Mathematical Programming B manuscript No.
(will be inserted by the editor)

A Storm of Feasibility Pumps for Nonconvex MINLP

Claudia D'Ambrosio[1,2] · **Antonio Frangioni**[3] · **Leo Liberti**[4] · **Andrea Lodi**[1]

Received: ? / Accepted: ?

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 successive projection 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 the main innovations in this paper are tailored algorithmic methods to overcome such a difficulty. We present extensive computational results on the MINLPLib, showing the effectiveness and efficiency of our algorithm.

Keywords feasibility pump · MINLP · global optimization · nonconvex NLP

1 Introduction

Mixed Integer Nonlinear Programming (MINLP) problems are Mathematical Programs (MP) of the following form:

$$\left.\begin{array}{rl} \min & f(x,y) \\ & g(x,y) \leq 0 \\ & x \in X \cap \mathbb{Z}^n \subseteq \mathbb{R}^p \\ & y \in Y \subseteq \mathbb{R}^q, \end{array}\right\} [P] \tag{1}$$

where x are integer decision variables, y are continuous decision variables, X, Y are two polyhedra (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

This paper extends [12].

[1]: DEIS, Università di Bologna, Italy E-mail: {c.dambrosio,andrea.lodi}@unibo.it ·
[2]: ISyE, University of Wisconsin-Madison, USA E-mail: cdambrosio@wisc.edu ·
[3]: Dipartimento di Informatica, Università di Pisa, Italy E-mail: frangio@di.unipi.it ·
[4]: LIX, École Polytechnique, France E-mail: liberti@lix.polytechnique.fr

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 [2,10]). 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 [29,5]. Heuristic approaches to solving MINLPs include Variable Neighbourhood Search [24], automatically tuned variable fixing strategies [6], Local Branching [25] and others; specifically, most exact approaches for convex MINLPs [16, 7] 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 [30].

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) [15], where f, g are linear forms, and then extended to convex MINLPs [7], 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 [7] where P_2 is a MILP, while cycling might happen for the original FP version proposed for MILP [15] 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 [8]. 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.

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

2 An Abstract Feasibility Pump

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 for all $i \in C$, and $N = \{1,\ldots,m\} \setminus C$. We denote the list of all convex constraints by g_C, and let $\mathscr{C} = \{(x,y) \mid g_C(x,y) \leq 0\} \subseteq \mathbb{R}^{p+q}$ be a convex relaxation of the feasible region of P. 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 property, by inclusion, extends to the continuous relaxation of the feasible region $\mathscr{P} = \mathscr{C} \cap \mathscr{N}$. We now let $\mathscr{I} = \mathscr{C} \cap \{(x,y) \mid x \in \mathbb{Z}^p\}$ be a relaxation of the feasible region involving the convex and integrality constraints of P. 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 P, into its two components $\mathscr{I}$ and $\mathscr{P}$, in order to address each separately.

FP algorithms are instantiations of a more general class of algorithms called Successive Projection Methods (SPM), targetting the problem of deciding emptyness of a set intersection $\mathscr{A} \cap \mathscr{B}$. This is equivalent to the following problem:

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

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

$$\forall i \geq 1 \quad z^i \in \operatorname{argmin}\{\|z-w^{i-1}\| \mid z \in \mathscr{A}\} \tag{3}$$

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

where the norm is assumed Euclidean for now. The advantage of an SPM, compared to solving (2) directly, is that it only requires optimization over $\mathscr{A}$, $\mathscr{B}$ separately. By (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 $\delta_i = \|z^i-w^i\|$ for all $i \geq 1$ is nonincreasing, hence the method is locally convergent (in the sense that $\delta_i \to \delta_\infty \geq 0$). The method is also globally convergent to the unique optimal solution of (2) when $\mathscr{A}$, $\mathscr{B}$ are convex, as $\|\cdot\|$ is strictly convex [18]. In the nonconvex MINLP case the involved sets are far from being convex; convergence of an SPM applied to $\mathscr{I} \cap \mathscr{P}$ is therefore an issue.

