J Glob Optim (2018) 71:593–612
https://doi.org/10.1007/s10898-018-0623-4

Integrality gap minimization heuristics for binary mixed integer nonlinear programming

Wendel Melo[1] · **Marcia Fampa**[2] · **Fernanda Raupp**[3]

Received: 26 January 2017 / Accepted: 12 February 2018 / Published online: 20 February 2018

Abstract We present two feasibility heuristics for binary mixed integer nonlinear programming. Called integrality gap minimization algorithm (IGMA)—versions 1 and 2, our heuristics are based on the solution of integrality gap minimization problems with a space partitioning scheme defined over the integer variables of the problem addressed. Computational results on a set of benchmark instances show that the proposed approaches present satisfactory results.

Keywords Binary mixed integer nonlinear programming · Heuristics · Integrality gap minimization · Solver

1 Introduction

Mixed integer nonlinear programming (MINLP) problems are characterized by the simultaneous presence of integer variables and nonlinear expressions in the objective function and/or constraints. In this work, we address the following binary MINLP problem:

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s10898-018-0623-4) contains supplementary material, which is available to authorized users.

✉ Wendel Melo
wendelmelo@ufu.br

Marcia Fampa
fampa@cos.ufrj.br

Fernanda Raupp
fernanda@lncc.br

1 College of Computer Science, Federal University of Uberlandia, Uberlândia, Brazil

2 Institute of Mathematics and COPPE, Federal University of Rio de Janeiro, Rio de Janeiro, Brazil

3 National Laboratory for Scientific Computing (LNCC) of the Ministry of Science, Technology and Innovation, Petrópolis, Brazil

$$
\begin{aligned}
(P)\ \min_{x,y}\ & f(x, y) \\
\text{s. t.:}\ & g(x, y)\ \le\ 0, \\
& x \in X,\ \ y \in Y \cap \{0,\ 1\}^{n_y},
\end{aligned}
\tag{1}
$$

where X and Y are polyhedral subsets of $\mathbb{R}^{n_x}$ and $\mathbb{R}^{n_y}$, respectively, $f : \mathbb{R}^{n_x+n_y} \to \mathbb{R}$ and $g : \mathbb{R}^{n_x+n_y} \to \mathbb{R}^m$ are convex and twice continuously differentiable. For the presentation of the ideas in this work, we consider the case where all functions in (P) are convex. However, the algorithms proposed can be applied to problems where nonconvexities are present. In addition, we consider that (P) has one or more optimal solutions.

Among the MINLP applications found in the literature, we can mention, for example: telecommunications network planning [1], scheduling operations in oil refineries [34], power generation systems [31], supply chain design [43], industrial robot paths [23], synthesis of chemical processes [17], prevention of aircraft conflicts [12], metabolic networks in biotechnology [26], water distribution networks [11], valve trajectory in series of tanks in cascades [25], thermal insulation layer design for the Large Hadron Collider [29] and recharge of nuclear reactors [29].

In recent years, several exact algorithms have been proposed to solve MINLP problems to optimality (the interested reader can find good bibliographic surveys in [9,14,27,41]). Many of these methods can have their performance improved if good feasible solutions are known *a priori*. For example, linear approximation algorithms (e.g. Outer Approximation [17,20]) can use these solutions to construct good linearizations of nonlinear expressions, consequently saving some iterations. Branch-and-bound algorithms can also use the upper bound provided by a feasible solution (minimization) to perform pruning, and avoid the exploration of nonpromising partitions of the feasible region. Furthermore, in many practical cases, exact algorithms are not able to find a feasible solution in reasonable time, and heuristic procedures become particularly handy. Some specific applications may also require the generation of feasible solutions in short time, which makes it impossible to adopt exact methods. Finally, some algorithms, such as *Local Branching* [19], aim at improving a given feasible solution, and, therefore, require a feasible solution as input data.

The situations described above indicate the importance of the development of procedures capable of providing feasible solutions in short computational time. These procedures are known as *feasibility heuristics*, and their relevance may be even greater in cases where the problem involves nonconvex objective function and/or nonconvex feasible region. As MINLP problems present many different characteristics, especially concerning their nonlinear terms, it may be helpful if MINLP solvers have a diverse set of heuristics to be applied in order to expand their ability to cover a vast number of applications, as results in [5] suggest.

Some heuristics proposed for Mixed Integer Linear Programming (MILP) were adapted for convex MINLP. For example, [7,8] present adaptations of Feasibility Pump for convex MINLP. Indeed, the Feasibility Pump algorithm was originally proposed for MILP in [3] and is characterized by maintaining two sequences of solutions. Solutions in the first sequence satisfy the constraints in the continuous relaxation of (P), but are not necessarily integer, while solutions on the second sequence are integer but do not necessarily satisfy the other constraints in (P). The algorithm aims to have both sequences converging to the same solution, which would then satisfy all constraints in (P). In [7], the strategy applied to obtain the sequence of integer solutions in the Feasibility Pump algorithm is based on the solution of MILP problems. Whereas in [8], a rounding procedure is applied for the same purpose. Versions of Feasibility Pump for the nonconvex MINLP problems were also presented in [2,13].

