Comput Optim Appl (2012) 51:729–747
DOI 10.1007/s10589-010-9350-6

Heuristics for convex mixed integer nonlinear programs

Pierre Bonami · João P.M. Gonçalves

Received: 15 September 2008 / Published online: 14 September 2010

Abstract In this paper, we describe the implementation of some heuristics for convex mixed integer nonlinear programs. The work focuses on three families of heuristics that have been successfully used for mixed integer linear programs: diving heuristics, the Feasibility Pump, and Relaxation Induced Neighborhood Search (RINS). We show how these heuristics can be adapted in the context of mixed integer nonlinear programming. We present results from computational experiments on a set of instances that show how the heuristics implemented help finding feasible solutions faster than the traditional branch-and-bound algorithm and how they help in reducing the total solution time of the branch-and-bound algorithm.

Keywords Mixed integer nonlinear programming · Heuristics

1 Introduction

Finding good primal feasible solutions quickly is a key ingredient in state of the art software for Mixed Integer Linear Programming (MILP). It is a NP-hard problem

Supported by ANR grant BLAN06-1-138894.

Electronic supplementary material The online version of this article (doi:10.1007/s10589-010-9350-6) contains supplementary material, which is available to authorized users.

P. Bonami
Laboratoire d'Informatique Fondamentale de Marseille, CNRS-Aix-Marseille Universités, Marseille, France
e-mail: pierre.bonami@lif.univ-mrs.fr

J.P.M. Gonçalves (✉)
IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA
e-mail: jpgoncal@us.ibm.com

which can be very difficult in practice. Nevertheless, heuristic methods can be used to try to find solutions. Such heuristics can be very useful in two ways. On the one hand, they provide solutions that may be used in cases where finding an optimal solution may take too long. On the other hand, heuristics may help an exact MILP algorithm such as branch-and-bound find the optimal solution faster by providing a cutoff value which can be used to prune nodes or fix variables.

In recent years, the quest for better heuristic methods has been a very active area of research in MILP [3, 8, 10–12]. Several of those can be seen as local search heuristics where a sub-MILP is solved to search a certain neighborhood of a solution. This is the case of Local Branching [11], Relaxation Induced Neighborhood Search (RINS) [8], and Relaxation Enforced Neighborhood Search (RENS) [3]. Another heuristic recently proposed and that has been successfully applied to find solutions for difficult MILP instances is called Feasibility Pump [10, 12]. These heuristics have been successful in the sense that they have been incorporated in most state of the art commercial and open-source MILP solvers and experimentally helped solve numerous problems.

Mixed Integer Nonlinear Programming (MINLP) has also been a very active area of research in recent years. In particular, several exact solution algorithms based on branch-and-bound algorithms have been proposed for problems which feature a convex continuous relaxation [2, 5, 13–15]. Among these efforts is ongoing development of the open source software Bonmin [5, 7] in COIN-OR,[1] which was started during a collaborative effort between researchers at Carnegie Mellon University and IBM. The algorithms developed in Bonmin (and in the other works cited above) being branch-and-bounds somewhat similar to those used to solve MILPs, it seems natural that they would get the same benefits from efficient heuristics methods. But heuristics have not yet received the same level of attention in the context of MINLP. Three recent works can be cited. Bonami et al. [6] developed a version of the Feasibility Pump for MINLP particularly suited for algorithms based on Outer Approximation [9]. Abhishek et al. [1] studied several variants of the Feasibility Pump and their integration in the solver FilMINT [2]. Very recently, Berthold and Gleixner [4] proposed an heuristic that they named Undercover and that is based on constructing a mixed integer linear subproblem, by fixing a subset of the variables in the original MINLP, and solving it with a MILP solver.

In this work, we have implemented several heuristics for finding feasible solutions for MINLPs which we have incorporated in Bonmin. The heuristics that we have implemented are extensions or adaptations of well known heuristics for finding feasible solutions for mixed integer linear programs (MILPs). Given the success that those heuristics have enjoyed, namely by being incorporated in commercial and open-source MILP solvers, it seemed natural to try to apply the same ideas in the context of MINLP.

[1] www.coin-or.org.

This paper deals with the problem of finding a feasible solution to a MINLP represented as follows

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to} \quad & g(x) \leq 0 \\ & x^L \leq x \leq x^U \\ & x \in \mathbb{R}^n, \quad x_j \in \mathbb{Z}, \ \forall j \in I \end{aligned} \tag{1}$$

where $f : \{x \in \mathbb{R}^n : x^L \leq x \leq x^U\} \to \mathbb{R}$, $g : \{x \in \mathbb{R}^n : x^L \leq x \leq x^U\} \to \mathbb{R}^m$ and $I \subseteq N = \{1, \ldots, n\}$. The functions f and g are assumed to be twice continuously differentiable and convex. Although all the heuristics we present apply to (1). It is worth pointing out that in all the problems in our test set (and in most application we know of) the integer variables are actually binary. In that respect, although all the heuristics we present apply to (1) our computational findings are limited to the 0-1 case. Our work focuses on efficient implementations of diving heuristics, the Feasibility Pump, and RINS to find feasible solutions for (1). We have implemented two types of diving heuristics. The first type has the elements of a typical diving heuristic and is based on the work of Berthold [3]. The second type combines a phase of diving with the solution of a MILP and is a new kind of heuristic specifically developed for MINLP. Our implementation of the Feasibility Pump is based on the original version proposed by Fischetti et al. [10] for MILPs. We study four variants of it based on different objective functions and rounding schemes. Finally, we have implemented the RINS heuristic following the ideas introduced by Danna et al. [8].

The rest of the paper is organized as follows. In Sect. 2, we describe the heuristics that we have implemented and highlight the adjustments needed to make them successful in the context of MINLP. In Sect. 3, we describe our computational experiments and discuss the results obtained. Finally, we present conclusions and discuss the main contributions of this paper in Sect. 4.

2 Heuristics

In this section, we present the heuristics that we have implemented. As mentioned above, the paper concentrates on the diving heuristics, the Feasibility Pump, and RINS.