Local convergence of SPM is proved in [28] under mild assumptions; again, this requires the Euclidean norm. The proof, as in the case of [18], actually works for the more general case where the objective function of (2) is

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

where $h : \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\}$ and $k : \mathbb{R}^q \to \mathbb{R} \cup \{+\infty\}$ are 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. When Q is C^1 the results can be extended [3] to the stabilized version:

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

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 sequence (z^i, w^i) is shown to converge to a critical point of L. In the nonconvex MINLP case, where $h = k = 0$ and Q is given by $\|z - w\|_2^2$, this means finding a local optimum of the (nonconvex) problem (2). Thus, if $\delta_i \to \delta_\infty > 0$, then either $\mathscr{A} \cap \mathscr{B} = \emptyset$ or the algorithm converges 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.

The straightforward application of SPM in our setting is:

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

The corresponding FP algorithm alternates between solving the nonconvex NLP (8) and the convex MINLP (9). In order to retain the local convergence property of (3)-(4), both problems need to be solved exactly: a difficult task in both cases. In the rest of the paper, we discuss several modifications to this approach in order to better exploit problem structure.

3 Using Different Norms

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

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

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 that can be reformulated exactly to a MILP by means of standard techniques [22], provided the constraints indexed by C are linear. 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 $||z^{i+1} - w^{i+1}||_B \geq ||z^{i+1} - w^{i+1}||_A$. We deal with this case by replacing (10) by:

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

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 also facilitates 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. 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

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

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 $B = 1$ or ∞. 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 Figure 1 for the case $B = 1$, where $Q'(z,w;d_z,0) = Q'(z,w;0,d_w) = 0$, but $Q'(z,w;d_z,d_w) = -1$.

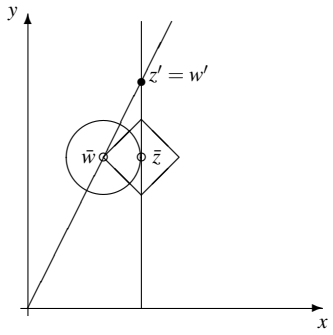

Fig. 1 Non-reguarity of $||\cdot||_1$

Example 1 Based on Figure 1, we construct an example of nonconvergence of the FP with $A = 2$ and $B = 1$: we define $\mathscr{A} = \{(x,y) \mid x \geq 3\}$ 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 $\|z' - w'\|_1 = \|z' - w'\|_2 = 0 < 1 = \|\bar{z} - \bar{w}\|_1 = \|\bar{z} - \bar{w}\|_2$. □

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 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 will consider

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

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

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.

4 Approximate solution of the subproblems

In practice, solving (8) and (9) pose different challenges; both should be solved to global optimality in order for the FP to have good convergence properties, yet both are difficult problems. Thus, in the following we discuss several options and strategies we used to solve both (8) and (9). We conducted extensive computational tests with all possible configurations. The results are reported in Section 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 in itself. Although applying GO would be too time consuming, we still attempt to solve the problem globally by using two different approaches:

1. a simple stochastic multi-start approach [27] 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) [19] scheme [23,24].

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

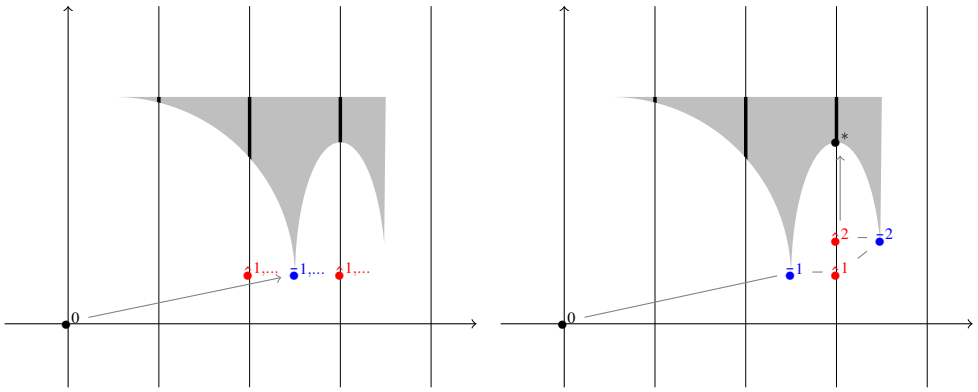

Fig. 2 Solving (8) heuristically (left) or to global optimality (right): helping to prevent cycling.

be found if a globally optimal (or, at least, better) solution for (8) be determined, as shown in Figure 2.

