Math. Program., Ser. B (2012) 136:375–402
DOI 10.1007/s10107-012-0608-x

FULL LENGTH PAPER

A storm of feasibility pumps for nonconvex MINLP

Claudia D'Ambrosio · Antonio Frangioni · Leo Liberti · Andrea Lodi

Received: 14 August 2010 / Accepted: 12 February 2011 / Published online: 2 November 2012

Abstract One of the foremost difficulties in solving Mixed-Integer Nonlinear Programs, either with exact or heuristic methods, is to find a feasible point. We address this issue with a new feasibility pump algorithm tailored for nonconvex Mixed-Integer Nonlinear Programs. Feasibility pumps are algorithms that iterate between solving a continuous relaxation and a mixed-integer relaxation of the original problems. Such approaches currently exist in the literature for Mixed-Integer Linear Programs and convex Mixed-Integer Nonlinear Programs: both cases exhibit the distinctive property that the continuous relaxation can be solved in polynomial time. In nonconvex Mixed-Integer Nonlinear Programming such a property does not hold, and therefore special care has to be exercised in order to allow feasibility pump algorithms to rely only on local optima of the continuous relaxation. Based on a new, high level view of feasibility pump algorithms as a special case of the well-known successive projection method, we show that many possible different variants of the approach can be developed, depending on how several different (orthogonal) implementation choices are

This paper extends [16].

C. D'Ambrosio · L. Liberti
LIX, Ecole Polytechnique, Paris, France
e-mail: dambrosio@lix.polytechnique.fr

L. Liberti
e-mail: liberti@lix.polytechnique.fr

A. Frangioni
Dipartimento di Informatica, Università di Pisa, Pisa, Italy
e-mail: frangio@di.unipi.it

A. Lodi (✉)
DEI, Universitá di Bologna, Bologna, Italy
e-mail: andrea.lodi@unibo.it

taken. A remarkable twist of feasibility pump algorithms is that, unlike most previous successive projection methods from the literature, projection is "naturally" taken in two different norms in the two different subproblems. To cope with this issue while retaining the local convergence properties of standard successive projection methods we propose the introduction of appropriate *norm constraints* in the subproblems; these actually seem to significantly improve the practical performance of the approach. We present extensive computational results on the MINLPLib, showing the effectiveness and efficiency of our algorithm.

Keywords Feasibility pump · MINLP · Global optimization · Nonconvex NLP

Mathematics Subject Classification 90C11 Mixed integer programming · 90C26 Nonconvex programming, global optimization · 90C59 Approximation methods and heuristics

1 Introduction

Mixed-Integer Nonlinear Programming (MINLP) problems are mathematical programs of the following form:

$$\min \left\{ f(x, y) \ : \ g(x, y) \leq 0, \quad (x, y) \in \mathscr{X}, \quad x \in \mathbb{Z}^p \right\} \tag{1}$$

where x are integer decision variables, y are continuous decision variables, $\mathscr{X} \subseteq \mathbb{R}^{p+q}$ is a polyhedron (which possibly include variable bounds), $f : \mathbb{R}^{p+q} \to \mathbb{R}$ and $g : \mathbb{R}^{p+q} \to \mathbb{R}^m$. We remark that f can be assumed convex without loss of generality (if it were not, we might replace it by an added variable v and adjoin the constraint $f(x) - v \leq 0$ as a further component of g).

The exact solution of nonconvex MINLP is only possible for certain classes of functions f, g (e.g. if f is linear and g involve bilinear terms xy [2,11]). In general, the spatial Branch-and-Bound (sBB) algorithm is used to obtain ε-approximate solutions for a given positive constant ε. The sBB computes upper and lower bounds to the objective function value within sets belonging to an iteratively refined partition of the feasible region. The search is pruned when the lower bound on the current set is worse than the best feasible so far (the incumbent), when the problem restricted to the current set is infeasible, and when the two bounds for the current set are within ε. Otherwise, the current set is partitioned and the search continues recursively [6,35]. Heuristic approaches to solving MINLPs include Variable Neighbourhood Search [30], automatically tuned variable fixing strategies [7], Local Branching [31] and others; specifically, most exact approaches for convex MINLPs [8,21] work as heuristic for nonconvex MINLPs. In heuristic approaches, however, one of the main algorithmic difficulties connected to MINLPs is to find a feasible solution. From the worst-case complexity point of view, finding a feasible MINLP solution is as hard as finding a feasible Nonlinear Programming (NLP) solution, which is NP-hard [36].

In this paper we address the issue of MINLP feasibility by extending a well-known approach, namely the Feasibility Pump (FP) to the nonconvex MINLP case.

The FP algorithm was originally proposed for Mixed-Integer Linear Programming (MILP) [20], where f, g are linear forms, and then extended to convex MINLPs [8], where g are convex functions. In both cases the feasible region is partitioned so that two subproblems are iteratively solved: a problem P_1 involving the continuous variables y with relaxed integer variables x, and a problem P_2 involving both integer and continuous variables x, y targeting, through its objective function, the continuous solution of P_1. The two subproblems are iteratively solved, generating sequences of values for x and y. One of the main theoretical issues in FP is to show that these sequences do not cycle, i.e., are not periodic but converge to some feasible point (x, y). This is indeed the case for the FP version proposed for convex MINLP [8] where P_2 is a MILP, while cycling might happen for the original FP version proposed for MILP [20] where randomization is effectively (and cheaply) used as an escaping mechanism. In the FP for MILPs, P_1 is a Linear Program (LP) and P_2 a rounding phase; in the FP for convex MINLPs, P_1 is a convex NLP and P_2 a MILP iteratively updated with Outer Approximation (OA) constraints derived from the optimum of the convex NLP. In both cases one of the subproblems (P_1) can be solved in polynomial time; in the FP for convex MINLPs, P_2 is NP-hard in general. Extensions for both FPs exist, addressing solution quality in some cases [1] and CPU time in others [9]. The added difficulty in the extension proposed in this paper is that P_1 is a nonconvex NLP, and is therefore NP-hard: thus, in our decomposition, both subproblems are difficult, and special care has to be exercised in order to allow FP algorithms to rely only on local optima of the continuous relaxation.

A contribution of the present paper is to present FP algorithms as a special case of the well-known Successive Projection Method (SPM). By doing so we show that many possible different variants of the approach can be developed, depending on how several different (orthogonal) implementation choices are taken. A remarkable twist of FP algorithms is that, unlike most previous SPMs from the literature, projection is "naturally" taken in two different norms in P_1 and P_2. To cope with this issue while retaining the local convergence properties of standard SPMs we propose the introduction of appropriate *norm constraints* in the subproblems, an idea that could be generalized to other nonconvex applications of the SPM. In particular, adding a norm constraint to P_1, besides providing nice theoretical convergence properties, actually seem to significantly improve the practical performance of the approach.

The rest of this paper is organized as follows. In Sect. 2 we frame the FP algorithm within the class of Successive Projection Methods, describing their convergence properties. In Sect. 3 we discuss the use of different norms within the two subproblems of the FP algorithm. In Sect. 4 we list our solution strategies for both subproblems. In Sect. 5 we present comparative computational results illustrating the efficiency of the proposed approach. Section 6 concludes the paper.

2 A view on feasibility pumps

A hitherto seemingly unremarked fact is that FP algorithms are instantiations of a more general class of algorithms, called Successive Projection Methods, for finding a point in a set intersection $\mathscr{A} \cap \mathscr{B}$ under the (informal) assumption that "optimization over

each set separately is much easier than optimization over the intersection". SPMs have been proposed more than 60 years ago (cf. [37]), have found innumerable applications, and have been developed in very many variants: the excellent—and not particularly recent—survey of [5] provides more than one hundred references. The basic idea of SPM is to restate the feasibility problem in terms of the optimization problem

$$\min\{\|z - w\| \mid z \in \mathscr{A} \wedge w \in \mathscr{B}\}. \tag{2}$$

Given an initial point w^0, a SPM generates a sequence of iterates $(z^1, w^1), (z^2, w^2), \ldots$ defined as follows:

$$z^i \in \operatorname{argmin}\{\|z - w^{i-1}\| \mid z \in \mathscr{A}\} \tag{3}$$
$$w^i \in \operatorname{argmin}\{\|z^i - w\| \mid w \in \mathscr{B}\}, \tag{4}$$

where on first reading the norm could be assumed Euclidean. It is easy to see that, as discussed below, each iteration improves the objective function of (2), so that one can hope that the sequence will converge to some fixed point (z^*, w^*) where $z^* = w^*$, which therefore provides a positive answer to the feasibility problem. The standing assumption simply says that (3)–(4) are much easier problems than the whole of (2) is, and therefore that the iterative approach may make sense.

While in the vast majority of the applications $\mathscr{A}$ and $\mathscr{B}$ are "easy" convex sets, and it was intended that optimization was to be exact, our nonconvex FP setting also fits under the basic assumption of the approach. Indeed, $\mathscr{X}$ is the coarsest relaxation of the feasible region of (1) and let $C \subseteq \{1, \ldots, m\}$ be the set of constraint indices such that $g_i(x, y)$ is a convex function of (x, y) (note that these do not include the linear defining inequalities of $\mathscr{X}$, if any), and $N = \{1, \ldots, m\} \setminus C$. We denote the list of all convex constraints by g_C, so that $\mathscr{C} = \{(x, y) \mid g_C(x, y) \leq 0\} \subseteq \mathbb{R}^{p+q}$ also is a convex relaxation of the feasible region of (1). We also denote by g_N the constraints indexed by N and let $\mathscr{N} = \{(x, y) \mid g_N(x, y) \leq 0\}$. We remark that deciding whether $\mathscr{N}$ is empty involves the solution of a nonconvex NLP and is therefore a hard problem. This hardness, by inclusion, extends to the continuous relaxation of the feasible region $\mathscr{P} = \mathscr{C} \cap \mathscr{N} \cap \mathscr{X}$. Now, let $\mathscr{Z} = \{(x, y) \mid x \in \mathbb{Z}^p\}$, so that $\mathscr{I} = \mathscr{C} \cap \mathscr{X} \cap \mathscr{Z}$ is the relaxation of the feasible region involving all the convex and integrality constraints of (1). Deciding emptiness of $\mathscr{I}$ involves solving a convex MINLP and is therefore also hard, but for different reasons than $\mathscr{P}$. More specifically, solving nonconvex NLPs globally requires solving nonconvex NLPs locally as a sub-step, whereas solving convex MINLPs involves the solution of convex NLPs (globally) as a sub-step. The numerical difficulties linked to these two tasks are very different, in particular with respect to the reliability of finding the solution: with nonconvex NLPs, for example, Sequential Quadratic Programming (SQP) algorithms might yield an infeasibile linearization step even though the original problem is feasible. It therefore makes sense to decompose $\mathscr{F} = \mathscr{I} \cap \mathscr{P}$, the feasible region of (1), into its two components $\mathscr{I}$ and $\mathscr{P}$, in order to address each of the difficulties separately.