2.1 Diving heuristics

The basic idea of the diving heuristics is to explore a possible path from the root node to a leaf node of the branch-and-bound tree and hope that this will lead to a feasible solution. In the first type of diving heuristic that we have implemented, the process starts with the solution of the nonlinear program (NLP) obtained by relaxing the integer requirements of variables x_j, $j \in I$, in the original MINLP (1) (this nonlinear program is known as NLP relaxation). Let $\bar{x}$ be the solution of the NLP relaxation. If all $\bar{x}_j$, $j \in I$, are integer, then we have found a solution to the original MINLP

problem. Otherwise, we select a $j \in I$ such that $\bar{x}_j$ is fractional and either bound up the variable x_j by setting $x_j^L = \lceil \bar{x}_j \rceil$ or bound down by setting $x_j^U = \lfloor \bar{x}_j \rfloor$. We then solve the modified NLP and iterate the process.

The process of solving the NLP relaxation and bounding a fractional variable repeats until one of the following occurs:

1. The solution $\bar{x}$ of the NLP relaxation is feasible for the MINLP.
2. The NLP relaxation becomes infeasible.
3. The objective function value of the optimal solution of the NLP relaxation is greater or equal than the objective function value of the best integer solution known.
4. Some termination criterion such as an iteration limit is reached.

If 1 occurs, the heuristic returns the feasible solution found. If any of 2–4 occur, the heuristic stops and does not return any feasible solution.

As described above, at each iteration one has to select a variable $x_j,\ j \in I$, such that $\bar{x}_j$ is fractional and then bound that variable. We have adapted two ideas for selecting a fractional variable described in [3].

The first idea is called Fractional Diving and consists of selecting the variable $x_j,\ j \in I$, with smallest value of $|\bar{x}_j - [\bar{x}_j]|$ (where $[\cdot]$ represents scalar rounding to the nearest integer) and bounding it by the nearest integer.

The second idea is called Vectorlength Diving and consists of selecting the fractional variable $x_j,\ j \in I$, with the smallest ratio

$$\begin{cases} \dfrac{(\lceil \bar{x}_j \rceil - \bar{x}_j)\frac{\partial f(\bar{x})}{\partial x_j} + 10^{-6}}{A_j + 1}, & \text{if } \dfrac{\partial f(\bar{x})}{\partial x_j} \geq 0 \\ \dfrac{(\lfloor \bar{x}_j \rfloor - \bar{x}_j)\frac{\partial f(\bar{x})}{\partial x_j} + 10^{-6}}{A_j + 1}, & \text{otherwise} \end{cases}$$

where A_j is the number of functions $g_i(x)$ for which the variable x_j exists, i.e., x_j does not have a zero coefficient. The variable selected is rounded up if $\frac{\partial f(\bar{x})}{\partial x_j} \geq 0$ and rounded down otherwise. In the case of MILP, $\frac{\partial f(\bar{x})}{\partial x_j}$ corresponds to the value of the objective function coefficient for variable x_j. In that case, the ratio above corresponds to the change in the objective function value per row that is affected by the rounding. The selection of a small ratio favors a rounding that will produce a small change in the objective function value and a large number of rows affected by the rounding. One hopes that the more rows affected by the rounding the larger the number of variables that will be fixed at their bounds in future solutions of the NLP relaxation. The implementation for the MINLP case uses the local change in the objective function value as an estimate of the change in the objective function value due to the corresponding rounding.

A major difference between MINLP and the usual setting for MILP is the computational cost for resolving the continuous relaxation. In our case, we use an interior point method to solve the NLPs whereas in MILP one usually uses the dual simplex method. A drawback is that our interior point method does not resolve problems after bounds modifications as fast as the dual simplex method does. Therefore, fixing

variables one at a time and resolving the NLP (which is often done in MILP as for instance in [3]) would have a too great computational cost. In our implementation, we fix at each iteration up to K variables $x_j,\ j \in I$, for which $\bar{x}_j = x_j^L$ or $\bar{x}_j = x_j^U$, where $K = \lfloor pvf \times \#\text{integer variables} \rfloor$ and pvf $(0 \leq pvf \leq 1)$ is a user parameter. If the NLP relaxation is infeasible then we remove the variables fixings and resolve the NLP relaxation. If the new NLP relaxation is also infeasible, then we take the fractional variable selected for bounding and bound it to the other side. After this change, we resolve the NLP relaxation and quit the heuristic without having found a solution if that NLP relaxation is infeasible. The diving heuristic is described in pseudocode form in Fig. 1.

The second type of diving heuristic that we have implemented combines the diving strategies described above with the solution of a mixed integer linear program. The strategy employed by traditional diving heuristics may be seen as trying to fix all integer constrained variables in order to obtain a continuous subproblem directly solvable by an NLP solver (or LP solver in the traditional MILP case). In this second type of heuristic, we follow a different strategy: we try to fix variables which appear in a nonlinear term in order to build a MILP subproblem which can be directly treated by a black box MILP solver (typically much more efficient than any MINLP solver). Note that the recent work of Berthold and Gleixner [4] uses a similar idea although the construction of the MILP subproblem is done in a different way.

We start by solving the NLP relaxation of the original MINLP (1) obtained by relaxing the integer requirements of variables $x_j,\ j \in I$. The solution $\bar{x}$ of the NLP relaxation is the solution of the original MINLP if all $\bar{x}_j,\ j \in I$, are integer. If any one of the $\bar{x}_j,\ j \in I$, is fractional, then we let I_L and I_{NL} be subsets of I such that the variables $x_j,\ j \in I_L$, appear only in linear terms of the objective function and constraints and the variables $x_j,\ j \in I_{NL}$, appear in at least one nonlinear term of the objective function or constraints. Our first goal is to fix all variables in I_{NL} while keeping the ones in I_L free. Let $\bar{I}_{NL}$ be the subset of I_{NL} such that the variables $\bar{x}_j,\ j \in I_{NL}$, are fractional. If $\bar{I}_{NL} \neq \emptyset$, we select a $j \in \bar{I}_{NL}$ and either bound up the variable x_j by setting $x_j^L = \lceil \bar{x}_j \rceil$ or bound down by setting $x_j^U = \lfloor \bar{x}_j \rfloor$. We then repeat the process of solving the NLP relaxation and bounding a variable $x_j,\ j \in \bar{I}_{NL}$, until $\bar{I}_{NL}$ is empty or one of the four conditions described above for the first type of diving heuristics is verified. The selection of the variable $x_j,\ j \in \bar{I}_{NL}$, that is going to be bounded is done according to the same two criteria as above: Fractional Diving and Vectorlength Diving. Note that, in the case of Vectorlength Diving, we still use for A_j the number of functions $g_i(x)$ for which the variable x_j exists even if x_j appears in linear terms in some of those functions.

