet, C., Dennis Jr, J.E.: Pattern search algorithms for mixed variable programming. SIAM Journal on Optimization **11**(3), 573–594 (2001)
6. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM computing surveys (CSUR) **35**(3), 268–308 (2003)
7. Conn, A., Scheinberg, K., Vicente, L.N.: Introduction to derivative-free optimization, vol. 8. Siam (2009)
8. Diniz-Ehrhardt, M., Martínez, J., Raydán, M.: A derivative-free nonmonotone line-search technique for unconstrained optimization. Journal of computational and applied mathematics **219**(2), 383–397 (2008)
9. García-Palomares, U.M., Rodríguez, J.F.: New sequential and parallel derivative-free algorithms for unconstrained minimization. SIAM Journal on optimization **13**(1), 79–96 (2002)
10. Grippo, L., Rinaldi, F.: A class of derivative-free nonmonotone optimization algorithms employing coordinate rotations and gradient approximations. Computational Optimization and Applications **60**(1), 1–33 (2015)
11. Halton, J.: On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. Numerische Mathematik **2**, 84–90 (1960)
12. Liuzzi, G., Lucidi, S., Rinaldi, F.: Derivative-free methods for bound constrained mixed-integer optimization. Computational Optimization and Applications **53**(2), 505–526 (2012)
13. Liuzzi, G., Lucidi, S., Rinaldi, F.: Derivative-free methods for mixed-integer constrained optimization problems. Journal of Optimization Theory and Applications **164**(3), 933–965 (2015)
14. Lucidi, S., Piccialli, V., Sciandrone, M.: An algorithm model for mixed variable programming. SIAM Journal on Optimization **15**(4), 1057–1084 (2005)
15. Moré, J., Wild, S.: Benchmarking derivative-free optimization algorithms. SIAM Journal on Optimization **20**(1), 172–191 (2009)
16. Müller, J., Shoemaker, C.A., Piché, R.: So-mi: A surrogate model algorithm for computationally expensive nonlinear mixed-integer black-box global optimization problems. Computers & Operations Research **40**(5), 1383–1400 (2013)
17. Müller, J., Shoemaker, C.A., Piché, R.: So-i: a surrogate model algorithm for expensive nonlinear integer programming problems including global optimization applications. Journal of Global Optimization **59**(4), 865–889 (2014)
18. Newby, E., Ali, M.M.: A trust-region-based derivative free algorithm for mixed integer programming. Computational Optimization and Applications **60**(1), 199–229 (2015)
19. Porcelli, M., Toint, P.L.: Bfo, a trainable derivative-free brute force optimizer for nonlinear bound-constrained optimization and equilibrium computations with continuous and discrete variables. ACM Transactions on Mathematical Software **44**(1), 1–25 (2017)
20. V. Lukšan, J.V.: Test problems for nonsmooth unconstrained and linearly constrained optimization. Technical report VT798-00, Institute of Computer Science, Academy of Sciences of the Czech Republic (2000)