Thus, by taking e.g. $\mathscr{A} = \mathscr{P}$ and $\mathscr{B} = \mathscr{I}$ one can fit the FP approach under the generic SPM framework. Note that with this choice the (nonlinear) convex constraints

g_C are included in the definition of both $\mathscr{P}$ and $\mathscr{I}$ (although they can possibly be outer-approximated in the latter, as discussed below). This makes sense since $\mathscr{C}$ represents, in this context, an "easy" part of (1): adding it to either set of constraints do not fundamentally change the difficulty of the corresponding problems, while clearly helping to convey as much information of $\mathscr{F}$ as possible. Yet, other decompositions could make sense as well. For instance, one may alternatively set $\mathscr{B} = \mathscr{X} \cap \mathscr{Z}$ in order to keep P_2 a linear problem without having to resort to outer approximation techniques (assuming that C only contains the *nonlinear* convex constraints, with all the linear ones represented in $\mathscr{X}$). Alternatively, $\mathscr{B} = \mathscr{Z}$ could also make sense, since then P_2 actually simplifies to a simple rounding operation (the choice of the original FP [20], cf. §4.2). Thus, different variants of FP for (1) can be devised which can all be interpreted as special cases of SPM for proper choices of $\mathscr{A}$ and $\mathscr{B}$. Therefore, in the following we will keep the "abstract" notation with the generic sets $\mathscr{A}$ and $\mathscr{B}$ whenever we discuss general properties of the approach which do not depend on specific choices within the FP application.

Convergence of SPMs has been heavily investigated. An easy observation is that, directly from (3)–(4),

$$\|z^{i-1} - w^{i-1}\| \geq \|z^i - w^{i-1}\| \geq \|z^i - w^i\|,$$

i.e., the sequence given by $\{\delta_i = \|z^i - w^i\|\}$ is nonincreasing, hence the method is at least locally convergent (in the sense that $\delta_i \to \delta_\infty \geq 0$). Global convergence results can be obtained under several different conditions, typically requiring convexity of $\mathscr{A}$ and $\mathscr{B}$ (so that (2) is a convex feasibility problem) [5]. For instance, the original result of [37] for the case of hyperplanes (where projection (3)–(4) is so easy that it can be accomplished by a closed formula) proves convergence of the whole sequence to the point of $\mathscr{A} \cap \mathscr{B}$ closest to the starting point w^0. Convergence results (for the convex case) can be obtained for substantially more complex versions of the approach, although many of these results become particularly significant when the number of intersecting sets is (much) larger than two. For instance, the general scheme analyzed in [5] considers the iteration

$$v^i = \lambda_0^i\big((1-\alpha_0^i)v^{i-1} + \alpha_0^i P_{\mathscr{B}}(v^{i-1})\big) \;+\; \lambda_1^i\big((1-\alpha_1^i)v^{i-1} + \alpha_1^i P_{\mathscr{A}}(v^{i-1})\big)$$

where $\alpha_h^i \in [0,2]$ are the *(over/under)relaxation* parameters and $\lambda_h^i \geq 0$ are the *weights* for $h = 0, 1$, $\lambda_0^i + \lambda_1^i = 1$, and

$$P_{\mathscr{A}}(\bar{v}) \in \operatorname{argmin}\{\|v - \bar{v}\| \mid v \in \mathscr{A}\}, \quad P_{\mathscr{B}}(\bar{v}) \in \operatorname{argmin}\{\|\bar{v} - v\| \mid v \in \mathscr{B}\}.$$

Thus, SPMs can allow *weighted simultaneous projection* and *relaxation*; we mention in passing that these algorithms bear more than a casual resemblance with *subgradient methods* [18], as discussed in [5, §7]. The scheme (3)–(4) clearly corresponds to $\alpha_h^i = 1$ ("unrelaxed") and $\lambda^i_{(i \mod 2)} = 1$ ("cyclic control"), so that only one among the two projections actually need to be computed at any iteration ($z^i = v^{2i-1}$ and $w^i = v^{2i}$). While simultaneous projection is unlikely to be attractive in the FP setting, relaxation

is known to improve the practical performance of SPMs in some cases, and it could be considered.

All these global convergence results rely on convexity, but in our case the involved sets are far from being convex; convergence of an SPM applied to the nonconvex MINLP is therefore an issue. In order to consider the nonconvex case, the typical strategy is that of recasting SPMs in terms of *block Gauss-Seidel* approaches applied to the minimization of a block-structured objective function $Q(z, w)$. These approaches, based on the same idea to iteratively minimize over one block of variables at a time, can be shown to be (locally) convergent under much less stringent conditions than convexity, especially in the two-blocks case of interest here. Different convergence results, under different assumptions, can be found e.g. in [23,34] for the even more general setting where the objective function of is

$$L(z, w) = h(z) + Q(z, w) + k(w),$$

with $h : \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\}$ and $k : \mathbb{R}^q \to \mathbb{R} \cup \{+\infty\}$ proper lower semicontinuous functions, neither convex nor differentiable, and $Q : \mathbb{R}^{p+q} \to \mathbb{R}$ is *regular*, i.e.,

$$Q'(z, w; d_z, 0) \geq 0 \wedge Q'(z, w, 0, d_w) \geq 0 \Rightarrow Q'(z, w; d_z, d_w) \geq 0 \tag{5}$$

for all feasible z, w, where $Q'(z, w; d_z, d_w)$ is the directional derivative of Q at (z, w) along the direction (d_z, d_w). Smooth functions are regular, while in general nonsmooth ones are not. With stricter conditions on Q (e.g. C^1) the results can be extended and somewhat strengthened [3] to the *stabilized* version

$$z^i \in \operatorname{argmin}\{h(z) + Q(z, w^{i-1}) + \lambda_i \|z - z^{i-1}\|_2^2 \mid z \in \mathscr{A}\} \tag{6}$$
$$w^i \in \operatorname{argmin}\{Q(z^i, w) + k(w) + \mu_i \|w - w^{i-1}\|_2^2 \mid w \in \mathscr{B}\}, \tag{7}$$

where the penalty terms are added to discourage large changes in the current z, w iterates. The method (6)–(7) holds under mild assumptions such as upper and lower boundedness of λ_i and μ_i; the whole sequence (z^i, w^i) is shown to converge to a critical point of L (as opposed to only convergence of subsequences like in [23]).

One can then fit the FPs for the nonconvex MINLP case in the above setting by choosing e.g. $h = k = 0$, $Q(z, w) = \|z - w\|_2^2$, $\lambda_i = \mu_i = 0$, and $\mathscr{A}, \mathscr{B}$ in any one of the possible ways discussed above. Note that different variants could be also discussed, such as using non-zero $h(z)$ and $k(w)$ to include a weighted contribution of the objective function to try to improve on the quality of the obtained solutions, a-la [1]. Sticking to the simplest case, one has that the sequence defined by

$$(\bar{x}^i, \bar{y}^i) \in \operatorname{argmin}\{\|(x, y) - (\hat{x}^{i-1}, \hat{y}^{i-1})\| \mid g(x, y) \leq 0 \wedge (x, y) \in \mathscr{X}\} \tag{8}$$
$$(\hat{x}^i, \hat{y}^i) \in \operatorname{argmin}\{\|(x, y) - (\bar{x}^i, \bar{y}^i)\| \mid g_C(x, y) \leq 0 \wedge (x, y) \in \mathscr{X} \wedge x \in \mathbb{Z}^p\}. \tag{9}$$

(or any other appropriate splitting of the constraints) converges to a local optimum of the (nonconvex) problem (2). Thus, if $\delta_i \to \delta_\infty > 0$, then either $\mathscr{F} = \emptyset$ or the algorithm has converged to a critical point which is not a global minimum. Telling

these two cases apart is unfortunately difficult in general; however, because we are proposing a MINLP heuristic, rather than an exact algorithm, we shall typically assume the latter case holds, and we shall therefore employ some Global Optimization (GO) techniques to reach a putative global optimum.

It should be remarked that (3)–(4) is only the most straightforward application of SPM in our setting. A number of issues arises:

- The algorithm alternates between solving the nonconvex NLP (8) and the convex MINLP (9). In order to retain the local convergence property, both problems would need to be solved exactly: a difficult task in both cases.
- Being (9) a mixed-integer program, it would be very attractive to be able to use the efficient available MILP solvers to tackle it. However, in order to do that one would — as the very first step — need to substitute the Euclidean norms with "linear" ones (L_1, L_∞).
- In the standard FP approach [8,20] the distance is actually only measured on the integer (x) variables, as opposed to the full pair (x, y).

In the rest of the paper, we discuss several modifications to this approach in order to address the above issues and better exploit the structure of the problem at hand.