If the diving process stops because all $\bar{x}_j,\ j \in I_{NL}$, are integer and there is at least one variable $x_j,\ j \in I_L$, for which $\bar{x}_j$ is fractional, we construct the sub-MILP by taking the original MINLP (1) and substituting every variable $x_j,\ j \in N$, that appears in at least one nonlinear term by the current value $\bar{x}_j$. This problem can then be solved by using an MILP solver (in our case Cbc[2]). If the problem is infeasible, we quit the heuristic without finding a feasible solution of the original MINLP. Otherwise, any feasible solution of the MILP is a feasible solution of the original MINLP. In our

[2] https://projects.coin-or.org/cbc.

$z^* := +\infty$ or $z^* := f(x^*)$, where x^* is a known feasible solution to (1);
$\bar{x} :=$ solution of the NLP relaxation at the current node;
$\bar{I} := \{j \in I : \bar{x}_j \text{ is fractional}\}$;
$iter := 1$; $K := \lfloor pvf \times |I| \rfloor$, $0 \leq pvf \leq 1$;
while $\bar{I} \neq \varnothing$ and $iter \leq maxIter$ do
    Select $j \in \bar{I}$;
    $\bar{x}_j^L := x_j^L$; $\bar{x}_j^U := x_j^U$;
    $x_j^L := \lceil \bar{x}_j \rceil$ or $x_j^U := \lfloor \bar{x}_j \rfloor$;
    $i := 0$; $nFixed := 0$; $S := \varnothing$; $\tilde{I} := I \setminus \bar{I}$; $nNoFrac := |\tilde{I}|$;
    while $nFixed < K$ and $i < nNoFrac$ do
        Select $k \in \tilde{I}$;
        $\tilde{I} := \tilde{I} \setminus \{k\}$;
        if $\bar{x}_k = x_k^L$
          then $\bar{x}_k^L := x_k^L$; $\bar{x}_k^U := x_k^U$;
               $x_k^U := x_k^L$; $S := S \cup \{k\}$; $nFixed := nFixed + 1$; fi
        if $\bar{x}_k = x_k^U$
          then $\bar{x}_k^L := x_k^L$; $\bar{x}_k^U := x_k^U$;
               $x_k^L := x_k^U$; $S := S \cup \{k\}$; $nFixed := nFixed + 1$; fi
        $i := i + 1$;
    od
    $i := 0$;
    while $i < 3$ do
        Solve NLP relaxation;
        if NLP relaxation infeasible and $i = 0$
          then
            while $S \neq \varnothing$ do
                Select $k \in S$; $S := S \setminus \{k\}$;
                $x_k^L := \bar{x}_k^L$; $x_k^U := \bar{x}_k^U$;
            od
        fi
        if NLP relaxation infeasible and $i = 1$
          then
            if $x_j^L = \lceil \bar{x}_j \rceil$ then $x_j^L := \bar{x}_j^L$; $x_j^U := \lfloor \bar{x}_j \rfloor$;
                          else $x_j^L := \lceil \bar{x}_j \rceil$; $x_j^U := \bar{x}_j^U$; fi
        fi
        if NLP relaxation infeasible and $i = 2$
          then Stop - No solution found; fi
        if NLP relaxation feasible
          then $\bar{x} :=$ solution of the NLP relaxation;
            if $f(\bar{x}) \geq z^*$
              then Stop - Solution can't be better than x^*; fi
            $i := 2$;
        fi
        $i := i + 1$;
    od
    $iter := iter + 1$;
od

Fig. 1 Diving heuristic

```
z* := +∞ or z* := f(x*), where x* is a known feasible solution to (1);
x̄ := solution of the NLP relaxation at the current node;
I_L := {j ∈ I : x_j appears only in linear terms};
I_NL := {j ∈ I : x_j appears in at least one nonlinear term};
Ī_NL := {j ∈ I_NL : x̄_j is fractional};
iter := 1;
while Ī_NL ≠ ∅ and iter ≤ maxIter do
      Select j ∈ Ī_NL;
      x_j^L := ⌈x̄_j⌉ or x_j^U := ⌊x̄_j⌋;
      Solve NLP relaxation;
      if NLP relaxation infeasible
        then Stop - No solution found;
        else x̄ := solution of the NLP relaxation;
             if f(x̄) ≥ z*
               then Stop - Solution can't be better than x*;
             fi
      fi
      iter := iter + 1;
od
Ī_L := {j ∈ I_L : x̄_j is fractional};
N_NL := {j ∈ N : x_j appears in at least one nonlinear term};
if Ī_L ≠ ∅ and Ī_NL = ∅
  then while N_NL ≠ ∅ do
             Select j ∈ N_NL; N_NL := N_NL \ {j};
             x_j^L := x̄_j; x_j^U := x̄_j;
       od
       Solve MILP;
       if MILP infeasible
         then Stop - No solution found;
         else return solution of the MILP;
       fi
fi
```

Fig. 2 Dive + MILP heuristic

implementation, we search for the optimal solution of the MILP (i.e. we don't set any early termination limit). The pseudocode of this heuristic is given in Fig. 2.

2.2 Feasibility pump

Our implementation of the Feasibility Pump heuristic is based on the original version proposed for MILP by Fischetti et al. [10]. This is a computationally less intensive version than the one proposed by Bonami et al. [6] for MINLP.

The Feasibility Pump heuristic starts from any point $x^* \in F = \{x : g(x) \leq 0, x^L \leqq x \leq x^U\}$. We define the rounding $\tilde{x}$ of x^* by setting $\tilde{x}_j := [x_j^*]$ if $j \in I$ and $\tilde{x}_j := x_j^*$ otherwise, where $[\cdot]$ represents scalar rounding to the nearest integer. Note that, although $\tilde{x}$ satisfies the integer requirements in (1), it will not typically sat-

isfy the other constraints in (1). Each iteration of the Feasibility Pump heuristic consists of finding a point $x^* \in F$ that is as close as possible to $\tilde{x}$ according to a certain distance function $\Delta(x, \tilde{x})$. This corresponds to solving the following NLP problem

$$\begin{aligned} \min \quad & \Delta(x, \tilde{x}) \\ \text{subject to} \quad & g(x) \leq 0 \\ & x^L \leq x \leq x^U \\ & x \in \mathbb{R}^n \end{aligned} \tag{2}$$

If $\Delta(x^*, \tilde{x}) = 0$, then x^* is a feasible solution to the original MINLP (1) and the heuristic returns that solution. Otherwise, we replace $\tilde{x}$ by the rounding of x^* and repeat.