However, sometimes both strategies 1 and 2 might fail in finding a feasible solution for (8) (for example due to a time limit, see Section 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 prevent slow convergence, as shown in Figure 3.

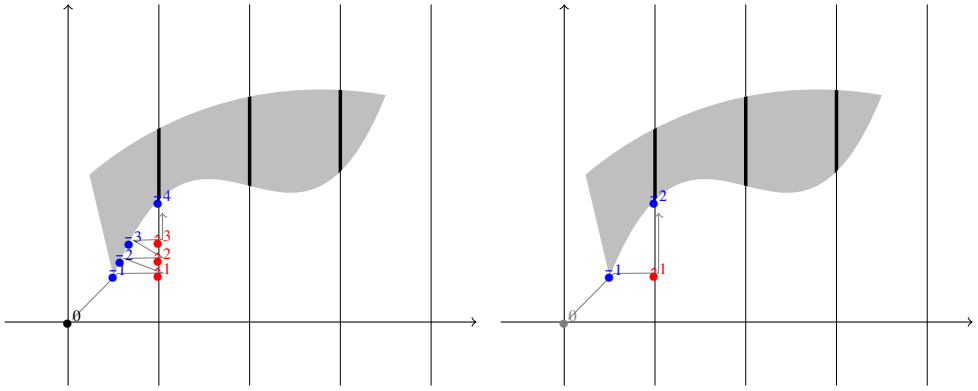

Fig. 3 Solving (8) without (left) and with (right) the fixing strategy b: accelerating convergence.

Finally, as discussed in Section 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 (see next section). If this is the 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 $\|x-\hat{x}^{i-1}\|_B \leq \beta\|\bar{x}^{i-1}-\hat{x}^{i-1}\|_B$ (see (12)), and solve it as usual[1].

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 Section 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. In addition, we also make use of partial norms as detailed in Section 3.1 so as to take into account in the objective function only its integral part.
   As for the constraints, we partition the index set C of convex constraints into C_L and C_N, where g_ℓ is an affine form for all $\ell \in C_L$, and g_ℓ is a nonlinear convex function for all $\ell \in C_N$. For all iteration indices $k \in \mathbb{N}$ we let $\bar{C}_N^k \subseteq C_N$ be the set indexing convex nonlinear constraints that are active at $(\bar{x}^k, \bar{y}^k)$.
   Thus, the MILP relaxation of (9) at iteration $i \in \mathbb{N}$ reads as follows:

$$\min \|\bar{x}^i - x\|_1 \tag{16}$$

$$\forall k \leq i, \ell \in \bar{C}_N^k \quad g_\ell(\bar{x}^k,\bar{y}^k) + \nabla g_\ell(\bar{x}^k,\bar{y}^k)\begin{pmatrix} x-\bar{x}^k \\ y-\bar{y}^k \end{pmatrix} \leq 0 \tag{17}$$

$$\forall \ell \in C_L \quad g_\ell(x,y) \leq 0 \tag{18}$$

$$x \in X \cap \mathbb{Z}^p \tag{19}$$

$$y \in Y, \tag{20}$$

   where the norm in (16) has been selected to be L_1 (instead of L_∞) due to preliminary computational experiments. Constraints (17) are the classical Outer Approximation cuts [14].

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 [9],

[1] 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.

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 (16) but with the same set of (linear) constraints (17)-(20). This is done to simplify the problem and being able to use a sophisticated general-purpose MIQP solver, in our case CPLEX [20].
3 we remove all constraints (17)-(20), only keeping $x \in \mathbb{Z}^p$ and bound constraints, and solve (9) by rounding. This is in the spirit of both [15] and [8].

b. If instead the L_1 norm is used and the MILP relaxation (16)-(20) is defined, we solve the MILP as is by means of a sophisticated general-purpose MILP solver, in our case CPLEX [20],

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 [7] as shown for example in Figure 4.