In [8], the Diving heuristic is generalized for MINLP. This heuristic searches for feasible solutions by simulating a sort of “diving” in a path of the BB enumeration tree. A version of the improvement heuristic RINS [15] for MINLP is also presented in [8], where a search for better solutions is performed by fixing a subset of integer variables and solving the resulting MINLP subproblem. Still applying the same idea of fixing variables and solving the resulting subproblems, the heuristic RENS [6] searches for the best rounded feasible solution around a solution for the continuous relaxation of (P). Also in [4], considering a fixed subset of variables, MILP subproblems are solved with the purpose of finding feasible solutions for (P). Another improvement heuristic for MILP based on the resolution of integer subproblems is Local Branching [19], extended to MINLP in [36]. Finally, in [30] a heuristic combining elements from distinct methodologies, such as Local Branching, branch-and-bound, and metaheuristics, is proposed.

In this work, we propose two algorithms for binary MINLP in the context of feasibility heuristics. Our algorithms are based on the idea of minimizing the integrality gap, and can be applied even in cases where nonconvexities occur in the problem formulation. Section 2 introduces the fundamental ideas behind our approach, as well as an exact (not practical) basic algorithm for solving binary MINLP problems. In Sect. 3, we use the fundamentals of Sect. 2 to develop a feasibility heuristic for the problem addressed. We have observed that the algorithm presented in this section is able to find good feasible solutions, yet at a relatively high computational cost. To overcome this drawback, we develop in Sect. 4 an algorithm with the same purpose, but focusing on finding feasible solutions with less computational cost. Numerical results are shown in Sect. 5, where we compare the performance of our methods with algorithms from the literature. Finally, in Sect. 6, we draw conclusions about the work developed.

2 Fundamentals

Solution methods for problem (P) commonly address its relaxation, where the integrality constraints are omitted. Therefore, the following Nonlinear Programming (NLP) problem, known as the continuous relaxation of (P), is considered:

$$\begin{array}{rll} (\tilde{P}) \ \min\limits_{x,\,y} & f(x, y) & \\ \text{s. t.:} & g(x, y) \ \leq \ 0, & \\ & 0 \leq y \leq 1, & \\ & x \in X, \ \ y \in Y. & \end{array} \tag{2}$$

Let $(\tilde{x}, \tilde{y})$ be a solution of $(\tilde{P})$. We introduce the *integrality gap* (or simply *gap*) of $(\tilde{x}, \tilde{y})$ as a measure of how far this solution is from satisfying the integrality constraints, which are relaxed in $(\tilde{P})$. As the integer variables are all binary in (P), a possible expression for the integrality gap is:

$$\text{gap}(y) = \sum_{i=1}^{n_y} y_i(1 - y_i).$$

To take into account the integrality constraints of (P) consists, therefore, in considering only the solutions in the feasible region of $(\tilde{P})$, for which the respective gap is equal to zero. Our motivation for the development of a new algorithm for binary MINLP comes, thereby, from the fact that we can substitute the integrality constraints with the constraints $\text{gap}(y) = 0$ and $0 \leq y \leq 1$, obtaining the following NLP problem:

$$
\begin{aligned}
(\bar{P})\quad \min_{x,y}\quad & f(x,y) && (3)\\
\text{s. t.:}\quad & g(x,y) \le 0, && (4)\\
& \sum_{i=1}^{n_y} y_i(1-y_i) = 0, && (5)\\
& 0 \le y \le 1, && (6)\\
& x \in X,\ y \in Y. && (7)
\end{aligned}
$$

Although $(\bar{P})$ is a problem in continuous domain, constraint (5) adds a high degree of nonconvexity and leads to a disconnected feasible region as much as the integrality restrictions do, which greatly hinders the practical process of obtaining one of its global optimal solutions. For this reason, Murray and Ng propose in [35] an algorithm for an MINLP problem with binary variables and linear constraints, where constraint (5) is penalized in the objective function. To deal with the nonconvexity of the resulting objective function, a smoothing strategy is then adopted. This same strategy of penalizing constraint (5) in the objective function had previously been addressed by Raghavachari in [38] as well, still in the context of Mixed Integer Linear Programming (MILP). More recently, the reformulation $(\bar{P})$ of the MINLP problem (P) was also considered in [32] by López and Beasley. In their solution approach, called formulation space search, the authors replace constraint (5) by $\sum_{i=1}^{n_y} y_i(1-y_i) \le \delta$, and solve a sequence of problems, where δ is iteratively reduced from a suitably selected initial value. The goal is to solve a sequence of different formulations of the problem aiming at having the nonlinear solvers converging to diverse feasible solutions, and increasing the chances of obtaining solutions of good quality.

We propose a different approach to deal with the difficulty introduced by constraint (5). Noting that the minimum possible value assumed by the measure gap on a feasible solution of $(\tilde{P})$ is zero, we consider the solution of another optimization problem where the objective consists only in minimizing the integrality gap:

$$
\begin{aligned}
\left(P^G\right)\quad \min_{x,y}\quad & \sum_{i=1}^{n_y} y_i(1-y_i) && (8)\\
\text{s. t.:}\quad & g(x,y) \le 0, && (9)\\
& 0 \le y \le 1, && (10)\\
& x \in X,\ y \in Y. && (11)
\end{aligned}
$$

We refer to problem (P^G) as the *integrality gap minimization problem*. Although the objective function of (P^G) is concave (not convex), which still brings difficulties to get its solution, its feasible region is convex, which favors the application of general NLP packages to the problem. Note also that this problem does not consider the original objective function of (P), aiming at obtaining only a feasible solution. It is straightforward to verify that if the feasible region of (P) is nonempty, $(\bar{x}, \bar{y})$ is a global optimal solution of (P^G) if, and only if, it is feasible for (P).