The distance function proposed by Fischetti et al. [10] is the L_1-norm defined as

$$\Delta(x, \tilde{x}) = \sum_{j \in I} |x_j - \tilde{x}_j|$$

which for the case where all variables $x_j,\ j \in I$, are binary can be written as

$$\Delta(x, \tilde{x}) = \sum_{j \in I\,:\,\tilde{x}_j = 0} x_j + \sum_{j \in I\,:\,\tilde{x}_j = 1} (1 - x_j) \tag{3}$$

Our implementation deals with the case where all variables $x_j,\ j \in I$, are binary and thus we use (3). We have also implemented the case where the distance function is the L_2-norm as proposed by Bonami et al. [6].

As pointed out by Fischetti et al. [10], the Feasibility Pump heuristic can stall or cycle. If the solution x^* of (2) is not integer (i.e., not all $x^*_j,\ j \in I$, are integer), then we round x^*. However, if $[x^*_j] = \tilde{x}_j,\ \forall j \in I$, the heuristic stalls. In this case, Fischetti et al. [10] flip a random number of $TT \in \{\frac{1}{2}T, \ldots, \frac{3}{2}T\}$ of the components $\tilde{x}_j,\ j \in I$, chosen such that the increase in the total distance $\Delta(x^*, \tilde{x})$ is minimized (i.e., flip the TT entries $\tilde{x}_j,\ j \in I$, with highest $|x^*_j - \tilde{x}_j|$). In their computational experiments, Fischetti et al. set T to 20. Experiments with our implementation of the Feasibility Pump for MINLP revealed that flipping only the entry $\tilde{x}_j,\ j \in I$ with highest value of $|x^*_j - \tilde{x}_j|$ produced the best performance.

Cycling occurs when the heuristic enters a loop where the same sequence of points x^* and $\tilde{x}$ is repeatedly visited. We have implemented the mechanism proposed by Fischetti et al. [10] to detect and overcome this problem. A cycle is detected by comparing the current solution with those found in the last three iterations. As soon as a cycle is detected, the rounding step in the heuristic is substituted by a random perturbation move that consists of taking each $j \in I$ and flipping $\tilde{x}_j$ in case $|x^*_j - \tilde{x}_j| + \max\{\rho_j, 0\} > 0.5$, where $\rho_j \in [-0.3, 0.7]$ is a uniformly distributed random value.

Let $x^* \in F = \{x : g(x) \leq 0,\ x^L \leq x \leq x^U\}$;
if x^* is feasible for (1) then return x^* fi
$\tilde{x} :=$ rounding of x^*(i.e., $\tilde{x}_j := [x_j^*]$ if $j \in I$ and $\tilde{x}_j := x_j^*$ otherwise);
$iter := 1$;
while $iter \leq maxIter$ do
    $x^* :=$ solution of NLP problem (2);
    if x^* is feasible for (1) then return x^*; fi
    if $\exists j \in I : [x_j^*] \neq \tilde{x}_j$
      then $\tilde{x} :=$ rounding of x^*(i.e., $\tilde{x}_j := [x_j^*]$ if $j \in I$ and $\tilde{x}_j := x_j^*$ otherwise);
       else Flip $\tilde{x}_j,\ j \in I,$ with highest value of $|x_j^* - \tilde{x}_j|$;
    fi
    if Cycling detected
      then for all $j \in I$ do
            Generate uniformly random value $\rho_j \in [-0.3, 0.7]$;
            if $|x_j^* - \tilde{x}_j| + \max\{\rho_j, 0\} > 0.5$ then Flip $\tilde{x}_j$; fi
          od
    fi
    $iter := iter + 1$
od

Fig. 3 Feasibility pump

If the original MINLP (1) contains constraints of the form

$$\sum_{j=1}^{k} x_j = 1$$

where $x_j \in \{0, 1\},\ j = 1, \ldots, k$, then when we round x^* we can take this into account. Specifically, only one variable $x_j,\ j = 1, \ldots, k$, should be rounded to one. All others should be rounded to zero. We have implemented a rounding scheme used by Abhishek et al. [1] in a version of a Feasibility Pump heuristic that does exactly that. We refer to it as GUB rounding. It consists of setting $\tilde{x}_j = 1$ for $j = [s]$ and $\tilde{x}_j = 0$ otherwise, where

$$s = \sum_{j=1}^{k} j x_j^*$$

In our implementation, the input parameter *maxIter* allows the user to limit the number of iterations that the Feasibility Pump heuristic performs. The pseudocode of the Feasibility Pump heuristic implemented is given in Fig. 3.

2.3 RINS

The RINS heuristic was proposed by Danna et al. [8] for MILP. It is an improvement heuristic that takes a known integer feasible solution for the MILP and uses local search to find an improved solution. It consists of solving a sub-MILP to search a

$z^* := f(x^*)$, where x^* is a known feasible solution to (1);
$x :=$ solution of the NLP relaxation at the current node;
$nFix := 0$;
$j := 1$;
while $j \leq n$ do
    if $x_j^* = \bar{x}_j$ then
        $x_j^L := \bar{x}_j$; $x_j^U := \bar{x}_j$;
        $nFix := nFix + 1$;
    fi
    $j := j + 1$;
od
Add to (1) the following constraint $f(x) \leq (1 - \epsilon)z^*$, $\epsilon > 0$;
if $nFix \geq |I|/10$ then Solve sub-MINLP fi

Fig. 4 RINS

neighborhood of the known solution. The neighborhood is constructed based on the observation that at a node of the branch-and-bound tree some variables of the solution of the continuous relaxation take the same values as those of the known integer feasible solution. Those variables are fixed and a sub-MILP on the remaining variables is solved. The sub-MILP contains a cutoff constraint to force the solution to be better than the current integer feasible solution known. Our implementation follows this strategy with the additional rule that we only solve the sub-MINLP in case the number of variables fixed is greater than one tenth of the total number of integer variables in the problem. The pseudocode for our implementation is given in Fig. 4.

Although RINS is a very simple heuristic, in order for it to be efficient, a key element is to be able to solve quickly the sub-MINLPs. Typically, the sub-MINLPs are simpler to solve than the original MINLP. In our experiment, using an NLP based branch-and-bound to solve it is too slow and did not give satisfying computational results. For this reason, we used the Quesada and Grossmann algorithm [15] to solve the sub-MINLP. Furthermore the Quesada and Grossmann algorithm stops after the third integer feasible solution (improving the incumbent) is found and is given a node limit of 1000 nodes.