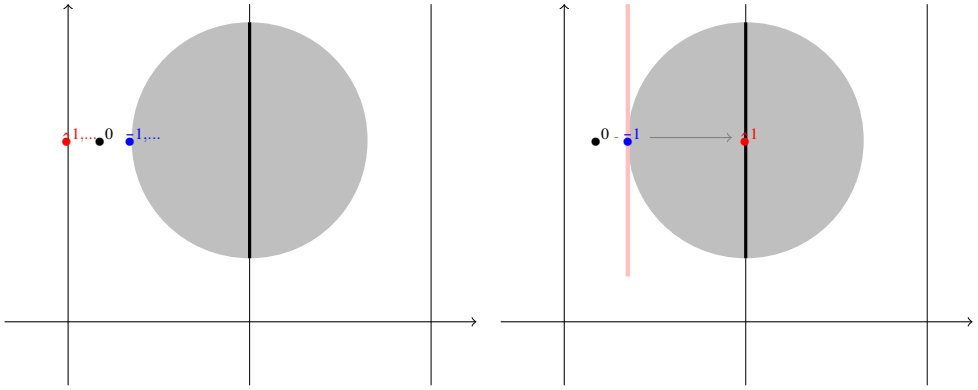

Fig. 4 Solution of (9) without (left) and with (right) OA cuts: helping to prevent cycling.

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 Section 5.1.

Example 2 In Figure 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 Figure 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

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

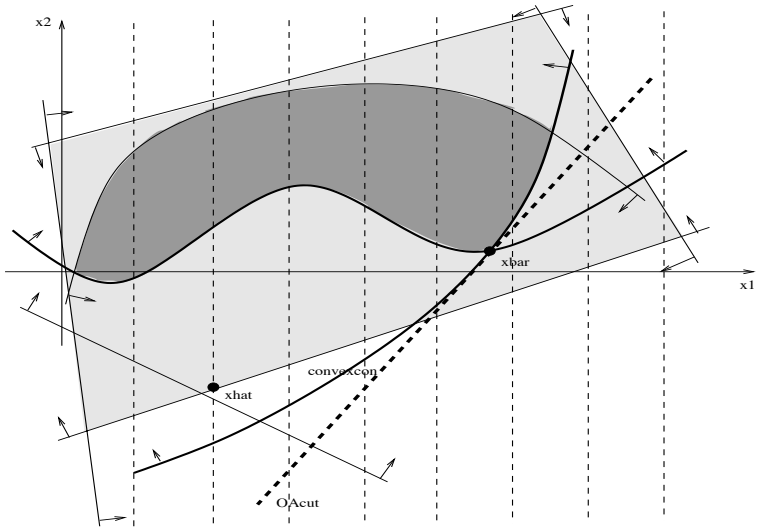

Fig. 5 The OA cut from γ does not cut off $\hat{x}$.

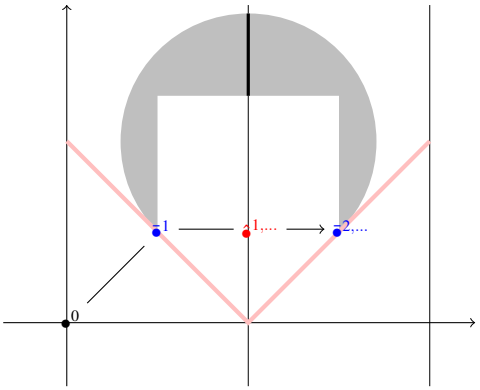

Fig. 6 OA cuts may not prevent the FP from cycling.

variants) any of the minimum distance problems is solved (even if 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

$$||x-\hat{x}^i||_A \geq \varepsilon \tag{21}$$

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

$$\mathscr{A} \cap \mathscr{B} = (\mathscr{A} \cap \{ (x,y) \ : \ (21) \}) \cap \mathscr{B} = \mathscr{A} \cap (\mathscr{B} \cap \{ (x,y) \ : \ (21) \}) \ .$$

In other words, the *nonlinear and nonconvex "cut"* (21) 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* (21), and therefore (21) provides–at least in theory–a convenient globalization mechanism.