3 Using different norms

In this section we consider employing two different norms $\|\cdot\|_A$ and $\|\cdot\|_B$ in the two subproblems (3)–(4):

$$z^i \in \operatorname{argmin}\{\|z - w^{i-1}\|_A \mid z \in \mathscr{A}\} \tag{10}$$
$$w^i \in \operatorname{argmin}\{\|z^i - w\|_B \mid w \in \mathscr{B}\}. \tag{11}$$

The Euclidean norm is appropriate in (8) because of its smoothness property and because (8) is already nonlinear. In the case of (9), however, the L_1 or L_∞ norms yield a convex MINLP whose objective function can be linearized by means of standard techniques [28]. Provided the constraints indexed by C are linear, or they are outer linearized, (9) then becomes a MILP, which allows use of the available efficient off-the-shelf general-purpose solvers. Replacing the norm in (9), however, prevents us from establishing monotonicity of the sequence $\{\delta_i \mid i \in \mathbb{N}\}$: assuming $A = 2$ and (say) $B = \infty$, for example, one uses $\|\cdot\|_A \geq \|\cdot\|_B$ to derive

$$||z^i - w^i||_A \geq ||z^{i+1} - w^i||_A \geq ||z^{i+1} - w^i||_B \geq ||z^{i+1} - w^{i+1}||_B,$$

but nothing ensures $||z^{i+1} - w^{i+1}||_B \geq ||z^{i+1} - w^{i+1}||_A$. A way to deal with this case is to replace (10) by

$$z^i \in \operatorname{argmin}\{\|z - w^{i-1}\|_A \mid z \in \mathscr{A} \wedge \|z - w^{i-1}\|_B \leq \beta\|z^{i-1} - w^{i-1}\|_B\}, \tag{12}$$

for $\beta \in (0, 1]$. This implies monotonicity for all $\beta \leq 1$ and strict monotonicity for all $\beta < 1$:

$$||z^i - w^i||_B \geq \beta||z^i - w^i||_B \geq ||z^{i+1} - w^i||_B \geq ||z^{i+1} - w^{i+1}||_B.$$

For $B = \infty$, the required reformulation for (10) only amounts to restricting variable bounds; as this restricts the feasible region, it is also expected to facilitate the task of solving (12). Local convergence — that is, no cycling—of the sequence of iterates generated by the FP algorithm given by (12) and (11) can be established by ensuring that the sequence $\|z^i - w^i\|_B$ converges. A convergence failure might occur if (12) becomes infeasible because of the restricted variable bounds.[1] This shows that

$$\min\{||z - w^{i-1}||_B \mid z \in \mathscr{A}\} \;\geq\; \beta||z^{i-1} - w^{i-1}||_B,$$

which in turn implies that, for $\beta \approx 1$, z^{i-1} is a good candidate for a local minimum, leading to the choice $z^i = z^{i-1}$. If the mixed-integer iterate (11) also cannot improve the objective, then (z^i, w^i) can be assumed to be a local minimum; this case will be dealt with later.

We remark that the fact that

$$\begin{aligned} \forall z \in \mathscr{A} \quad ||z - \bar{w}||_B \geq ||\bar{z} - \bar{w}||_B \\ \forall w \in \mathscr{B} \quad ||\bar{z} - w||_B \geq ||\bar{z} - \bar{w}||_B \end{aligned} \tag{13}$$

is not sufficient to ensure that $(\bar{z}, \bar{w})$ is a local minimum: this is because $\|\cdot\|_B$ is not regular in the sense of (5) if $B \in \{1, \infty\}$. Indeed, it is clear that for $Q(z, w) = g(z-w)$, one has $Q'(z, w; d_z, d_w) = g(z - w; d_z - d_w)$, which means that $Q'(z, w; d_z, 0) = g(z-w; d_z)$ and $Q'(z, w; 0, d_w) = g(z-w; -d_w)$. Thus, for (z, w) such that $||\cdot||_B$ is not differentiable at $z - w$, it is not difficult to construct a counterexample to (5). One is shown in Fig. 1 for the case $B = 1$, where $Q'(z, w; d_z, 0) = Q'(z, w; 0, d_w) = 0$, but $Q'(z, w; d_z, d_w) = -1$.

Example 1 Based on Fig. 1, we construct an example of nonconvergence of the FP with $A = 2$ and $B = 1$. Let $\mathscr{A} = \{(x, y) \mid x \geq 3\}$ and $\mathscr{B} = \{(x, y) \mid 2x \leq y\}$, and consider $\bar{z} = (3, 4)$ and $\bar{w} = (2, 4)$. It is easy to verify that

$$\begin{aligned} \bar{z} \in \operatorname{argmin}\{\|(x, y) - (2, 4)\|_2 \mid (x, y) \in \mathscr{A}\} \\ \bar{w} \in \operatorname{argmin}\{\|(3, 4) - (x, y)\|_1 \mid (x, y) \in \mathscr{B}\}, \end{aligned}$$

which implies that $(\bar{z}, \bar{w})$ is a fixed point for the sequence generated by the FP. However, $(\bar{z}, \bar{w})$ is not a local minimum of (2): by moving a step of length 2 along the feasible direction $(d_z, d_w) = (0, 1, 1/2, 1)$ we obtain $z' = (3, 6)$ and $w' = (3, 6)$, and $\|z' - w'\|_1 = \|z' - w'\|_2 = 0 < 1 = \|\bar{z} - \bar{w}\|_1 = \|\bar{z} - \bar{w}\|_2$. □

[1] These failures happen in practice as shown in Sect. 5.4.

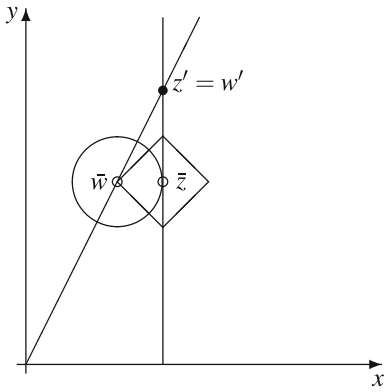

Fig. 1 Non-reguarity of $||\cdot||_1$

Hence, the modification (12) of the FP still guarantees convergence of the δ_i sequence, and therefore (at least for $\beta < 1$) ensures that no cycling can occur. Convergence may occur to a local minimum when using "nonsmooth" norms such as L_1 and L_∞ even if $\mathscr{A}$ and $\mathscr{B}$ were convex, but this is not a major issue since the sets are nonconvex, and therefore there is no guarantee of convergence to a global minimum anyway. Other mechanisms in the algorithm (cf. §4.2) are designed to take care of this.

3.1 Partial norms

A structural property of the specific nonconvex MINLP setting is that whenever $z = (x, y) \in \mathscr{A}$ has the property that there exists some $\tilde{z} = (x, \tilde{y}) \in \mathscr{B}$, then $z \in \mathscr{F}$; in other words, the difficulty of optimizing over $\mathscr{B}$ is given by the integer constrained variables x. Thus, for our purposes we can consider

$$(\bar{x}^i, \bar{y}^i) \in \operatorname{argmin}\{||x - \hat{x}^{i-1}|| \mid (x, y) \in \mathscr{A}\} \tag{14}$$

$$(\hat{x}^i, \hat{y}^i) \in \operatorname{argmin}\{||\bar{x}^i - x|| \mid (x, y) \in \mathscr{B}\}. \tag{15}$$

instead of (10)–(11). This means that we need only to consider the distance between the projection of $\mathscr{A}$ and $\mathscr{B}$ on the x-subspace, as opposed to the distance between the full (x, y) iterates. This does not have any significant impact on the approach. From the theoretical viewpoint, it just says that the function $Q(z, w)$ in (6)–(7) is constant (hence, smooth) on a subset of the variables, i.e.,

$$Q(\ (\bar{x}, \bar{y})\ ,\ (\hat{x}, \hat{y})\ ) = ||\bar{x} - \hat{x}||.$$

Things are, in theory, rather more complex when two different norms $\|\cdot\|_A$ and $\|\cdot\|_B$ are used in (14)–(15) (respectively), since then there is no longer a well-defined function Q to minimize. However, this is as well the case when using different norms on the whole iterates, as advocated in the previous section. From the practical viewpoint,

using different partial norms only amounts to the fact that the norm constraint in (12) actually reads

$$\|x - \hat{x}^{i-1}\|_B \leq \beta \|\bar{x}^{i-1} - \hat{x}^{i-1}\|_B \tag{16}$$

which of course is still enough to guarantee the (strict, if $\beta < 1$) monotonicity of the sequence $\{\delta_i\}$. This still provides all the local convergence properties that the FP requires.

4 Approximate solution of the subproblems

The convergence theory for SPMs would require solving (8) and (9) to global optimality. As already remarked, this is extremely challenging and not very likely to be effective in the context of what, overall, remains a heuristic method, which at any rate does not provide any theoretical guarantee of success. Furthermore, even if the subproblems were actually solved to global optimality, several variants of the FP approach—most notably, those employing two different norms–would not still entirely fit into the theoretical framework for which convergence proofs are readily available. This frees us to consider several different options and strategies to solve both (8) and (9), as discussed in this section, which give rise to "a storm" of many different configurations that we extensively tested computationally. The results are reported in Sect. 5, either in detail for the most successful algorithms or in summary for the unsuccessful ones.

4.1 Addressing the nonconvex NLP (8)

As already mentioned solving (8) to global optimality is difficult by itself, mainly because of the nonconvex constraints. Indeed, if only convex constraints were present, every local optimum would be guaranteed to be also a global optimum. For this reason, considering both the convex and nonconvex constraints does not make the subproblem much harder in practice than only considering the nonconvex ones. Although applying GO techniques to obtain a provably optimal solution would likely be too time consuming, we still attempt to solve the problem globally by using two different approaches:

1. a simple stochastic multi-start approach [33] in which the NLP solver is provided with different randomly generated starting points in order to try to escape from possible local minima;
2. a Variable Neighborhood Search (VNS) [24] scheme [29,30].