3 Computational results

In this section, we describe our computational experiments and report the results obtained. The heuristics were implemented in Bonmin [7] version 0.99 and use Ipopt3.4[3] to solve the nonlinear programs and Cbc2.1[4] to solve the mixed integer linear programs. We tested the heuristics on 120 instances from the set of convex MINLP instances discussed in [5] representing applications from operations research

[3] https://projects.coin-or.org/ipopt.

[4] https://projects.coin-or.org/cbc.

and chemical engineering. All tests were performed on an Intel Core 2 Duo 2.4 GHz CPU laptop with 3 gigabytes of RAM and running Red Hat Enterprise Linux 5.2. In the experiments, the computing times were measured in CPU seconds.

We present three experiments. The first one is aimed at studying the performance of the heuristics implemented in terms of quickly finding a first solution. The second one is aimed at studying if the heuristics help the branch-and-bound algorithm to solve instances faster. Finally, the third experiment is aimed at determining if the heuristics can help finding better solutions to unsolved problems.

3.1 Comparison of first solution found

In our first experiment, we compare the solutions obtained with the heuristics and the first solution obtained by the branch-and-bound algorithm implemented in Bonmin. The default parameters were used in Bonmin, which includes the $dynamic$ node selection strategy. This is a strategy that tries to find good feasible solutions at the start of the tree search by selecting the least integer infeasible node as the next node to process (the strategy is followed until 5 feasible solutions are found and then switches to *best-bound*). Each of the diving heuristics (the pure diving and the diving followed by the solution of a MILP) was run with both variable selection criteria (Fractional and Vectorlength). The parameter pvf (see Fig. 1) was set to 20%. The Feasibility Pump was run using the L_1-norm and the L_2-norm in the distance function of the objective function and also with and without the GUB rounding scheme used by Abhishek et al. [1]. The parameter *maxIter* (see Fig. 1) was set to 200.

In the experiment, we also included the version of the Feasibility Pump for MINLP developed by Bonami et al. [6] which is included in Bonmin. This heuristic can return more than one integer feasible solution but we limited it to stop after the first one. Both the branch-and-bound algorithm and the heuristics were given a time limit of one hour. The RINS heuristic is not used in this experiment since it is an improvement heuristic and thus requires the input of a known integer feasible solution. The detailed results of the first experiment are given in Electronic Supplementary Material (see Tables 4 to 7). A summary of the results is given in Table 1. In that table we provide a comparison between the solutions found by each heuristic and the first solution obtained by the branch-and-bound algorithm. Each row in the table corresponds to the comparison between the heuristic listed in the first column and the branch-and-bound algorithm. The second column in the table shows the percentage of instances used in the comparison. These are the instances for which both the heuristic and the branch-and-bound algorithm found a solution. The rest of the table is divided in two blocks of three columns each where in the first block we report the results of the heuristic and in the second block we report the results of the branch-and-bound algorithm. In the first column of each block we report the geometric mean of the relative difference between the solutions found and the corresponding best solution known. The second column of each block contains the geometric mean of the times to run each instance. Finally, the third column of each block contains the percentage of instances for which the solution obtained by the heuristic was better than the solution obtained by the branch-and-bound algorithm, or vice versa.

Although not reported in the table, the branch-and-bound algorithm found an integer feasible solution for 96% of the instances. The Fractional Diving heuristic found

Table 1 Comparison of performance of diving heuristics and branch-and-bound algorithm until first solution found. Column labeled "Instances (%)" contains the percentage of instances for which each heuristic found a solution. Column labeled "Difference (%)" contains the geometric mean of the relative difference between the solutions found and the corresponding best known solution. Column labeled "Time (s)" contains the geometric mean of the times. Column labeled "Wins (%)" contains the percentage of instances where the heuristic or BB obtained a better solution

	Instances (%)	Heuristic			Branch-and-Bound		
		Difference (%)	Time (s)	Wins (%)	Difference (%)	Time (s)	Wins (%)
Dive Fractional	78	10.5	1.2	71	23.2	5.5	29
Dive Vectorlength	73	11.3	0.7	72	21.9	5.1	28
Dive Fractional + MILP	48	1.1	0.5	75	6.6	4.1	25
Dive Vectorlength + MILP	48	1.9	0.6	74	6.6	4.1	26
FP, L_1-norm	93	22.1	0.4	67	28.3	6.3	33
FP, L_2-norm	93	23.2	0.4	64	28.6	6.2	36
FP, L_1-norm, GUB rounding	94	27.3	0.3	61	28.9	6.4	39
FP, L_2-norm, GUB rounding	88	27.3	0.3	59	27.7	5.2	41
FP by Bonami et al. [6]	96	38.7	0.8	57	29.3	6.7	43

an integer feasible solution for 78% of the instances and for 71% of those instances the solution found is better than the solution found by the branch-and-bound algorithm. In addition to finding better solutions for a larger number of instances, the Fractional Diving heuristic is also, in general, significantly faster than the branch-and-bound algorithm. The Vectorlength Diving found an integer feasible solution for 73% of the instances and its performance is only slightly worse than the Fractional Diving. In terms of quality of the solutions found (when both heuristics found a solution), our experiments also give a slight advantage to Fractional Diving: the solution it found was better in 41 cases (on average by 112%) while the solution found by Vectorlength Diving was better in 32 cases (on average by 31%).

The Fractional Diving + MILP heuristic found a solution for 48% of the instances and the solution is better than the one found by the branch-and-bound algorithm for 75% of those instances. The heuristic is also, in general, faster than the branch-and-bound algorithm. Although this is the heuristic which found solutions for the least problems, it is remarkable in the sense that in almost all cases the solution found is optimal or very close to optimal. Furthermore, for most instances the heuristic is very fast either when it finds a solution or when it quits without finding a solution. The performance of the Fractional and Vectorlength Diving + MILP heuristics is very similar.

A question that immediately arises when considering the Dive + MILP heuristic results is whether the instances in the test set actually have integer variables that appear nonlinearly in the problem (for when there are not the heuristic consists just in fixing the continuous nonlinear variables at their current values and solving the resulting MILP). A detailed analysis of the results of the Fractional Diving + MILP heuristic revealed that 39% of the instances have integer variables appearing in nonlinear terms (i.e., $I_{NL} \neq \emptyset$) but only for 27% of the instances the solution of the NLP relaxation of the original MINLP contains variables $\bar{x}_j$, $j \in I_{NL}$ which are fractional

(i.e., $\bar{I}_{NL} \neq \emptyset$). For those 27% of the instances, the average size of the set $\bar{I}_{NL}$ was 4.9 and the average number of times the NLP relaxation had to be solved before fixing variables that appear in nonlinear terms and solving the resulting MILP was 1.8. Also, among those 27% of the instances the heuristic found a solution in 81% of the cases and for those cases the average size of $\bar{I}_{NL}$ was 4.