No-good cuts [13] were originally introduced in [4] with the name of "canonical" cuts and recently used within the context of MINLP [13,26]. If x are binary variables and $\|\cdot\|$ is the L_1 norm, we can take $\varepsilon = 1$ and reformulate (21) linearly as

$$\sum_{\substack{j\le p\\ \hat{x}_j=0}} x_j + \sum_{\substack{j\le p\\ \hat{x}_j=1}} (1-x_j) \ge 1. \tag{22}$$

For general integer variables, an exact linear reformulation is given, for example, in [13] and involves adding $2p$ new continuous variables, p new binary variables and adding $3p+1$ linear equations to the problem.

It is not difficult to see that the size of such a reformulation is unfortunately excessive 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 $||x-\hat{x}^{i-1}||_2^2 \ge \varepsilon$ 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 a3 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 (2. above),
II. how to define the feasible region of (9) and solve it: MINLP (a1 above) versus MIQP (a2 above) versus rounding (a3 above) or MILP (b above) , and
III. how to avoid cycling: tabu list (i. 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 [17]. 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 [22]), of which we exploit the following features.

- Model analysis: getting information about nonlinearity and convexity of the constraints and integrality requirements of the variables, necessary 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 [21] to determine whether it is an affine combination of convex functions. We call such a function "evidently convex" [22]. 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 pretty 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 [**?**] and used among others in [24, 12]. Only 65 among such 243 instances are those in which the open-source Global Optimization solver COUENNE 0.1 [5] (available from COIN-OR [**?**]) is *unable* to find a feasible solution within a time limit of 5 minutes 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 minutes of CPU time. This is because FP can clearly be used as a stand-alone heuristic algorithm for nonconvex MINLP. On the other hand, we also wanted to be effective, in longer computing times, on the 65 "hard" instances, thus suggesting a fruitful integration of FP within COUENNE or any other GO solver (as happened for FP algorithms in MILP).

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

[3] No detailed computational results are reported for the preliminary computational investigation.

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 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 [11], while the MILP and MIQP solvers are those of CPLEX 11 [20]. 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 [13] 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 [15]: 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.

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 hours 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 *four* 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 Section 5.2. More precisely, the four variants have the characteristics reported in Table 1.

Table 1 FP variants.

	Problem (8)			Problem (9)		
variant	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

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-hours 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).

Table 2 Comparing the four FP variants, aggregated results.

	FP-1	FP-2	FP-3	FP-4
successes	49	45	22	23
successes alone	6	3	1	0
time limit reached	11	14	42	32
fails	5	6	1	9
wins	27	20	10	4
time geomean	151.02	104.45	17.59	76.14

The detailed results are reported in Table 3 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 and 3 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. Finally, 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.

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 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.

Because evil 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 feasibility pump 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. Four 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".

References

1. T. Achterberg and T. Berthold. Improving the feasibility pump. *Discrete Optimization*, 4:77–86, 2007.
2. F.A. Al-Khayyal and H.D. Sherali. On finitely terminating branch-and-bound algorithms for some global optimization problems. *SIAM Journal of Optimization*, 10(4):1049–1057, 2000.