We should notice that at this point the integrality gap minimization problem may resemble the rounding step of the nonlinear Feasibility Pump in [7], because it tries to go towards integer feasibility, while neglecting the original objective. However, although both procedures aim to find an integer solution, they have different concepts when feasibility is considered. The rounding step in Feasibility Pump aims to find an integer solution that is close to the current feasible solution for the continuous relaxation, but not necessarily feasible for the original problem. On the other hand, the solution of our gap minimization problem aims to find an integer solution that is feasible to the original problem. Unlike the Feasility Pump rounding step, the resolution of the gap minimization problem concerns about the feasibility of the original problem, with no intention of being close to any solution of the continuous relaxation of the problem.

It is also important to note that we could bring some convexity to problem $\left(P^G\right)$, by replacing the gap in its objective function by the penalized function $f(x, y) + \mu \sum_{i=1}^{n_y} y_i$ $(1 - y_i)$, thus taking advantage of the convexity of the original objective function of the problem. Nevertheless, this would divert us form our main goal of generating feasible solutions as quick as possible, as better contextualized in the next sections.

Obtaining a global optimal solution for problem (P^G) certainly yields a feasible solution for (P). Nevertheless, we cannot predict much about the quality of this solution, since the original objective function of (P) is not considered in this new problem. In fact, any feasible solution for (P) could be obtained by this process, including a solution that maximizes the objective function $f(x, y)$ instead of minimizing it. To remedy this fact, we adopted two strategies to improve the solution of (P^G). The first strategy consists of a local search on a neighborhood of the solution. Denoting by $(\bar{x}, \bar{y})$ a global optimal solution obtained for (P^G) (feasible for (P)), we perform a local search in its neighborhood solving the following NLP problem obtained from (P) when we fix the variable y at $\bar{y}$:

$$\begin{aligned} (P(\bar{y}))\ \min_{x}\ & f(x, \bar{y}) \\ \text{s. t.:}\ & g(x, \bar{y}) \ \le\ 0, \\ & x \in X. \end{aligned} \tag{12}$$

We note that the resolution of $(P(\bar{y}))$ provides a solution $\tilde{x}$, such that $(\tilde{x}, \bar{y})$ is the best solution for (P) having y fixed at $\bar{y}$ (or one of the best solutions, in case $(P(\bar{y}))$ have multiple optimal solutions). It is important to mention that this strategy of solving problem $(P(\bar{y}))$ is adopted in some algorithms for MINLP that use linear approximations, such as Generalized Benders Decomposition [24], Outer Approximation [17,20], and LP/NLP based branch-and-bound [37]. The solution of $(P(\bar{y}))$ is used in these algorithms to generate valid upper bounds for (P), as well as to strengthen the respective MILP relaxations considered by them.

Our second strategy to improve the solution of (P^G) is to solve the following NLP problem, which also aims at minimizing the integrality gap, but adopts an objective level cut:

$$(\hat{P}^G(z^u, \epsilon_c))\ \min_{x,\, y}\ \sum_{i=1}^{n_y} y_i(1 - y_i) \tag{13}$$

$$\text{s. t.:}\ g(x, y) \ \le\ 0, \tag{14}$$

$$f(x, y) \ \le\ z^u - \epsilon_c, \tag{15}$$

$$0 \le y \le 1, \tag{16}$$

$$x \in X,\ \ y \in Y, \tag{17}$$

where z^u is the best upper bound known for (P), and $\epsilon_c > 0$ is a convergence tolerance. We refer to problem $(\hat{P}^G(z^u, \epsilon_c))$ as the *integrality gap minimization problem with objective level cut*. Note that the only difference between problems (P^G) and $(\hat{P}^G(z^u, \epsilon_c))$ is that the latter brings the objective level cut (15), which has the purpose of generating a feasible solution that improves the best known upper bound, when possible.

Having described the subproblems of interest, we next introduce an algorithm to solve problem (P). Although this algorithm is not of practical use to solve the problem to optimality, it contains the fundamental ideas considered to develop the feasibility heuristics proposed in the remainder of this paper. We present this basic algorithm as Algorithm 1.

```
Input: $(P)$: binary MINLP problem, $\epsilon_c$: convergence tolerance.
Output: $(x^*, y^*)$: optimal solution for $(P)$.
1  Let $(\bar{x}^0, \bar{y}^0)$ be a global optimal solution of $(P^G)$;
2  if $\bar{y}^0$ is integer then
3      Let $\tilde{x}^0$ be an optimal solution of $(P(\bar{y}^0))$;
4      $(x^*, y^*) = (\tilde{x}^0, \bar{y}^0)$;
5      $z^u = f(\tilde{x}^0, \bar{y}^0)$;
6      $k = 1$;
7      while $(\hat{P}^G(z^u, \epsilon_c))$ is feasible do
8          Let $(\bar{x}^k, \bar{y}^k)$ be a global optimal solution of $(\hat{P}^G(z^u, \epsilon_c))$;
9          if $\bar{y}^k$ is integer then
10             Let $\tilde{x}^k$ be an optimal solution of $(P(\bar{y}^k))$;
11             $(x^*, y^*) = (\tilde{x}^k, \bar{y}^k)$;
12             $z^u = f(\tilde{x}^k, \bar{y}^k)$;
13         else
14             stop the algorithm;
15         $k = k + 1$;
```

Algorithm 1: Basic algorithm for binary MINLP.