In terms of the number of solutions obtained, both versions of the Feasibility Pump with the regular rounding and the version with L_1-norm and GUB rounding have an excellent performance finding a solution for 93–94% of the instances. Moreover, they are, in general, faster than the branch-and-bound algorithm. In addition, the solution quality is better for 61% of the instances using the version with L_1-norm and GUB rounding up to 67% of the instances using the version with L_1-norm and regular rounding. Note that the GUB rounding can only be useful for 58% of the instances since the remaining instances do not contain the constraints used in that type of rounding.

In all our versions of the Feasibility Pump, stalling and cycling did not occur for the majority of the instances in this test set. For example, for the version with L_1-norm and GUB rounding stalling and cycling occurred only for 32% and 18% of the instances, respectively. For those instances, stalling and cycling occurred on average for 68% and 24% of the iterations, respectively.

The version of the Feasibility Pump proposed by Bonami et al. [6] found a solution for 96% of the instances (the same number as the branch-and-bound algorithm) but in terms of solution quality it performed worse than any of the other heuristics on average.

A performance profile comparing the solution quality of some of the heuristics and the branch-and-bound algorithm is given in Fig. 5. In this profile we measure the solutions quality in terms of their performance ratios (if O is the optimal value and h is the value of the solution found by the heuristic the performance ratio ρ is given by $\rho := \max\{\frac{O}{h}, \frac{h}{O}\}$). Since the performance profile with the results of all the heuristics would be too confusing we included only one heuristic of each type. The diving heuristics included are of the type Fractional Diving and the Feasibility Pump (noted FP in the legend) is the one with L_1-norm and GUB rounding. The heuristic named "FP for MINLP" is the one proposed by Bonami et al. [6]. On the left side of the performance profile it can be seen that the Diving + MILP heuristic finds the best known solution for more than 40% of the instances, which is a better performance than any other heuristic or the branch-and-bound algorithm. It can also be seen more towards the right side of the plot that the quality of the solution found by our implementation of the Feasibility Pump is quite good.

Figure 6 shows a performance profile comparing the time to get a solution by both the heuristics and the branch-and-bound algorithm. The heuristics used in the previous performance profile are also used in this one. Our version of the Feasibility Pump has the best performance regarding the time to get a solution.

3.2 Comparison of complete solutions

The objective of our second experiment was to study how each of the heuristics may help the branch-and-bound algorithm find the optimal solution of a MINLP faster. We

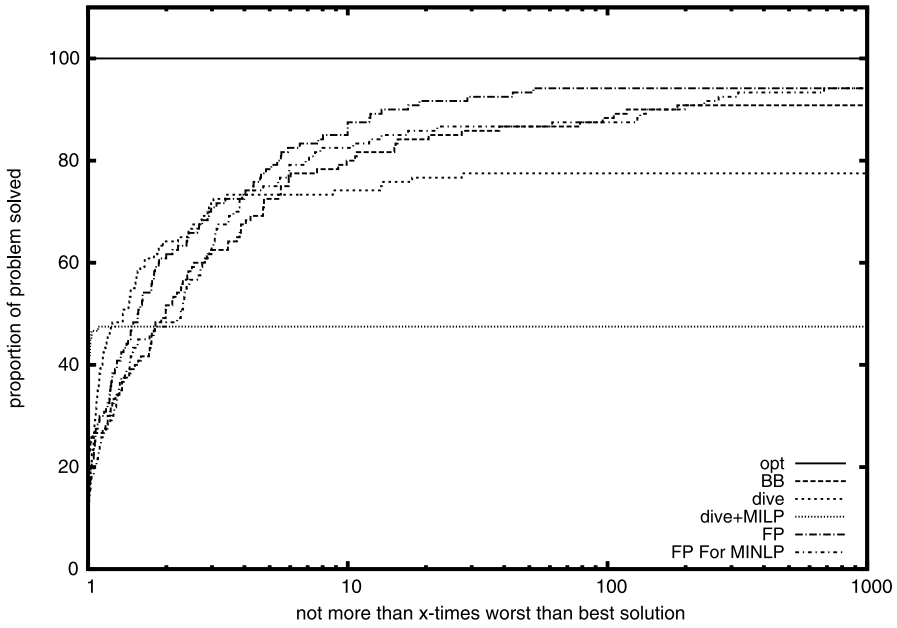

Fig. 5 Performance Profile comparing solution quality

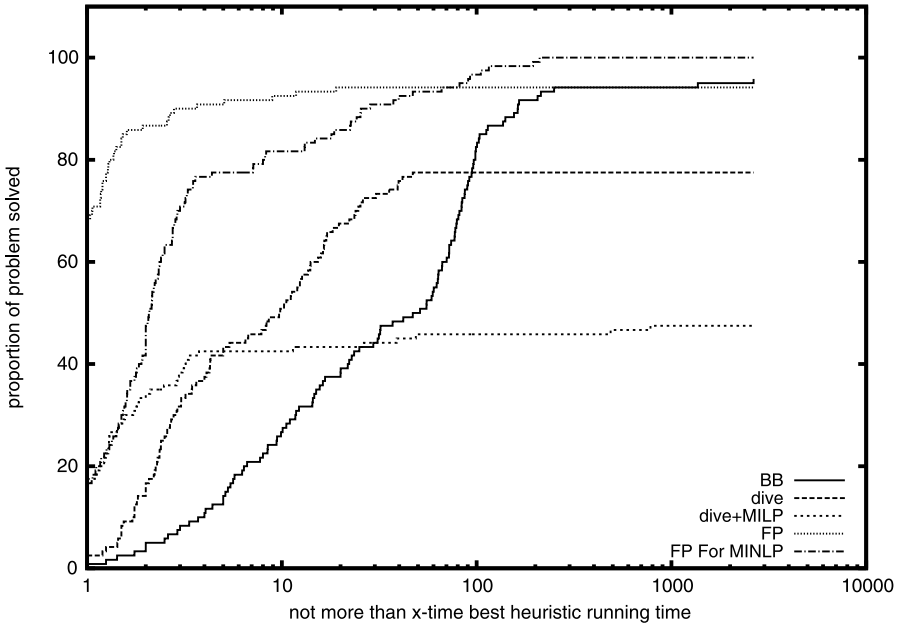

Fig. 6 Performance Profile comparing time to find first solution

Table 2 Comparison of performance of branch-and-bound algorithm with and without heuristics