In general, finding any feasible solution for (8) such that $||\bar{x} - \hat{x}^{i-1}||_A < ||\bar{x}^{i-1} - \hat{x}^{i-1}||_B$ is enough to retain the monotonicity property of the sequence; thus, the solution process to (8) can be terminated as soon as this happens. Failure to obtain this condition may lead to declare the failure of a local phase, without identifying a feasible solution, even if one could be found if a globally optimal (or, at least, better) solution for (8) be determined, as shown in Fig. 2. In particular, we consider a nonconvex MINLP in two variables. On the x-axis we have an integer variable,

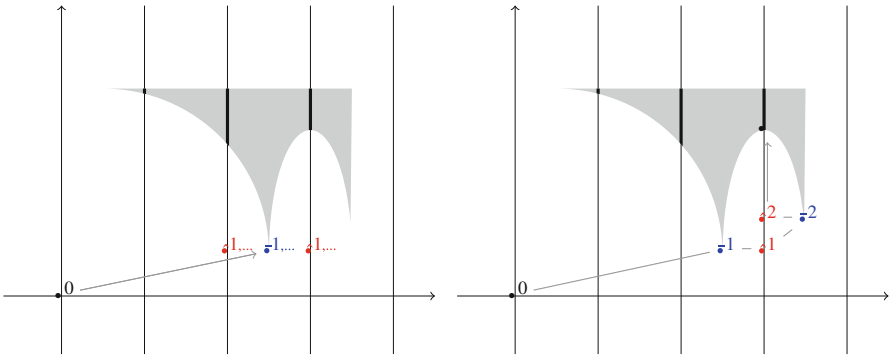

Fig. 2 Solving (8) heuristically (*left*) or to global optimality (*right*): helping to prevent cycling

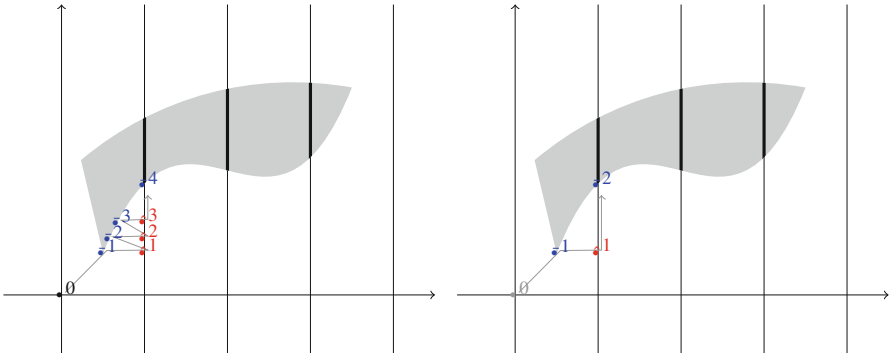

Fig. 3 Solving (8) without (*left*) and with (*right*) the fixing strategy b: accelerating convergence

while on the y-axis a continuous one. The gray part is the feasible region of the NLP relaxation of P while the set of the bold lines represents the feasible region of P. The symbols $\hat{\bullet}$ represent solutions $\hat{x}^i$ where i is the iteration number. Similarly, the symbols $\bar{\bullet}$ represent solutions $\bar{x}^i$ where i is again the iteration number. The figure shows that only requiring local optimality in (8) can lead to cycling: on the left, the a local optimum $\hat{\bullet}$ to (8) does not allow the algorithm to proceed to $\bar{\bullet}^2$ and eventually to the feasible MINLP solution $\hat{\bullet}^2$ (on the right).

However, sometimes both strategies 1 and 2 might fail in finding a feasible solution for (8) (for example due to a time limit, see Sect. 5) and that can happen even if they claim the returned solution is NLP feasible. In such case we experimented two options:

a. we define (9) by using any infeasible solution of (8);
b. we *fix* the integer variables x and we solve again (locally) a modified version of problem (8) in which the objective function is replaced by the original objective of (1).

The fixing strategy b might improve convergence speed, as shown in Fig. 3: if x_1 is fixed to the value given by $\hat{\bullet}^1$ then the next NLP solution is likelier to directly lead to the feasible MINLP region.

Finally, as discussed in Sect. 3, one might implement the overall FP scheme by using the Euclidean norm for (8) and a different norm in (9), like L_1 or L_∞, to simplify it. As previously discussed, this may well impair the only remaining (weak) convergence property of the approach, i.e., monotonicity of the sequence $\{\delta_i\}$, making it harder to declare that a "local optimum" has reached. For this case, two options can be implemented:

i. we forget about such a difference in norms and we hope for the best;
ii. we amend (8) by the norm constraint (16), and solve it as usual. We remark here that preliminary computational experiments have shown that the value of β does not strongly influence the results, thus we used $\beta = 1$ in the computational results of Sect. 5.

In summary, three main decisions have to be taken to define and solve (8):

I. solution algorithm: multi-start (1. above) versus VNS (2. above),
II. additional fixing step: NO (a. above) versus YES (b. above), and
III. norm correction: NO (i. above) versus YES (ii. above).

4.2 Addressing the convex MINLP (9)

The first decision that has to be taken for addressing problem (9) concerns the norm to use in the objective function, i.e., how to formulate (9) in practice.

1. Of course, the most trivial option is to keep the Euclidean norm so as (9) is a convex MINLP.
2. As discussed in Sect. 3, the main alternative is to employ either the L_1 or the L_∞ norm in the objective function so that it can be linearly reformulated in standard ways (via the introduction of a few auxiliary continuous variables). This is in the attempt to replace (9) with a MILP relaxation, because MILP solution technology is currently more advanced than its convex MINLP equivalent. This, however, requires the constraints to be linearized as well. This can be done by means of standard *Outer Approximation* approaches. That is, assuming C contains only *nonlinear* convex constraints (the linear ones being left in $\mathscr{X}$), one can approximately solve (9) at the generic iteration $i \in \mathbb{N}$ by means of its MILP relaxation

$$\min \|\bar{x}^i - x\|_B \tag{17}$$

$$g_\ell(\bar{x}^k, \bar{y}^k) + \nabla g_\ell(\bar{x}^k, \bar{y}^k) \begin{pmatrix} x - \bar{x}^k \\ y - \bar{y}^k \end{pmatrix} \leq 0 \quad \ell \in \bar{C}^k,\ k \leq i \tag{18}$$

$$(x, y) \in \mathscr{X}, \quad x \in \mathbb{Z}^p \tag{19}$$

where the norm B in (17) can be either L_1 or L_∞ and $\bar{C}^k \subseteq C$ is the set of convex nonlinear constraints that are active at $(\bar{x}^k, \bar{y}^k)$. In other words, one keeps collecting the classical *Outer Approximation cuts* [19] (18) along the iterations and uses them to define a polyhedral outer approximation of $\mathscr{I}$. Note that while

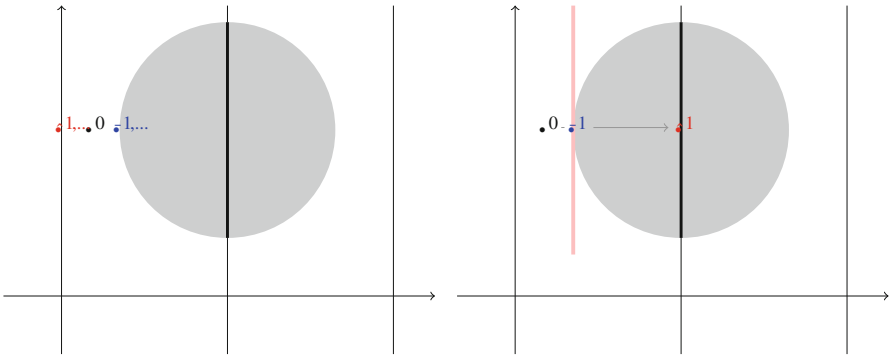

Fig. 4 Solution of (9) without (*left*) and with (*right*) OA cuts: helping to prevent cycling

(18) could seem to require that each g_ℓ for $\ell \in C$ be a differentiable function, this is only assumed for the sake of notational simplicity: notoriously, subgradients of nondifferentiable convex functions can be used as well (e.g. [17]).

In both cases, we employ partial norms as detailed in Sect. 3.1 so as to take into account in the objective function only its integral part.

The second decision is how to formulate the feasibility space of problem (9), i.e., how to deal with the original set of constraints and relaxing them if needed. This depends on the first decision as well, i.e., on the objective function, either 1. or 2. above, which has been selected. Of course, such a decision is inherently linked to the solution algorithm.

a. If the Euclidean norm is used in (9), then we investigate three options:
   1. we solve the convex MINLP as is by means of a sophisticated general-purpose MINLP solver, in our case BONMIN solver [10],
   2. we solve a convex mixed-integer quadratic problem (MIQP) relaxation of the MINLP. Precisely, the MIQP is obtained by using the objective function[2] $\min \|\bar{x}^i - x\|_2$ instead of (17) but with the same set of (linear) constraints (18)-(19). This is done to simplify the problem and being able to use a sophisticated general-purpose MIQP solver, in our case CPLEX [25].
   3. we remove all constraints (18)–(19), only keeping $x \in \mathbb{Z}^p$ and bound constraints, and solve (9) by rounding. This is in the spirit of both [20] and [9].

b. If instead the L_1/L_∞ norm is used and the MILP relaxation (17)–(19) is defined, we solve the MILP as is by means of a sophisticated general-purpose MILP solver, in our case CPLEX [25].

The third decision is how to address the issue of cycling. Indeed, because problem (9) only takes into account the subset of convex constraints (or a relaxation of them in the MILP case) the resulting FP algorithm might cycle, i.e., visit the same mixed-integer solution more than once. Note that if (1) is instead a convex MINLP, OA cuts are shown to be sufficient to guarantee that the FP algorithm does not cycle [8] as shown for example in Fig. 4.

[2] Note that again the objective function is defined in the MIQP case only on the integer variables.