We note that a feasible solution $(\bar{x}^0, \bar{y}^0)$ for (P) can be obtained in line 1 of Algorithm 1, by solving (P^G). In case of success, we take $\bar{y}^0$ to solve $(P(\bar{y}^0))$ hoping to improve the solution obtained. The loop between lines 7 and 15 considers problem $(\hat{P}^G(z^u, \epsilon_c))$ instead of (P^G) with the purpose of obtaining a sequence of increasingly better solutions $(\tilde{x}^k, \bar{y}^k)$ with respect to the objective function, until $(\hat{P}^G(z^u, \epsilon_c))$ gets infeasible or its resolution gives a non-integer solution $\bar{y}^k$. Here, it is supposed that (P) has one or more optimal solutions. Even so, if problem (P) is infeasible, this fact will be easily detected in lines 1 and 2.

The following theorem assures the convergence of Algorithm 1 in a finite number of iterations, if global optimality is ensured when solving (P^G) and $(\hat{P}^G(z^u, \epsilon_c))$.

Theorem 1 *If the integrality gap minimization problems (P^G) and $(\hat{P}^G(z^u, \epsilon_c))$ are solved to global optimality during the execution of* Algorithm 1*, then the algorithm converges to an optimal solution of (P), with tolerance ϵ_c, in a finite number of iterations.*

Proof Whereas (P^G) is solved to global optimality, as (P) is feasible, Algorithm 1 obtains, in line 1, an initial feasible solution $(\bar{x}^0, \bar{y}^0)$, with $\bar{y}^0$ integer. The resolution of $(P(\bar{y}^0))$ yields then the solution $(\tilde{x}^0, \bar{y}^0)$, which is the best feasible solution of (P) having y fixed at $\bar{y}^0$, and $f(\tilde{x}^0, \bar{y}^0)$ is used as the initial upper bound z^u.

In the remaining of its execution, Algorithm 1 considers problem $(\hat{P}^G(z^u, \epsilon_c))$, which contains the objective level cut associated to the current upper bound z^u. Since this cut initially corresponds to the upper bound given by the best feasible solution that has y fixed at $\bar{y}^0$, we can guarantee that $\bar{y}^0$ cannot appear as a value for y in any feasible solution of $(\hat{P}^G(z^u, \epsilon_c))$.

Thereby, if $(\tilde{x}^0, \bar{y}^0)$ is not an optimal solution to (P) (with tolerance ϵ_c), the solution $(\bar{x}^1, \bar{y}^1)$ is feasible to (P) and is such that $\bar{y}^1 \neq \bar{y}^0$ and $f(\bar{x}^1, \bar{y}^1) < f(\tilde{x}^0, \bar{y}^0)$. This solution is used to generate $(\tilde{x}^1, \bar{y}^1)$, which is the best feasible solution of (P) having y fixed at $\bar{y}^1$, and finally, this last solution is used to update the upper bound z^u.

Extending this reasoning, we have that, at each iteration k, the resolution of $(\hat{P}^G(z^u, \epsilon_c))$ produces an integer solution for y with value $\bar{y}^k$ different from all the values of y obtained in previous iterations (i.e., $\bar{y}^k \neq \bar{y}^j, \forall j \in \{0, 1, \ldots, k-1\}$), until $(\hat{P}^G(z^u, \epsilon_c))$ becomes infeasible or produces a non-integer $\bar{y}^k$ on its global optimal solution. Moreover, the sequence of upper bounds z^u is a decreasing sequence. As the number of distinct binary solutions for y is finite, the algorithm will stop in a finite number of iterations, and the last feasible solution obtained to (P) will be optimal for (P) with tolerance ϵ_c. □

As Theorem 1 states, the convergence of Algorithm 1 to an optimal solution of (P) is strongly based on obtaining global optimal solutions for the integrality gap minimization problems (P^G) and $(\hat{P}^G(z^u, \epsilon_c))$. Although these nonconvex problems have particularities that could facilitate their treatment, as concave and separable quadratic objective function, together with a known lower bound, the practical process of solving these problems can still be quite hard. This disadvantage is worsened by the fact that a number of such problems needs to be addressed and it may be difficult to obtain an efficient computational routine to perform this task. These reasons make Algorithm 1 not practical for solving binary MINLP problems to optimality. However, the idea of minimizing the integrality gap can still be very useful in the search for feasible solutions, as we will see in the following sections.

3 Integrality gap minimization algorithm, version 1

Although ensuring global optimality when solving gap minimization problems is a hard task, we observe that the basic idea of minimizing the integrality gap without the commitment of global optimality can lead to the generation of feasible solutions quickly. In this context, we developed a feasibility heuristic based on the idea of gap minimization integrated to a Branch-and-Bound (BB) procedure. This new algorithm does not require global optimality in the resolution of integrality gap minimization problems, which allows it to be implemented with some of the several available computational routines of nonlinear programming. We refer to our algorithm as *integrality gap minimization algorithm 1*, or IGMA1.