3. H. Attouch, J. Bolte, P. Redont, and A. Soubeyran. Alternating minimization and projection methods for nonconvex problems. 0801.1780v2[math.oc], arXiv, 17 Jan 2008.
4. E. Balas and R. Jeroslow. Canonical cuts on the unit hypercube. *SIAM Journal on Applied Mathematics*, 23(1):61–69, 1972.
5. P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter. Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods and Software*, 24(4):597–634, 2009.
6. T. Berthold and A. Gleixner. Undercover — primal minlp heuristic. In P. Bonami, L. Liberti, A. Miller, and A. Sartenaer, editors, *Proceedings of the European Workshop on Mixed-Integer Nonlinear Programming*, pages 103–113, Marseille, 2010. Université de la Méditerranée.
7. P. Bonami, G. Cornuéjols, A. Lodi, and F. Margot. A feasibility pump for mixed integer nonlinear programs. *Mathematical Programming*, 119(2), 2009.
8. P. Bonami and J. Gonçalves. Primal heuristics for mixed integer nonlinear programs. Technical Report RC24639, IBM, 2008.
9. P. Bonami and J. Lee. `BONMIN` user's manual. Technical report, IBM Corporation, June 2007.
10. M. Bruglieri and L. Liberti. Optimal running and planning of a biomass-based energy production process. *Energy Policy*, 36:2430–2438, 2008.
11. COIN-OR. *Introduction to IPOPT: A tutorial for downloading, installing, and using IPOPT*, 2006.
12. C. D'Ambrosio, A. Frangioni, L. Liberti, and A. Lodi. Experiments with a feasibility pump approach for nonconvex MINLPs. In P. Festa, editor, *Symposium on Experimental Algorithms*, volume 6049 of *LNCS*, Heidelberg, 2010. Springer.
13. C. D'Ambrosio, A. Frangioni, L. Liberti, and A. Lodi. On interval subgradient and no-good cuts. *Operations Research Letters*, accepted.
14. M. Duran and I. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36:307–339, 1986.
15. M. Fischetti, F. Glover, and A. Lodi. The feasibility pump. *Mathematical Programming*, 104(1):91–104, 2005.
16. R. Fletcher and S. Leyffer. Solving mixed integer nonlinear programs by outer approximation. *Mathematical Programming*, 66:327–349, 1994.
17. R. Fourer, D. Gay, and B. Kernighan. *AMPL: A Modeling Language for Mathematical Programming, 2nd edn.* Duxbury Press/Brooks/Cole Publishing Co., Florence, KY, USA, 2003.
18. L. Grippo and M. Sciandrone. On the convergence of the block nonlinear Gauss-Seidel method under convex constraints. *Operations Research Letters*, 26:127–136, 2000.
19. P. Hansen and N. Mladenović. Variable neighbourhood search: Principles and applications. *European Journal of Operations Research*, 130:449–467, 2001.
20. ILOG. *ILOG CPLEX 11.0 User's Manual*. ILOG S.A., Gentilly, France, 2008.
21. L. Liberti. Writing global optimization software. In L. Liberti and N. Maculan, editors, *Global Optimization: from Theory to Implementation*, page 211262. Springer, Berlin, 2006.
22. L. Liberti, S. Cafieri, and F. Tarissan. Reformulations in mathematical programming: A computational approach. In A. Abraham, A.-E. Hassanien, P. Siarry, and A. Engelbrecht, editors, *Foundations of Computational Intelligence Vol. 3*, number 203 in Studies in Computational Intelligence, pages 153–234. Springer, Berlin, 2009.
23. L. Liberti and M. Dražic. Variable neighbourhood search for the global optimization of constrained NLPs. In *Proceedings of GO Workshop, Almeria, Spain*, 2005.
24. L. Liberti, N. Mladenović, and G. Nannicini. A good recipe for solving MINLPs. In V. Maniezzo, T. Stützle, and S. Voß, editors, *Hybridizing metaheuristics and mathematical programming*, volume 10 of *Annals of Information Systems*, pages 231–244, New York, 2009. Springer.
25. G. Nannicini and P. Belotti. Local branching for minlps. Technical Report Working paper, CMU, 2009.
26. G. Nannicini and P. Belotti. Rounding-based heuristics for nonconvex minlps with binary variables. Technical report, Working paper, 2009.
27. F. Schoen. Two-phase methods for global optimization. In P.M. Pardalos and H.E. Romeijn, editors, *Handbook of Global Optimization*, volume 2, pages 151–177. Kluwer Academic Publishers, Dordrecht, 2002.
28. P. Tseng. Convergence of a Block Coordinate Descent Method for Nondifferentiable Minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494, 2001.
29. H. Tuy. *Convex Analysis and Global Optimization*. Kluwer Academic Publishers, Dordrecht, 1998.
30. S.A. Vavasis. *Nonlinear Optimization: Complexity Issues*. Oxford University Press, Oxford, 1991.

Table 3 Comparing the four FP variants, detailed results.

	FP-1			FP-2			FP-3			FP-4		
instance	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
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
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