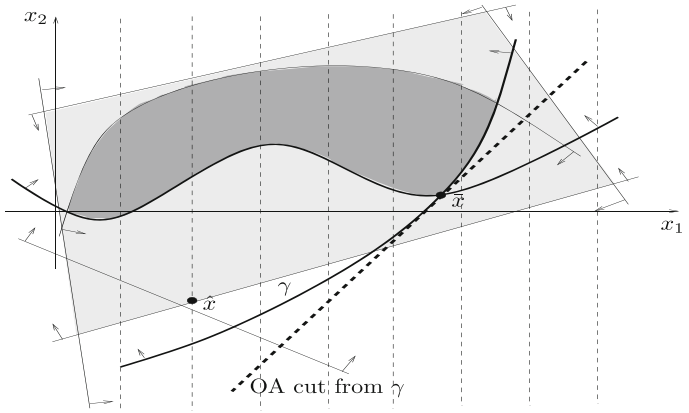

Fig. 5 The OA cut from γ does not cut off $\hat{x}$

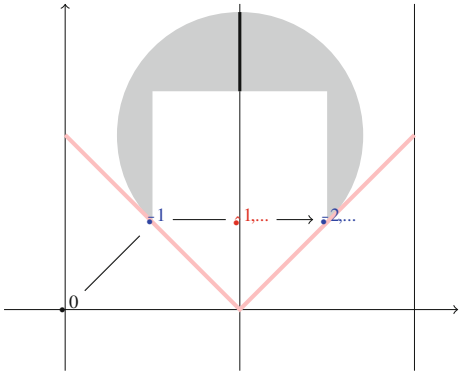

Fig. 6 OA cuts may not prevent the FP from cycling

In the nonconvex case, however, OA cuts are not enough, as discussed in Example 2. In addition, in the testbed we used to computationally test our approach, the number of OA cuts we could generate is somehow limited as discussed in detail in Sect. 5.1.

Example 2 In Fig. 5 a nonconvex feasible region and its current linear approximation are depicted. Let us consider $\bar{x}$ being the current solution of subproblem (8). In this case, only one OA cut can be generated, i.e., the one corresponding to convex constraint γ. However, it does not cut off $\hat{x}$, i.e., the solution of (9) at the previous iteration. In this example, the FP would not immediately cycle, because $\hat{x}$ is not the solution of (9) which is closest to $\bar{x}$. This shows that there is a distinction between cutting off and cycling. In general, however, failure to cut off previously visited integer solutions might lead to cycling, as shown in Fig. 6. □

One elegant possibility to prevent cycling is that of adding *no-good cuts* at iteration i to make $(\hat{x}^k, \hat{y}^k)$ infeasible for all $k < i$. This is possible if (as it happens in some of the variants) any of the minimum distance problems is solved (even if only approximately)

with an *exact* approach, which not only provides good feasible solutions, but also a *lower bound* on the optimal value of the problem to provide a guarantee of the accuracy. Indeed, if the solution method proves that the inequality

$$||x - \hat{x}^i||_A \geq \varepsilon \tag{20}$$

is satisfied for all $(x, y) \in \mathscr{A}$, then one has

$$\mathscr{A} \cap \mathscr{B} = (\mathscr{A} \cap \{(x, y) \ : \ (20)\}) \cap \mathscr{B} = \mathscr{A} \cap (\mathscr{B} \cap \{(x, y) \ : \ (20)\}).$$

In other words, the *nonlinear and nonconvex "cut"* (20) can be added to $\mathscr{B}$ without changing the feasible set of the problem. The interesting part is that, of course, $\hat{x}^i$ *violates* (20), and therefore (20) provides—at least in theory–a convenient globalization mechanism.

No-good cuts [17] were originally introduced in [4] with the name of "canonical" cuts and recently used within the context of MINLP [17,32]. If x are binary variables and $\|\cdot\|$ is the L_1 norm, we can take $\varepsilon = 1$ and reformulate (20) linearly as

$$\sum_{\substack{j \leq p \\ \hat{x}_j = 0}} x_j + \sum_{\substack{j \leq p \\ \hat{x}_j = 1}} (1 - x_j) \geq 1.$$

For general integer variables, an exact linear reformulation is given, for example, in [17] and involves adding $2p$ new continuous variables, p new binary variables and adding $3p + 1$ linear equations to the problem. Thus, the size of such a reformulation could rapidly become prohibitive in the context of an iterative method like FP. This is why no-good cuts are used in a limited form in our scheme and we instead implement two alternative, less elegant, approaches:

i. We employ a tabu list in order to prevent a MILP solver from finding the same solutions $(\hat{x}, \hat{y})$ at different iterations.
ii. We configure our solver to find a pool of solutions from which we choose the best non-forbidden one.

Clearly, the issue of preventing the FP scheme to cycle is not confined to the solution of problem (9) but is more a globalization strategy. Indeed, problem (8) could in turn be amended by no-good cuts in the form $||x - \hat{x}^{i-1}||_2^2 \geq \varepsilon$ which are reverse-convex constraints not different from those already in (8). However, we decided to concentrate our attention to (9) for two reasons. On the one side, this is the way both previous FP algorithms worked, namely the one for MILP, through random flipping of the rounding step, and that for convex MINLP, by means of OA cuts. On the other hand, the value to be assigned to ε would be any lower bound greater than 0 on the optimal solution of (9). However, we never really solve such a problem to optimality and in at least one case, the rounding option 3 above, we do not compute any lower bound either.

In summary, three main decisions have to be taken to define and solve (9):

I. the norm to be used in the formulation of (9): L_2 (1. above) versus L_1/L_∞ (2. above),

II. how to define the feasible region of (9) and solve it: MINLP (1 above) versus MIQP (2 above) versus rounding (3 above) or MILP (b above), and
III. how to avoid cycling: tabu list (ii. above) versus solution pool (ii. above).

5 Computational results

In this section we discuss the outcome of our extensive computational investigation.

5.1 Computational setting

The algorithms were implemented within the AMPL environment [22]. We chose to use this framework to make it easy to change subsolver. In practice, the user can select the preferred solver to solve NLPs, MINLPs, MIQPs or MILPs, exploiting their advantages.

We also use a new solver/reformulator called ROSE (Reformulation Optimization Software Engine, see [28,27]), of which we exploit the following features:

- Model analysis: getting information about nonlinearity and convexity of the constraints and integrality requirements of the variables, so as to define subproblems (8) and (9).
- Solution feasibility analysis: necessary to verify feasibility of the provided solutions.
- OA cut generation: necessary to update (9). In order to determine whether a constraint is convex, ROSE performs a recursive analysis of its expression tree [26] to determine whether it is an affine combination of convex functions. We call such a function "evidently convex" [28]. Evident convexity is a stricter notion than convexity: evidently convex functions are convex but the converse may not hold. Thus, it might happen that a convex constraint is labeled nonconvex; the information provided is in any case safe for our purposes, i.e., we generate OA cuts only from constraints which are certified to be convex. Unfortunately, the number of problems in the testbed (see next section) in which we are able to generate OA cuts is limited, around 15% of them, surely because of such a conservative (but *safe*) policy adopted by ROSE.

5.2 FP variants and preliminary results

Because of the multiple options which can be put in place to solve both (8) and (9), we had to implement and test more than twenty FP versions/variants to assert the effectiveness of each of the algorithmic decisions discussed in the two previous sections. Some of these options have been ruled out after a preliminary set of experiments involving 243 MINLP instances from MINLPlib [12] and used, among others, in [16,30]. Only 65 among such 243 instances are those in which the open-source Global Optimization solver COUENNE 0.1 [6] (available from COIN-OR [14]) is *unable* to find a feasible

solution within a time limit of 5 min on an Intel Xeon 2.4 GHz with 8 GB RAM running Linux.

Thus, the goal of the preliminary set of computational experiments was twofold. On the one side, we wanted to be quick and competitive on the "easy" instances, i.e., the 178 instances on which COUENNE is able to find a solution within 5 min of CPU time. This is because FP can clearly be used as a stand-alone heuristic algorithm for nonconvex MINLP, and must be competitive with a general-purpose solver used as well as a heuristic, i.e., truncated within a short time limit. That was achieved by the "best" FP versions/variants that will be discussed in the remainder of the section. To give an example, the version denoted as FP-1 (see Sect. 5.4) finds a feasible solution for 156 of the 178 "easy" instances within 5 min, encounters numerical troubles in 13 of them (because of the NLP solver) and requires more than 5 min in the remaining 9 instances. Because COUENNE 0.1 (like most GO solvers) mainly implemented simple heuristics based on reformulations and linearizations, it would have been relatively easy to recover those few instances with longer computing times (9) by ad-hoc policies. On the other hand, however, we wanted to be effective (in possibly longer computing times) on the 65 "hard" instances where simple heuristics and partial enumeration failed. In particular, FP should be effective on the instances in which the nonlinear aspects of the problems play a crucial role, thus suggesting its fruitful integration within COUENNE or any other GO solver (as happened for FP algorithms in MILP). Indeed, the current trunk version of COUENNE is more sophisticated in terms of heuristics also due to our investigation preliminary reported in [15,16] and some results at the of Sect. 5.4 seem promising in this concern.

The FP variants which did not "survive" the preliminary tests[3] are those that, at the same time, did not perform particularly well in the "easy"instances and did not add anything special on the "hard" ones. Namely,

1. Solving (8) by VNS was always inferior with respect to solve it by the stochastic multi-start approach. Such a poor performance of the VNS approach might be due to its iterative implementation within AMPL: at each iteration, a different search space is defined, starting from a small one and incrementing it so that at the last iteration the entire feasible region is considered. In particular, this approach seems to be too "conservative" with respect to the previous solution.
2. The additional fixing step which can be performed in case of fail when solving (8) by fixing the integer variables has a slight positive effect when the norm constraint is added while turns out to be crucial in case it is not. In a sense the theoretical convergence guaranteed by the use of norm constraints seems to make problems (8) easier, thus the benefit of the fixing step is particularly high if such constraints are not added. We then decided to always include the fixing step as well.
3. In case the Euclidean norm is kept in problem (9), we decided to solve the convex MIQP instead of the convex MINLP. The main reason (besides some technical issues related to modify a convex MINLP solver like BONMIN to implement mechanisms to prevent cycling) is that the number of evidently convex constraints as

[3] No detailed computational results are reported here for the preliminary computational investigation, some of them are discussed in detail in [16].

discovered by ROSE is very limited in the testbed. Thus, if the constraints in (9) are linear, then the MIQP solver of CPLEX is clearly more efficient than a fully general convex MINLP solver line BONMIN.