To our knowledge, the best methodology to solve the gap minimization problems is the use of a spatial branch-and-bound algorithm [18]. Considering the application of a basic spatial BB algorithm to one of the gap minimization problems, when branching, it's necessary to choose a variable y_p to generate two new branches. In one branch, the constraint $y_p \leq \upsilon$ is added, while in the other, the constraint $y_p \geq \upsilon$ is added, where υ is a real number selected in the interval $(\bar{l}_{y_p}, \ \bar{u}_{y_p})$, and $\bar{l}_{y_p}$ and $\bar{u}_{y_p}$ are, respectively, the lower and upper bounds for y_p in the branch being currently explored (initially, $\bar{l}_{y_p} = 0$ and $\bar{u}_{y_p} = 1$).

We point out that, although a basic spatial BB algorithm divides the space of the variables on fractional values υ, we know that the solutions of interest have integer values for the variables y, since these are the solutions that minimize our objective function, which measures the integrality gap. Based on this valuable information, we construct a new algorithm by integrating Algorithm 1 with a branch-and-bound procedure that divides the space based on the integer values that y can assume, similarly to the branch-and-bound used in integer programming.

We then define the integrality gap minimization problem with objective level cut taken at a given node k of the BB enumeration tree:

$$(\hat{P}^G(Y^k, z^u, \epsilon_c)) \ \min_{x,\, y} \sum_{i=1}^{n_y} y_i(1 - y_i) \tag{18}$$

$$\text{s. t: } g(x, y) \ \leq \ 0, \tag{19}$$

$$f(x,y) \leq z^u - \epsilon_c, \tag{20}$$
$$0 \leq y \leq 1, \tag{21}$$
$$x \in X,\ y \in Y^k, \tag{22}$$

where Y^k is a partition of Y relative to node k.

```
Input: $(P)$: binary MINLP problem, $\epsilon_c$: convergence tolerance.
Output: $(x^*, y^*)$: feasible solution of $(P)$.
1  $z^u = \infty$;
2  $Y^0 = Y$;
3  Let $N := \{0\}$ be the initial set of open nodes on the BB enumeration tree;
4  $i = 0$;
5  while $N \neq \emptyset$ do
6      Select a node $k$ from $N$;
7      if $(\hat{P}^G(Y^k, z^u, \epsilon_c))$ is feasible then
8          Let $(\bar{x}^k, \bar{y}^k)$ be an optimal solution (possibly local) of $(\hat{P}^G(Y^k, z^u, \epsilon_c))$;
9          while $\bar{y}^k$ is integer do
10             Let $\tilde{x}^k$ be an optimal solution of $(P(\bar{y}^k))$;
11             $(x^*, y^*) = (\tilde{x}^k, \bar{y}^k)$;
12             $z^u = f(\tilde{x}^k, \bar{y}^k)$;
13             if $(\hat{P}^G(Y^k, z^u, \epsilon_c))$ is infeasible then
14                 go to line 21;
15             Let $(\bar{x}^k, \bar{y}^k)$ be an optimal solution (possibly local) of $(\hat{P}^G(Y^k, z^u, \epsilon_c))$;
           // Branching
16         Select a variable $y_j$ with non-integer value $\bar{y}^k_j$;
17         $Y^{i+1} = Y^k \cap \{y \in \mathbb{R}^{n_y} : y_j = 0\}$;
18         $Y^{i+2} = Y^k \cap \{y \in \mathbb{R}^{n_y} : y_j = 1\}$;
19         $N = N \cup \{i+1, i+2\}$;
20         $i = i + 2$ ;
21     $N = N \backslash \{k\}$ ;
```

Algorithm 2: IGMA1 algorithm.

IGMA1 is designed to be used with NLP packages that can converge to local optimal solutions of the gap minimization problems. Each time an optimal solution $(\bar{x}^k, \bar{y}^k)$ with non-integer $\bar{y}^k$ is obtained for one of these problems, we partition the space branching on a variable y_j with fractional value $\bar{y}^k_j$. On the other hand, if an integer solution $\bar{y}^k$ is obtained, we use this new solution to solve $(P(\bar{y}^k))$ and generate the solution $(\tilde{x}^k, \bar{y}^k)$, which will update the upper bound z^u. We then solve once more the gap minimization problem in the same current partition k, but now with the updated objective level cut, considering the new upper bound. We repeat these steps in the current partition k until we get a solution $(\bar{x}^k, \bar{y}^k)$ for $(\hat{P}^G(Y^k, z^u, \epsilon_c))$, with $\bar{y}^k$ not integer, or until this problem becomes infeasible. In the first case, we branch. In the second case, we prune.

We present IGMA1 as Algorithm 2. Note that a unique branch-and-bound procedure is used in an integrated way to the solution of the gap minimization problems. A similar strategy was used by Quesada and Grossmann in [37], to present an algorithm for MINLP based on the Outer Approximation algorithm of [17,20], which originally can use several sequential applications of BB to solve approximate MILP problems. In the algorithm proposed in

[37], a unique BB procedure is used to solve an approximate MILP problem, which is then dynamically updated throughout the exploration of the nodes.