	Iterations total	Nodes		Time	
		Total	Geometric mean	Total	Geometric mean
No heuristic	10460772	426676	157.9	15742.3	14.1
Dive Fractional	9926752	424646	121.3	15745.6	13.6
Dive Fractinal + MILP	9693656	417871	96.9	14965.3	11.1
FP, L_1-norm, GUB rounding	10288376	434390	133.5	15135.3	13.7
FP by Bonami et al. [6]	11373283	414085	137.4	16491.6	16.9
Combination	9293954	408820	95.9	14023.7	11.1

divided this experiment in two parts. In the first part, we studied the 102 instances in our test set that can be solved with the branch-and-bound algorithm within one hour. In this case, we compare the results obtained by Bonmin's branch-and-bound algorithm without any heuristic and the results obtained by the same algorithm using each one of the heuristics. In all cases, the default parameters were used in Bonmin except for the node selection strategy which was set to *best-bound* since this gives the best performance of the branch-and-bound algorithm. We selected one version of each heuristic for this experiment. In the results, we did not include the RINS heuristic since in our experiments this heuristic was only helpful when solving harder unsolved instances (this concurs with the results in [8]). We chose the Fractional Diving heuristic, the Fractional Diving + MILP heuristic, the Feasibility Pump with L_1-norm and the GUB rounding scheme, and the Feasibility Pump proposed by Bonami et al. [6]. Both diving heuristics were called at the root node and at every 100th node. Both versions of the Feasibility Pump heuristic were called only at the root node. All cases were run with a time limit of one hour.

In view of the results with each heuristic, we also tested a combination of heuristics. Specifically, we tested the case where the Fractional Diving + MILP heuristic was first called at the root node. If that heuristic did not return a solution, then the Feasibility Pump with L_1-norm and the GUB rounding scheme was called. If that heuristic did not return a solution, then the Feasibility Pump proposed by Bonami et al. [6] was called.

The detailed results are given in Electronic Supplementary Material (see Tables 8 and 9). A summary of the results is given in Table 2. Each row of the table corresponds to one of the cases tested and the results presented are the total number of iterations, the total number of nodes and corresponding geometric mean, and the total time to solve the 102 instances and the geometric mean of the time. The smallest total time is obtained by the combination of heuristics and it corresponds to an improvement of 11% over the total time of the branch-and-bound algorithm without heuristics. The corresponding improvement in the geometric mean is 21%. Using the combination of heuristics provides an improvement in the total time over using only the Fractional Diving + MILP heuristic while the geometric mean of the times remains the same. The Feasibility Pump with L_1-norm and GUB rounding was the next best performer in terms of total time to solve the 102 instances with an improvement of 4% over the branch-and-bound algorithm without heuristics. Although slightly

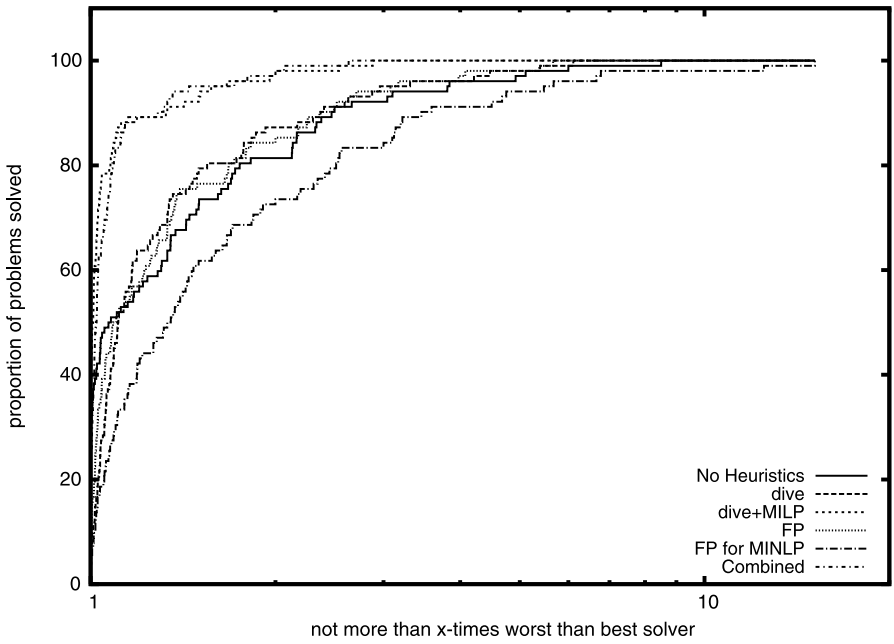

Fig. 7 Performance Profile of the solution time comparing results of the branch-and-bound algorithm with and without heuristics

better than the Feasibility Pump with L_1-norm and GUB rounding in terms of the geometric mean of the times, the diving heuristic took longer to solve all instances. Finally, using the Feasibility Pump proposed by Bonami et al. [6] reduced the number of nodes but took longer than the branch-and-bound algorithm without any heuristics. In Fig. 7, we present a performance profile of the solution time. It should be emphasized that the heuristics described in this paper are, in general, fast and that even for the instances where an heuristic does not help reduce the number of NLP iterations and the number of nodes of the branch-and-bound tree it also does not increase the time to solve the problem in a significant way.

3.3 Comparison on unsolved instances

In the third experiment, we focused on the 18 instances from our test set that were not solved by the branch-and-bound algorithm within one hour. In this case, we used the branch-and-bound algorithm without heuristics and the branch-and-bound algorithm with the combination of heuristics that we used in the first part of the experiment with the addition of the RINS heuristic. The RINS heuristic was called at every node of the branch-and-bound tree. We also used the Iterated Feasibility Pump (IFP) proposed by Bonami et at. [6] which in simple terms is an algorithm that consists on iterating the Feasibility Pump heuristic that they introduced in that paper. The time limit in this case was six hours.