4. Preventing cycling by using a pool of solutions was always inferior with respect to use the tabu list. Again, this might be due to the lack of flexibility of the (nice) solution pool feature of CPLEX 11 that we used in our experiments. Every time we need to solve (9), we ask CPLEX to produce a number of solutions equal to the number of tabu list solutions plus one. Once obtained the solutions pool, we analyze the solutions starting from the first and set $(\hat{x}^i, \hat{y}^i)$ as the first solution of the pool which is not present in the tabu list. However, we have to consider the two following drawbacks: (i) the solution pool is populated after the branch and bound is finished. Because we have a time limit for solving (9), it is not guaranteed that we would have a number of solutions sufficient to provide a non-forbidden solution (especially because providing a solution pool is a time-consuming feature); (ii) we cannot force CPLEX to measure the diversity of the solutions in the pool by neglecting the continuous part of the problem. Unfortunately, CPLEX can provide us a set of solutions which has the same integer values, but different continuous values. More generally, it might happen that only forbidden solutions are generated, for example if the continuous relaxation of (9) is integer feasible but forbidden. In this case the solution would be discarded, but no further solution can be generated.

Due to the above discussion, the only surviving subproblems to be solved are nonconvex NLPs and convex MILPs and MIQPs. The NLP solver used is IPOPT 3.5 trunk [13], while the MILP and MIQP solvers are those of CPLEX 11 [25]. Before ending the section we need to specify two more implementation details for the surviving FP variants.

Implementing a tabu list in CPLEX. Discarding a solution in the tabu list within the CPLEX branch and bound is possible using the incumbent callback function. The tabu list is stored in a text file which is then exchanged between AMPL and CPLEX. Every time CPLEX finds an integer feasible solution, a specialized incumbent callback function checks whether the new solution appears in the tabu list. If this is the case, the solution is rejected, otherwise the solution is accepted. CPLEX continues executing until either the optimal solution (excluding those forbidden) is found or a time limit is reached. In the case where an integer solution found by CPLEX at the root node appears in the tabu list, CPLEX stops and no new integer feasible solution is provided to FP[4]. In such a case, we amend problem (9) with a no-good cut [17] which excludes the solution and we call CPLEX again.

Avoid cycling when solving (9) *by rounding.* When the MILP relaxation of (9) is solved by rounding to the nearest integer the fractional values of vector $\bar{x}$, the methods for preventing the cycling cannot be implemented in the way we described above. The method adopted is taken from the original FP paper [20]: whenever a forbidden solution is found, the algorithm randomly flip some of the integer values so as to obtain a new solution.

[4] Note that this is the same issue discussed in the solution pool case.

Table 1 FP variants

Variant	Problem (8)			Problem (9)		
	Algorithm	Fixing step	Norm constraint	Norm	Algorithm	Cycling
FP-1	multi-start	YES	YES	L_1	MILP	tabu list
FP-2	multi-start	YES	NO	L_1	MILP	tabu list
FP-3	multi-start	YES	YES	L_2	Rounding	tabu list
FP-4	multi-start	YES	N/A	L_2	MIQP	tabu list
FP-5	multi-start	YES	YES	L_∞	MILP	tabu list
FP-6	multi-start	YES	NO	L_∞	MILP	tabu list

Table 2 Comparing the *six* FP variants, aggregated results

	FP-1	FP-2	FP-3	FP-4	FP-5	FP-6
Successes	49	45	22	23	44	46
Successes alone	0	3	0	0	1	1
Time limit reached	11	14	42	32	2	12
Fails	5	6	1	9	19	7
Wins	26	20	10	4	10	8
Time geomean	151.02	104.45	17.59	76.14	23.25	14.99

5.3 Code tuning

The algorithm terminates after the first MINLP feasible solution is found or a time limit is reached. The parameters are set in the following way: time limit of 2 h of user CPU time, the absolute feasibility tolerance to evaluate constraints is 1e-6, and the relative feasibility tolerance is 1e-3 (used if absolute feasibility test fails). The tabu list length was set adaptively to a value which was inversely proportional to the number of integer variables of the instance, i.e., the number of values to be stored for each solution of the tabu list. The value was 60,000 divided by the number of integer variables. The actual mean value, over the full set of 243 instances, of the solutions stored in the tabu list was 35.

5.4 Results

The *six* surviving FP variants have been extensively tested on the full set of 243 MINLP instances and, in particular, we discuss the results on the 65 "hard" instances introduced in Sect. 5.2. More precisely, the six variants have the characteristics reported in Table 1.

Table 2 reports the aggregated results on the 65 "hard" instances. In particular, we consider for each FP variant the number of times the algorithm terminated with a feasible solution within the 2-h of user CPU time limit (successes), the number of times the algorithm was the only one to find a feasible solution (successes alone), the number of times the time limit was reached without a feasible solution (time limit

reached), the number of times the algorithm encountered numerical issues (fails), the number of times the algorithm found the best—smallest—solution (wins) and the geometric mean of the computing time for the solved instances (time geomean).

The detailed results are reported in Tables 3 and 4 where, for each variant, we give the solution value (value), the computing time (time) and the number of iterations (it.s) which are roughly equal to the number of problems (8) and (9) solved. In case of numerical issues for a pair instance / FP variant, we report in all entries for such an instance some "++", whereas in case of time limit reached the entry value is set to "-" (while we correctly report the computing time of 7,200 CPU seconds and the number of iterations within such a time).

The results of Tables 2, 3 and 4 show that FP-1 is the most successful FP variant and is remarkably able to find a feasible solution in limited CPU time on 75 % of the "hard" instances in the testbed. A direct comparison with the closest variant, namely FP-2, shows that the use of the norm constraint is useful: although FP-1 does not dominate FP-2, it is overall superior on all entries and there are many instances in which FP-2 converges slowly whereas FP-1 reaches feasibility in a very small number of iterations. Variant FP-3 is very fast but seems to be a bit "unsophisticated" for those instances which look more difficult (in the "hard" testbed). However, it might be a viable option for a "cheap" FP variant executed extensively within a GO solver. Variant FP-4 does not look—at the moment—very competitive, although it is not fully dominated because it finds the smallest solution four times, in one case (deb8) a much smaller one, with respect to the other variants. One relevant issue for FP-4 seems that the MIQP solved as problem (9) is time consuming thus allowing only a limited number of FP iterations. Things might change in the future, depending on the solver or its settings. Finally, variants FP-5 and FP-6 are very close to FP-1 and FP-2, respectively, and they indeed lead to similar results. Specifically, FP-5, compared to FP-1, seems to have much more numerical troubles (due to the NLP solve, see Sect. 3) and is inferior in terms of quality of the solutions obtained (wins) but is much faster. Instead, variant FP-6 is almost equivalent, perhaps superior, to FP-2, the two main differences being the number of wins (20 for FP-2 with respect to 8 for FP-6) and the speed (104.45 CPU seconds for FP-2 with respect to 14.99 for FP-6). Overall, both FP-5 and FP-6 seem promising for further investigation.

Concerning the interaction of FP-1 with the GO solver COUENNE (or any other), note that in 14 cases FP-1 finds a feasible solution within 1 minute of CPU time (in 24 cases within 5 min), thus suggesting a profitable integration within the solver.

6 Conclusion

We have presented the theoretical foundation of an abstract Feasibility Pump scheme interpreted as a Successive Projection Method in which, roughly speaking, the set of constraints of the original problem is split (possibly in different ways) in two sets and the overall algorithm aims at deciding if the feasibility space given by the intersection of such two sets is empty. Such a scheme has been specialized for dealing with nonconvex Mixed-Integer Nonlinear Programming problems, the hardest class of (deterministic) optimization problems.

Table 3 Comparing FP variants FP-1, FP-2, FP-3 and FP-4, detailed results

Instance	FP-1			FP-2			FP-3			FP-4		
	Value	Time	it.s	Value	Time	it.s	Value	Time	it.s	Value	Time	it.s
beuster	++	++	++	++	++	++	–	7,200	28	++	++	++
csched2a	−102,002.02	5	2	−102,867.73	4	2	++	++	++	−112,174.73	624	2
csched2	−120,042.73	138	2	−120,066.02	241	2	–	7,200	41	++	++	++
deb10	++	++	++	++	++	++	223.29	25	14	++	++	++
deb6	234.78	197	29	237.11	4	4	290.90	7	2	235.81	9	4
deb7	411.00	139	4	345.76	10	3	419.78	218	8	451.05	13	2
deb8	8,453,005,065.59	23	2	185,839,836.37	2	1	8,453,005,005.71	30	1	416,332.32	3	1
deb9	444.67	33	2	425.34	16	4	444.67	39	1	438.39	59	2
detf1	11,497.56	368	2	15,976.03	131	2	8,455.75	961	1	15,976.03	731	2
eg_all_s	223.14	27	3	100,003.77	52	5	94,165.69	10	1	++	++	++
eg_disc2_s	65,822.96	7	1	100,004.34	5	1	65,822.96	7	1	100,004.34	5	1
eg_disc_s	94,165.42	8	1	100,003.69	7	1	94,165.42	8	1	100,003.69	7	1
eg_int_s	94,167.12	10	1	100,005.46	7	1	94,167.12	10	1	100,005.46	7	1
fo8_ar25_1	994,207.06	185	124	–	7,200	6	–	7,200	3,211	–	7,200	2
fo8_ar3_1	994,235.33	784	367	–	7,200	6	–	7,200	3,210	–	7,200	2
fo8	894,678.42	9	8	1,400,000.00	1,543	533	–	7,200	3,110	1,400,000.00	860	444
fo9_ar2_1	1,136,279.49	1,286	167	–	7,200	8	–	7,200	2,619	–	7,200	2
fo9_ar25_1	1,136,997.73	635	97	–	7,200	14	–	7,200	2,620	–	7,200	2
fo9_ar4_1	9,959.68	202	68	1,599,990.28	4,212	699	–	7,200	2,616	–	7,200	2
fo9_ar5_1	1,428,148.20	17	2	1,599,993.97	14	2	–	7,200	2,610	–	7,200	2
fo9	1,006,964.21	61	32	1,600,000.00	221	153	–	7,200	2,552	1,600,000.00	1,387	657
johnall	−201.15	615	2	−201.29	614	2	−201.16	2	2	−221.92	618	2
lop97ic	–	7,200	9	–	7,200	120	–	7,200	94	–	7,200	13