We emphasize that on IGMA1's branch-and-bound, no lower bounds are generated at each partition. For this reason, pruning by bound cannot be performed, with respect to the gap minimization objective and the original objective function also. Pruning by optimality is not also possible, since obtaining an integer solution in a partition does not guarantee the absence of better integer solutions in the same partition. Therefore, algorithm IGMA1 only eliminates partitions when pruning by infeasibility. Note, however, that the objective level cut (20) causes infeasibility on certain partitions where the upper bound cannot improve, i.e., partitions that would be pruned by bound in a traditional integer programming BB end up being pruned by infeasibility in our approach. Similarly, when the best integer solution of a partition is obtained, this same cut will cause infeasibility on the partition or on downstream partitions. Thus, pruning that would occur by optimality in a traditional BB also occur by infeasibility here. Finally, pruning that would occur in a traditional BB due to infeasibility, occurs in IGMA1 in the same way. It is important to remark that ϵ_c represents an absolute tolerance in Algorithm 2. For practical purposes, it is useful to adopt also a relative tolerance to compose the RHS term of the objective level cut (20) in order to avoid numerical instability that may appear with absolute tolerance alone. Thus, in our tests, we adopted both absolute and relative tolerances in the implementations of our approaches.

There is a clear correspondence between the traditional branch-and-bound algorithm for MINLP and IGMA1. The main difference concerns the fact that a traditional BB optimizes on each partition observing only the original objective function of the problem addressed, which can lead to solutions on the partitions that are far from integrality. A well-known consequence of this fact is that, often, a BB algorithm may require many iterations until it obtains a feasible solution. This fact can lead to high memory usage, as a list of open nodes is maintained in the procedure, and without feasible solutions, it is not possible to prune by bound and eliminate the corresponding nodes from this list. In some cases, delays in obtaining feasible solutions can also lead to unnecessary exploration of many nodes that would be pruned if a good upper bound was previously known.

Unlike the traditional branch-and-bound algorithm for MINLP, IGMA1 optimizes on the partitions searching for an integer solution that improves the best known upper bound. Wherefore, IGMA1 tends to find feasible solutions faster, enabling pruning in earlier stages, and consequently avoiding the unnecessary exploration of more nonpromising partitions.

Concerning the node selection strategy for IGMA1, one can select a node for which the parent node got the closest gap to zero, in addition, of course, to the traditional depth-first and breadth-first search strategies. Finally, as a strategy to select the branching variable, we can choose the variable y_j with the solution value farthest from an integer value, on the solution of the gap minimization problem associated to the partition considered.

It is straightforward to verify that IGMA1, described in Algorithm 2, even solving the gap minimization problems only locally, solves a binary convex MINLP problem to optimality in a finite number of iterations with convergence tolerance ϵ_c, since the algorithm is based on a space partitioning procedure and no partition is incorrectly discarded. Nevertheless, preliminary computational tests indicated that, compared to other exact algorithms from the literature, IGMA1 presented poor practical performance for this purpose, especially due to the concave objective function and the difficulty in attesting the infeasibility of the gap minimization problems. It is possible, however, that, as NLP solvers incorporate better techniques to deal with the minimization of concave quadratics on convex sets, IGMA1 becomes an exact algorithm more competitive in comparison to the others from the literature. Anyhow, the ability of quickly obtaining feasible solutions qualifies the algorithm to be used as a feasibility heuristic by simply adopting one or more additional stopping criteria, such as: (1) Maximum computation time; (2) Maximum number of iterations; (3) Maximum number of feasible solutions found.

It is worth mentioning that the solution of the gap minimization problems can be interrupted if it is detected that the computational routine used for this purpose is converging to an integer solution $\bar{y}^k$. In this case, one can directly solve problem $(P(\bar{y}^k))$ before the optimality of the gap minimization problems is attested.

4 Integrality gap minimization algorithm, version 2

We have verified that IGMA1 can, in several cases, quickly find feasible solutions to the problem addressed. However, since the objective function $f(x, y)$ is no longer optimized in the gap minimization problems, it is not recommended to stop the execution of the algorithm immediately after finding the first feasible solutions, as these first solutions may not be of good quality. Nevertheless, if we had a higher level of confidence that the first feasible solution found was good, we could stop the algorithm earlier and save computational effort. Aiming at developing a less computationally expensive algorithm, we present in this section the *integrality gap minimization algorithm 2*, or IGMA2.

The central idea behind IGMA2 is to adopt a standard nonlinear branch-and-bound procedure, where, at each node k, we solve the following continuous relaxation problem:

$$\begin{aligned} (\tilde{P}(Y^k)) \ \min_{x,y} \ & f(x, y) \\ \text{s. t.:} \ & g(x, y) \le 0, \\ & 0 \le y \le 1, \\ & x \in X, \ y \in Y^k, \end{aligned} \tag{23}$$

where Y^k is the partition of Y corresponding to node k. Assuming that $(\tilde{P}(Y^k))$ is feasible, let $(\tilde{x}^k, \tilde{y}^k)$ be one of its optimal solutions. Note that $f(\tilde{x}^k, \tilde{y}^k)$ is a lower bound for node k. If this value is smaller than the current upper bound z^u and $\tilde{y}^k$ is not integer, we solve then another variant of the gap minimization problem, where we are now looking for a solution that satisfy the integrality constraints in the neighborhood of $\tilde{x}^k$:

$$(\hat{P}^G(Y^k, \tilde{x}^k, \omega)) \ \min_{x,y} \ \sum_{i=1}^{n_y} y_i(1 - y_i) \tag{24}$$

$$\text{s. t:} \ g(x, y) \le 0, \tag{25}$$

$$\sum_{i=1}^{n_x} (x_i - \tilde{x}_i^k)^2 \le \omega^2, \tag{26}$$

$$0 \le y \le 1, \tag{27}$$

$$x \in X, \ y \in Y^k, \tag{28}$$

where Y^k is the partition of Y relative to node k. Note that constraint (26) is quadratic, separable and convex, and uses a parameter ω to determine the size of the neighborhood of $\tilde{x}^k$ being explored. Problem $(\hat{P}^G(Y^k, \tilde{x}^k, \omega))$, therefore, searches for a feasible solution for (P), considering a hypersphere centered at $\tilde{x}^k$.