Table 3 Comparison of first and last solutions obtained by the branch-and-bound algorithm with combination of heuristics and others

	BB (combination) vs. BB (no heuristic)		BB (combination) vs. IFP	
	First Int Sol	Last Int Sol	First Int Sol	Last Int Sol
Wins sol. quality and time (%)	38.9	66.7	27.8	38.9
Loses sol. quality and time (%)	0.0	11.1	38.9	16.7
Wins only sol. quality (%)	5.6	5.6	27.8	5.6
Wins only time (%)	55.6	16.7	5.6	38.9

Since none of the methods used could solve any of the 18 instances in a reasonable amount of time, we concentrate our analysis on the first and last integer feasible solutions found. The detailed results are given in Electronic Supplementary Material (see Tables 10 and 11). A summary of the results is given in Table 3. In the first two columns of results in this table we provide a comparison between the first and last integer feasible solutions found by the branch-and-bound algorithm with the combination of heuristics and the branch-and-bound algorithm without heuristics. Each number represents the percentage of instances for which the branch-and-bound algorithm with the combination of heuristics wins in terms of the solution quality and the time to get that solution, wins only in one of those categories, or loses. The last two columns contain the comparison between the branch-and-bound algorithm with the combination of heuristics and the IFP. The comparison between the two versions of branch-and-bound shows that using the combination of heuristics provides a first integer feasible solution faster in 94.5% (38.9% + 55.6%) of the instances and better in 38.9% of the instances than not using heuristics. It also shows that the last integer feasible solutions found by the version that uses heuristics are better in 66.7% of the instances than those found by the version that does not uses heuristics. The majority (55.6% = 27.8% + 27.8%) of first integer feasible solutions obtained by branch-and-bound with the combination of heuristics are better than those found by the IFP although the latter is faster for 66.7% (38.9% + 27.8%) of the instances. In terms of finding the last solution, branch-and-bound with the combination of heuristics is faster than the IFP in 77.8% (38.9% + 38.9%) of the instances. Those solutions are better in 44.5% (38.9% + 5.6%) of the instances.

Finally, it is worthwhile pointing out the usefulness of the RINS heuristic. As can be seen in Table 11 of Electronic Supplementary Material, in the runs of branch-and-bound with heuristics, the last solution was found by RINS in 66.7% of the instances. Similarly to what happens in the case of MILP, the RINS heuristic can be computationally expensive but it can also be very useful in finding solutions for hard instances.

4 Conclusions

In this paper, we described our implementation of several heuristics for convex MINLPs. We concentrated on diving heuristics, the Feasibility Pump, and RINS, all of which have been successfully used for MILP. We implemented two types of diving heuristics. The first type follows the ideas of traditional diving heuristics implemented

for MILP. The second type is a new kind of heuristic that combines a diving phase with the solution of a MILP. Our implementation of the Feasibility Pump is based on the original version proposed by Fischetti et al. [10] and includes the GUB rounding scheme used by Abhishek et al. [1]. Finally, the implementation of the RINS heuristic is based on the paper by Danna [8].

We presented computational results that show that the heuristics implemented are successful in finding feasible solutions faster than the branch-and-bound algorithm. The performance of the heuristics in terms of the number of instances for which a solution is found ranges considerably. In the lower end are the Diving + MILP heuristics which find a solution for 48% of the instances and in the higher end are three versions of the Feasibility Pump which find a solution for 93–94% of the instances. The quality of the solutions found by the heuristics is usually good and all heuristics find better solutions than the branch-and-bound algorithm for the majority of instances for which they find a solution. The Diving + MILP heuristics are noteworthy in that respect since the solutions they find are either optimal or very close to optimal.

Our experiments also show that the heuristics implemented can help accelerate the branch-and-bound algorithm. The total time to solve all 102 instances that can be solved in less than one hour can be reduced by up to 11% (with a corresponding geometric mean of 21%) using a combination of heuristics. Moreover, for the majority of the remaining 18 instances, the use of a combination of heuristics helps the branch-and-bound algorithm obtain better solutions or obtain solutions of the same quality but faster.

Acknowledgements We would like to thank Andreas Wächter for an useful discussion on this topic and for bringing reference [1] to our attention. We also thank the two anonymous referees for their insightful comments that helped improve the paper.

References

1. Abhishek, K., Leyffer, S., Linderoth, J.T.: Feasibility pump heuristics for mixed integer nonlinear programs. Unpublished Working Paper (2008)
2. Abhishek, K., Leyffer, S., Linderoth, J.T.: FilMINT: an outer approximation-based solver for convex mixed-integer nonlinear programs. INFORMS J. Comput. doi:10.1287/ijoc.1090.0373 (2010)
3. Berthold, T.: Primal heuristics for mixed integer programs. Master's thesis, Technical University of Berlin (2006)
4. Berthold, T., Gleixner, A.M.: Undercover—a primal heuristic for MINLP based on sub-MIPs generated by set covering. ZIB-Report 09-04, Konrad-Zuse-Zentrum für Informationstechnik Berlin (2009)
5. Bonami, P., Biegler, L.T., Conn, A.R., Cornuéjols, G., Grossmann, I.E., Laird, C.D., Lee, J., Lodi, A., Margot, F., Sawaya, N., Wächter, A.: An algorithmic framework for convex mixed integer nonlinear programs. Discrete Optim. **5**, 186–204 (2008)
6. Bonami, P., Cornuéjols, G., Lodi, A., Margot, F.: A feasibility pump for mixed integer nonlinear programs. Math. Program. **119**, 331–352 (2009)
7. Bonami, P., Lee, J.: Bonmin User's Manual (2007). www.coin-or.org
8. Danna, E., Rothberg, E., Pape, C.L.: Exploring relaxation induced neighborhoods to improve MIP solutions. Math. Program. **102**, 71–90 (2005)
9. Duran, M., Grossmann, I.: An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math. Program. **36**, 307–339 (1986)
10. Fischetti, M., Glover, F., Lodi, A.: The feasibility pump. Math. Program. **104**, 91–104 (2005)
11. Fischetti, M., Lodi, A.: Local branching. Math. Program. **104**, 23–47 (2003)

12. Fischetti, M., Salvagnin, D.: Feasibility pump 2.0. Math. Program. Comput. **1**, 201–222 (2009)
13. Fletcher, R., Leyffer, S.: Solving mixed integer nonlinear programs by outer approximation. Math. Program. **66**, 327–349 (1994)
14. Leyffer, S.: Integrating SQP and branch-and-bound for mixed integer nonlinear programming. Comput. Optim. Appl. **18**, 295–309 (2001)
15. Quesada, I., Grossmann, I.: An LP/NLP based branched and bound algorithm for convex MINLP optimization problems. Comput. Chem. Eng. **16**, 937–947 (1992)