Table 3 continued

Instance	FP-1			FP-2			FP-3			FP-4		
	Value	Time	it.s	Value	Time	it.s	Value	Time	it.s	Value	Time	it.s
mbtd	91.33	4,266	2	98.53	4,045	2	–	7,200	3	1,000,005.67	5,834	2
nuclear104	–	7,200	2	++	++	++	–	7,200	2	++	++	++
nuclear10a	–	7,200	2	–	7,200	5	–	7,200	3	–	7,200	4
nuclear10b	–	7,200	2	–	7,200	4	–	7,200	2	–	7,200	3
nuclear14a	−1.09	1,602	3	−1.11	2,641	173	–	7,200	38	–	7,200	14
nuclear14b	−1.10	646	2	−1.10	686	7	–	7,200	34	−1.09	1,922	81
nuclear14	−1.12	1,645	3	−1.12	847	15	–	7,200	107	–	7,200	14
nuclear24a	−1.09	1,602	3	−1.11	2,730	173	–	7,200	38	–	7,200	14
nuclear24b	−1.05	2,626	4	−1.09	1,584	59	–	7,200	35	−1.09	1,959	81
nuclear24	−1.12	1,655	3	−1.12	1,649	51	–	7,200	101	–	7,200	14
nuclear25a	−1.06	6,501	8	–	7,200	372	–	7,200	39	–	7,200	14
nuclear25b	−0.99	1,666	3	−1.05	707	8	–	7,200	33	–	7,200	19
nuclear49a	–	7,200	8	−1.11	6,266	67	–	7,200	15	–	7,200	12
nuclear49b	−1.06	4,367	4	−1.13	4,980	27	–	7,200	8	–	7,200	34
nuclear49	−1.14	1,165	2	–	7,200	13	–	7,200	5	–	7,200	11
nuclearva	−1.01	133	2	−1.01	244	2	−1.01	496	35	–	7,200	71
nuclearvb	−1.03	614	2	−1.02	710	3	−1.01	1,107	181	−1.02	613	2
nuclearvc	−1.00	2,064	6	−0.99	110	4	−0.99	1,702	149	–	7,200	51
nvs08	24,116.94	0	1	24,119.23	1	1	24,116.94	0	1	24,119.23	0	1
nvs20	146,475,177.22	0	1	138,691,481.67	0	1	146,475,177.22	0	1	138,691,481.67	0	1
nvs24	−342.20	1	4	−536.20	1	4	−517.80	0	2	–	7,200	2
o8_ar4_1	5,822,973.45	22	10	8,199,969.73	736	236	–	7,200	3,214	–	7,200	2
o9_ar4_1	6,877,522.82	198	59	8,199,964.62	6,206	698	–	7,200	2,611	–	7,200	2
qapw	468,078.00	372	2	460,118.00	637	2	–	7,200	21	464,259.68	684	2

Table 3 continued

Instance	FP-1			FP-2			FP-3			FP-4		
	Value	Time	it.s	Value	Time	it.s	Value	Time	it.s	Value	Time	it.s
saa_2	11,497.56	377	2	15,976.03	252	2	8,455.75	978	2	15,976.03	721	2
space25a	661.97	376	3	650.69	245	18	–	7,200	124	1,124.32	612	2
space25	661.97	413	3	650.69	773	18	–	7,200	45	1,124.38	619	2
space960	24,070,000.00	3,629	7	–	7,200	7	–	7,200	8	–	7,200	8
super1	++	++	++	++	++	++	–	7,200	18	–	7,200	2
super2	++	++	++	++	++	++	–	7,200	18	–	7,200	2
super3	++	++	++	++	++	++	–	7,200	18	–	7,200	2
super3t	–	7,200	18	–	7,200	19	–	7,200	20	++	++	++
tln12	–	7,200	14	–	7,200	10	–	7,200	2,403	–	7,200	2
tls12	–	7,200	10	–	7,200	10	–	7,200	445	–	7,200	9
tls2	5.30	720	6	5.30	1	5	–	7,200	6,569	++	++	++
tls5	–	7,200	24	22.50	58	21	–	7,200	2,783	–	7,200	100
tls6	–	7,200	9	–	7,200	26	–	7,200	2,226	–	7,200	17
tls7	–	7,200	9	37.80	2,892	38	–	7,200	1,464	–	7,200	8
uselinear	1,951.37	188	1	227,751.06	47	1	1,951.37	187	1	1,951.37	48	1
var_con10	475.36	449	54	463.17	12	5	562.62	29	4	++	++	++
var_con5	397.21	110	16	315.16	6	3	434.66	21	3	++	++	++
waste	62,025.78	50	1	306,239.04	19	1	62,025.78	50	1	306,232.46	70	2

Table 4 Comparing FP variants FP-1, FP-2, FP-5 and FP-6, detailed results

Instance	FP-1			FP-2			FP-5			FP-6		
	Value	Time	it.s	Value	Time	it.s	Value	Time	it.s	Value	Time	it.s
beuster	++	++	++	++	++	++	++	++	++	++	++	++
csched2a	−102,002.02	5	2	−102,867.73	4	2	−102,932.87	1	2	−102,932.87	0	2
csched2	−120,042.73	138	2	−120,066.02	241	2	−120,064.55	2	2	−120,066.02	2	2
deb10	++	++	++	++	++	++	331.39	1	2	++	++	++
deb6	234.78	197	29	237.11	4	4	950.00	1	2	949.94	2	2
deb7	411.00	139	4	345.76	10	3	447.77	9	2	449.38	7	2
deb8	8,453,005,065.59	23	2	185,839,836.37	2	1	185,839,836.37	3	1	185,839,836.37	2	1
deb9	444.67	33	2	425.34	16	4	443.97	8	2	434.33	7	2
detf1	11,497.56	368	2	15,976.03	131	2	15,976.27	113	2	15,976.03	110	2
eg_all_s	223.14	27	3	100,003.77	52	5	15.45	406	4	100,003.00	85	7
eg_disc2_s	65,822.96	7	1	100,004.34	5	1	100,004.34	8	1	100,004.34	5	1
eg_disc_s	94,165.42	8	1	100,003.69	7	1	100,003.69	11	1	100,003.69	8	1
eg_int_s	94,167.12	10	1	100,005.46	7	1	100,005.46	12	1	100,005.46	8	1
fo8_ar25_1	994,207.06	185	124	–	7,200	6	1,399,992.47	18	16	1,399,991.20	240	222
fo8_ar3_1	994,235.33	784	367	–	7,200	6	1,333,314.45	10	9	1,399,992.80	2	4
fo8	894,678.42	9	8	1,400,000.00	1,543	533	1,400,000.00	3	6	1,400,000.00	11	29
fo9_ar2_1	1,136,279.49	1,286	167	–	7,200	8	1,599,990.04	1,245	58	1,599,990.04	1,896	642
fo9_ar25_1	1,136,997.73	635	97	–	7,200	14	1,599,989.57	527	264	1,599,988.83	343	232
fo9_ar4_1	9,959.68	202	68	1,599,990.28	4,212	699	55,204.13	1,267	227	1,599,989.86	20	10
fo9_ar5_1	1,428,148.20	17	2	1,599,993.97	14	2	1,598,839.97	3	2	1,599,992.22	3	2
fo9	1,006,964.21	61	32	1,600,000.00	221	153	1,600,000.00	13	16	1,600,000.00	47	76
johnall	–201.15	615	2	–201.29	614	2	–201.15	1	2	–201.15	1	2
lop97ic	–	7,200	9	–	7,200	120	–	7,501	20	5,401.15	3,652	97
mbtd	91.33	4,266	2	98.53	4,045	2	89.53	5,253	2	89.53	6,103	2

Table 4 continued

Instance	FP-1			FP-2			FP-5			FP-6		
	Value	Time	it.s	Value	Time	it.s	Value	Time	it.s	Value	Time	it.s
nuclear104	–	7,200	2	++	++	++	++	++	++	++	++	++
nuclear10a	–	7,200	2	–	7,200	5	++	++	++	–	9,078	4
nuclear10b	–	7,200	2	–	7,200	4	−1.10	3,794	2	–	8,769	4
nuclear14a	−1.09	1,602	3	−1.11	2,641	173	++	++	++	−1.10	142	19
nuclear14b	−1.10	646	2	−1.10	686	7	−1.00	30	2	−1.07	25	3
nuclear14	−1.12	1,645	3	−1.12	847	15	−1.12	11	2	–	7,202	251
nuclear24a	−1.09	1,602	3	−1.11	2,730	173	++	++	++	−1.10	271	19
nuclear24b	−1.05	2,626	4	−1.09	1,584	59	−1.00	28	2	−1.07	45	3
nuclear24	−1.12	1,655	3	−1.12	1,649	51	−1.12	13	2	–	7,200	216
nuclear25a	−1.06	6,501	8	–	7,200	372	++	++	++	−1.05	214	16
nuclear25b	−0.99	1,666	3	−1.05	707	8	−1.06	18	2	−1.07	303	13
nuclear49a	–	7,200	8	−1.11	6,266	67	++	++	++	–	7,334	18
nuclear49b	−1.06	4,367	4	−1.13	4,980	27	−1.03	454	2	–	7,288	47
nuclear49	−1.14	1,165	2	–	7,200	13	−1.14	439	2	–	7,492	30
nuclearva	−1.01	133	2	−1.01	244	2	++	++	++	−1.01	5	4
nuclearvb	−1.03	614	2	−1.02	710	3	−1.02	2	2	−1.02	2	2
nuclearvc	−1.00	2,064	6	−0.99	110	4	−0.99	2	2	−0.99	3	3
nvs08	24,116.94	0	1	24,119.23	1	1	24,119.23	0	1	24,119.23	0	1
nvs20	146,475,177.22	0	1	138,691,481.67	0	1	138,691,481.67	0	1	138,691,481.67	0	1
nvs24	−342.20	1	4	−536.20	1	4	−506.60	360	4	−413.80	0	3
o8_ar4_1	5,822,973.45	22	10	8,199,969.73	736	236	8,171,278.66	947	305	8,199,970.33	85	108
o9_ar4_1	6,877,522.82	198	59	8,199,964.62	6,206	698	43,754.16	1,800	285	8,199,964.17	20	10
qapw	468,078.00	372	2	460,118.00	637	2	459,102.00	611	2	459,102.00	610	2