We hope that by solving problem $(\hat{P}^G(Y^k, \tilde{x}^k, \omega))$, we find a solution $(\bar{x}^k, \bar{y}^k)$ that has integer value for $\bar{y}^k$ with $\bar{x}^k$ next to $\tilde{x}^k$. Since $\tilde{x}^k$ is a solution for x in the continuous

```
Input: $(P)$: binary MINLP problem, $\epsilon_c$: convergence tolerance.
Output: $(x^*, y^*)$: feasible solution of $(P)$.
1  $z^u = +\infty$;
2  $Y^0 = Y$;
3  Let $N = \{0\}$ be the initial set of open nodes on the BB enumeration tree;
4  $i = 0$;
5  Let $L^i$ be the lower bound of node $i$;
6  $L^0 = -\infty$;
7  while $N \neq \emptyset$ do
8      Select a node $k$ from $N$;
9      if $(\tilde{P}(Y^k))$ is feasible then
10         Let $(\tilde{x}^k, \tilde{y}^k)$ be an optimal solution of $(\tilde{P}(Y^k))$;
11         if $f(\tilde{x}^k, \tilde{y}^k) < z^u - \epsilon_c$ then
12             if $\tilde{y}^k$ is integer then
13                 $(x^*, y^*) = (\tilde{x}^k, \tilde{y}^k)$;
14                 $z^u = f(\tilde{x}^k, \tilde{y}^k)$;
15                 stop the algorithm;
16             else
17                 Let $(\dot{x}^k, \dot{y}^k)$ be an optimal solution (possibly local) of $(\hat{P}^G(Y^k, \tilde{x}^k, \omega))$;
18                 $\bar{y}^k = [\dot{y}^k]$;                                        // rounding
19                 if $(P(\bar{y}^k))$ is feasible then
20                     Let $\check{x}^k$ be an optimal solution of $(P(\bar{y}^k))$;
21                     $(x^*, y^*) = (\check{x}^k, \bar{y}^k)$;
22                     $z^u = f(\check{x}^k, \bar{y}^k)$;
23                     stop the algorithm;
                   // Branching
24                 Select a variable $y_j$ with non-integer value $\tilde{y}^k_j$;
25                 $Y^{i+1} = Y^k \cap \{y \in \mathbb{R}^{n_y} : y_j = 0\}$;
26                 $Y^{i+2} = Y^k \cap \{y \in \mathbb{R}^{n_y} : y_j = 1\}$;
27                 $L^{i+1} = L^{i+2} = f(\tilde{x}^k, \tilde{y}^k)$;
28                 $N = N \cup \{i + 1, i + 2\}$;
29                 $i = i + 2$;
30     $N = N \backslash \{k\}$;
```

Algorithm 3: IGMA2 algorithm.

relaxation on node k, we expect that the solution obtained corresponds to a good value for the objective function. Based on this expectation, we can stop the execution of the algorithm after finding the first integer solution and submitting it to the local search procedure, by solving problem $(P(\bar{y}^k))$. Moreover, aiming at finding a feasible solution to (P) next to $(\tilde{x}^k, \tilde{y}^k)$, we take $(\tilde{x}^k, \tilde{y}^k)$ as the initial solution when solving $(\hat{P}^G(Y^k, \tilde{x}^k, \omega))$. We have noted that this strategy, together with the use of constraint (26), have in general led to less computational effort needed to solve $(\hat{P}^G(Y^k, \tilde{x}^k, \omega))$ when compared to the minimization gap problems in IGMA1.

IGMA2 is described in Algorithm 3. Note that IGMA2 stops after finding the first solution with integer y. However, the execution could continue if the user wanted to look for better feasible solutions. To increase the effectiveness of IGMA2, even when the resolution of $(\hat{P}^G(Y^k, \tilde{x}^k, \omega))$ does not present an integer solution, we take $\bar{y}^k$ as the rounding of this solution and then we proceed with the resolution of $(P(\bar{y}^k))$ (lines 18 and 20). Finally, we point out that problem $(\hat{P}^G(Y^k, \tilde{x}^k, \omega))$ does not adopt any objective level cut.

From Algorithm 3, we see that IGMA2 can be easily integrated into a branch-and-bound exact procedure, and may improve its performance by quickly locating feasible solutions and consequently enabling pruning by bound more effectively. Note that strategies used to improve the performance of traditional branch-and-bound algorithms, such as pseudo-costs, strong branching and cut generation, can also be adopted in IGMA2.