Table 4 continued

Instance	FP-1			FP-2			FP-5			FP-6		
	Value	Time	it.s	Value	Time	it.s	Value	Time	it.s	Value	Time	it.s
saa_2	11,497.56	377	2	15,976.03	252	2	15,976.27	117	2	15,976.03	116	2
space25a	661.97	376	3	650.69	245	18	++	++	++	671.09	5	5
space25	661.97	413	3	650.69	773	18	++	++	++	671.09	17	4
space960	24,070,000.00	3,629	7	–	7,200	7	24,070,000.00	1,513	7	24,070,000.00	2,558	9
super1	++	++	++	++	++	++	++	++	++	++	++	++
super2	++	++	++	++	++	++	++	++	++	++	++	++
super3	++	++	++	++	++	++	++	++	++	++	++	++
super3t	–	7,200	18	–	7,200	19	++	++	++	++	++	++
tln12	–	7,200	14	–	7,200	10	–	7,215	1,114	–	7,203	1,034
tls12	–	7,200	10	–	7,200	10	++	++	++	–	7,280	92
tls2	5.30	720	6	5.30	1	5	++	++	++	15.30	3	11
tls5	–	7,200	24	22.50	58	21	++	++	++	–	7,201	504
tls6	–	7,200	9	–	7,200	26	++	++	++	–	7,211	497
tls7	–	7,200	9	37.80	2,892	38	++	++	++	–	8,171	19
uselinear	1,951.37	188	1	227,751.06	47	1	227,751.06	58	1	227,751.06	96	1
var_con10	475.36	449	54	463.17	12	5	589.80	4	2	589.43	4	2
var_con5	397.21	110	16	315.16	6	3	532.55	3	2	532.50	5	2
waste	62,025.78	50	1	306,239.04	19	1	306,239.04	23	1	306,239.04	38	1

Because the devil is in the details, we analyzed a large number of options for (i) formulating and solving the two distinct problems originated by the above split and (ii) guaranteeing convergence of the global algorithm. The result has been more than twenty FP variants which have been computationally tested on a large number of MINLP instances from the literature to assert the viability of FP both as a stand-alone approximation algorithm and as a primal heuristic within a global optimization solver. Six especially interesting of these variants have been discussed in detail and extensive results have been presented on a set of 65 "hard" instances. The results show that feasibility pumps are indeed successful in finding feasible solutions for nonconvex MINLPs.

Acknowledgments One of the authors (LL) is grateful for the following financial support: ANR 07-JCJC-0151 "ARS"and 08-SEGI-023 "AsOpt"; Digiteo Emergence "ARM", Emergence "PASO", Chair "RMNCCO"; Microsoft-CNRS Chair "OSD". The remaining three authors are grateful to the other author (LL) for not making them feel too "poor". We thank two anonymous referees for a careful reading and useful comments. Part of this work was conducted while the first author was a post-doctoral student at the ISyE, University of Wisconsin-Madison and DEIS, University of Bologna. The support of both institutions is strongly acknowledged.

References

1. Achterberg, T., Berthold, T.: Improving the feasibility pump. Discrete Opt. **4**, 77–86 (2007)
2. Al-Khayyal, F.A., Sherali, H.D.: On finitely terminating branch-and-bound algorithms for some global optimization problems. SIAM J. Opt. **10**, 1049–1057 (2000)
3. Attouch, H., Bolte, J., Redont, P., Soubeyran, A.: Proximal Alternating Minimization and Projection Methods for Nonconvex Problems: An Approach Based on the Kurdyka-Lojasiewicz Inequality. Math. Oper. Res. **35**, 438–457 (2010)
4. Balas, E., Jeroslow, R.: Canonical cuts on the unit hypercube. SIAM J. Appl. Math. **23**, 61–69 (1972)
5. Bauschke, H.H., Borwein, J.M.: On projection algorithms for solving convex feasibility problems. SIAM Rev. **38**, 367–426 (1996)
6. 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)
7. Berthold, T., Gleixner, A.: Undercover primal MINLP heuristic. In: Bonami, P., Liberti, L., Miller, A., Sartenaer, A. (eds.) Proceedings of the European Workshop on Mixed-Integer Nonlinear Programming, pp. 103–113. Université de la Méditerranée, Marseille (2010)
8. Bonami, P., Cornuéjols, G., Lodi, A., Margot, F.: A feasibility pump for mixed integer nonlinear programs. Math. Prog. **119**, 331–352 (2009)
9. Bonami, P., Gonçalves, J.: Primal heuristics for mixed integer nonlinear programs. Technical report, IBM (2008)
10. Bonami, P., Lee, J.: BONMIN user's manual. IBM Corporation, Technical report (June 2007)
11. Bruglieri, M., Liberti, L.: Optimal running and planning of a biomass-based energy production process. Energy Policy **36**, 2430–2438 (2008)
12. Bussieck, M.R., Drud, A.S., Meeraus, A.: MINLPLib—a collection of test models for mixed-integer nonlinear programming. INFORMS J. Comput. **15**, 114–119 (2003)
13. COIN-OR. Introduction to IPOPT: a tutorial for downloading, installing, and using IPOPT (2006)
14. COUENNE. https://projects.coin-or.org/Couenne, v. 0.1
15. D'Ambrosio, C.: Application-oriented Mixed Integer Non-Linear Programming. PhD thesis, DEIS, Università di Bologna (2009)
16. D'Ambrosio, C., Frangioni, A., Liberti, L., Lodi, A.: Experiments with a feasibility pump approach for nonconvex MINLPs. In: Festa, P. (ed.) Symposium on Experimental Algorithms, volume 6049 of LNCS, pp. 350–360. Springer, Heidelberg (2010)
17. D'Ambrosio, C., Frangioni, A., Liberti, L., Lodi, A.: On interval-subgradient cuts and no-good cuts. Oper. Res. Lett. **38**, 341–345 (2010)

18. d'Antonio, G., Frangioni, A.: Convergence analysis of deflected conditional approximate subgradient methods. SIAM J. Opt. **20**, 357–386 (2009)
19. Duran, M., Grossmann, I.: An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math. Prog. **36**, 307–339 (1986)
20. Fischetti, M., Glover, F., Lodi, A.: The feasibility pump. Math. Prog. **104**, 91–104 (2005)
21. Fletcher, R., Leyffer, S.: Solving mixed integer nonlinear programs by outer approximation. Math. Prog. **66**, 327–349 (1994)
22. Fourer, R., Gay, D., Kernighan, B.: AMPL: A Modeling Language for Mathematical Programming, 2nd edn. Duxbury Press, Brooks, Cole Publishing Co., Florence, KY (2003)
23. Grippo, L., Sciandrone, M.: On the convergence of the block nonlinear Gauss-Seidel method under convex constraints. Oper. Res. Lett. **26**, 127–136 (2000)
24. Hansen, P., Mladenović, N.: Variable neighbourhood search: principles and applications. Eur. J. Oper. Res. **130**, 449–467 (2001)
25. ILOG. ILOG CPLEX 11.0 User's Manual. ILOG S.A., Gentilly, France (2008)
26. Liberti, L.: Writing global optimization software. In: Liberti, L., Maculan, N. (eds.) Global Optimization: From Theory to Implementation, pp. 211–262. Springer, Berlin (2006)
27. Liberti, L., Cafieri, S., Savourey, D.: Reformulation Optimization Software Engine. In: Fukuda, K., van der Hoeven, J., Joswig, M., Takayama, N. (eds.) Mathematical Software, vol. 6327, pp. 303–314. LNCS Springer, New York (2010)
28. Liberti, L., Cafieri, S., Tarissan, F.: Reformulations in mathematical programming: A computational approach. In: Abraham, A., Hassanien, A.-E., Siarry, P., Engelbrecht, A. (eds.) Foundations of Computational Intelligence, vol. 3, pp. 153–234. Studies in Computational Intelligence, Springer, Berlin (2009)
29. Liberti, L., Dražic, M.: Variable neighbourhood search for the global optimization of constrained NLPs. In: Proceedings of GO Workshop, Almeria, Spain (2005)
30. Liberti, L., Mladenović, N., Nannicini, G.: A good recipe for solving MINLPs. In: Maniezzo, V., Stützle, T., Voß, S. (eds.) Hybridizing metaheuristics and mathematical programming, vol. 10, pp. 231–244. Annals of Information Systems, Springer, New York (2009)
31. Nannicini, G., Belotti, P.: Local branching for MINLPs. Technical report, CMU (2009)
32. Nannicini, G., Belotti, P.: Rounding-based heuristics for nonconvex MINLPs. Technical report, CMU (2009)
33. Schoen, F.: Two-phase methods for global optimization. In: Pardalos, P.M., Romeijn, H.E. (eds.) Handbook of Global Optimization, vol. 2, pp. 151–177. Kluwer, Dordrecht (2002)
34. Tseng, P.: Convergence of a block coordinate descent method for nondifferentiable minimization. J. Opt. Theory Appl. **109**, 475–494 (2001)
35. Tuy, H.: Convex Analysis and Global Optimization. Kluwer, Dordrecht (1998)
36. Vavasis, S.A.: Nonlinear Optimization: Complexity Issues. Oxford University Press, Oxford (1991)
37. von Neumann, J.: Functional Operators, Vol. II. Princeton University Press, Princeton (1950)