4.1 Assignment of weights to integer variables

The gap minimization problems previously addressed, consider all integer variables with identical weights in the objective function. In practice, one often knows that prioritizing the integrality of a given subset of variables may facilitate the solution of the problem, for example, by speeding up the production of feasible solutions or by faster increasing the lower bounds on the partitions generated by the branching process. In both cases, pruning by bound is favored and the total time required to explore the nodes on the BB tree is reduced. From this perspective, many MILP solvers allow the user to assign different priorities to the integer variable of the problem optimized. These priorities are taken into account on the selection of the branching variable.

Both IGMA1 and IGMA2 can incorporate weights for integer variables more effectively than the traditional BB. In addition to considering the priorities when branching, one can also replace the objective function of the gap minimization problems with:

$$\sum_{i=1}^{n_y} \beta_i y_i (1 - y_i) \tag{29}$$

where $\beta_i \geq 0$ is the weight corresponding to variable y_i. We emphasize that, considering $\beta_i = 1$, for all $i = 1, \ldots, n_y$, is equivalent to consider the objective function of (P^G). Note also that it is possible to set $\beta_i = 0$ with no loss, if the integer variable y_i is already fixed.

We finally note that considering expression (29) together with a dynamic scheme of updating the weights, can be particularly interesting. For example, if it is detected that on several nodes of the BB tree, the local optimal solutions have positive integrality gap for a given variable y_i, i.e. $y_i(1 - y_i) > 0$, it is possible to increase the value of β_i during the execution of the algorithm, giving more priority from that point on, to the minimization of the integrality gap of that specific variable. On the other hand, if a given variable y_j easily reaches integrality in the optimal solutions of the subproblems, β_j can be decreased to encourage the model to drive the gap of other integer variables to zero.

Thus, for each node k explored in the branch-and-bound enumeration of IGMA1 and IGMA2, one can consider:

$$\beta_{i,k} = \begin{cases} 0, & \text{if } y_i \text{ is fixed at node } k \\ \beta_{i,0} + \gamma \sigma_i, & \text{otherwise} \end{cases} \tag{30}$$

where γ is a non-negative parameter, σ_i is the average *gap* of variable y_i on all nodes of the tree that were already explored, in which y_i is not fixed, and $\beta_{i,0}$ is the initial weight for y_i. One can take $\beta_{i,0} = 1$, or use weights pre-defined by the user. Note that in case $\gamma = 0$, each variable has the same weight on the objective functions of the gap minimization problems, on all nodes.

We have observed that this dynamic update of the weights β_i can improve the performance of IGMA1 and IGMA2, especially because the local optimal solution obtained for each gap

Table 1 Statistics on groups of test problems

Group	Data	Min.	Max.	Median
Convex	Vars	4	107,222	280
	Binary vars (%)	0.01	0.90	0.35
	Constrs	4	108,217	435
	Linear constrs (%)	0.17	1.00	0.97
Nonconvex	vars	2	31,926	232
	Binary vars (%)	0.01	1.00	0.41
	Constrs	1	164,321	250
	Linear constrs (%)	0.00	1.00	0.82

minimization problem is quite dependent on the algorithm considered in the local NLP routine, and also on its implementation. Thereby, the dynamic update can lead to more diversity in the process of obtaining local optimal solutions for the problems.

Finally, it is worth mentioning that the strategy of dynamic updating the weights β_i has some similarities with the well-known strategy commonly adopted in BB algorithms, denominated as pseudo-cost. However, these two strategies have distinct purposes, as pseudo-costs are used to select branching variables (The reader interested in information regarding pseudo-costs on the context of BB for MINLP is encouraged to consult [10]).

5 Computational results

To carry out computational experiments, we consider a large set of benchmark binary MINLP test instances from MINLPLib2 [22] written in AMPL language [21]. Altogether, 668 test instances divided into two groups were considered: convex (205) and nonconvex (463) problems. Table 1 provides basic information about the instances in each of these groups.

We compare IGMA1 and IGMA2 with three feasibility heuristics for MINLP found in the literature:

– Feasibility Pump (FP) [8];
– Diving heuristic (adopting fractional selection strategy) [8];
– Outer Approximation based Feasibility Pump (OAFP) [7].

We made our own implementations of all the algorithms mentioned in `C++`. For the compilation of the source code, we used the compiler `ICPC 16.0.4` [28]. The computational experiments were conducted on a machine with a core i7 6700 processor, 3.4 GHz, 8 MB cache, and running under the operating system `Ubuntu 16.04`. All algorithms have been configured to use only a single thread. For solving NLP problems, `Ipopt 3.12.6` [42] with `MA57` from `HSL 2014.01.10` [40] was adopted. For solving the MILP problems in OAFP, the package `Xpress 28.01` [16] was used. Based on the assumption that feasibility heuristics must find solutions quickly, the maximum execution time of each algorithm in each test instance was limited to 200 seconds. All the implementations mentioned here relied on the use of a preprocessing routine for linear constraints according to [39]. IGMA1 was run using the dynamic weight update strategy for the integer variables, described by (30), with $\gamma = 10$. For the convergence tolerance ϵ_c, an absolute tolerance of 0.0001 and a relative tolerance of 0.1 were adopted. The parameter ω in the constraint (26) in IGMA2, was defined as $1 + \lceil 0.05(n_x + n_y) \rceil$. IGMA1 adopted as the node selection strategy, the choice of the node

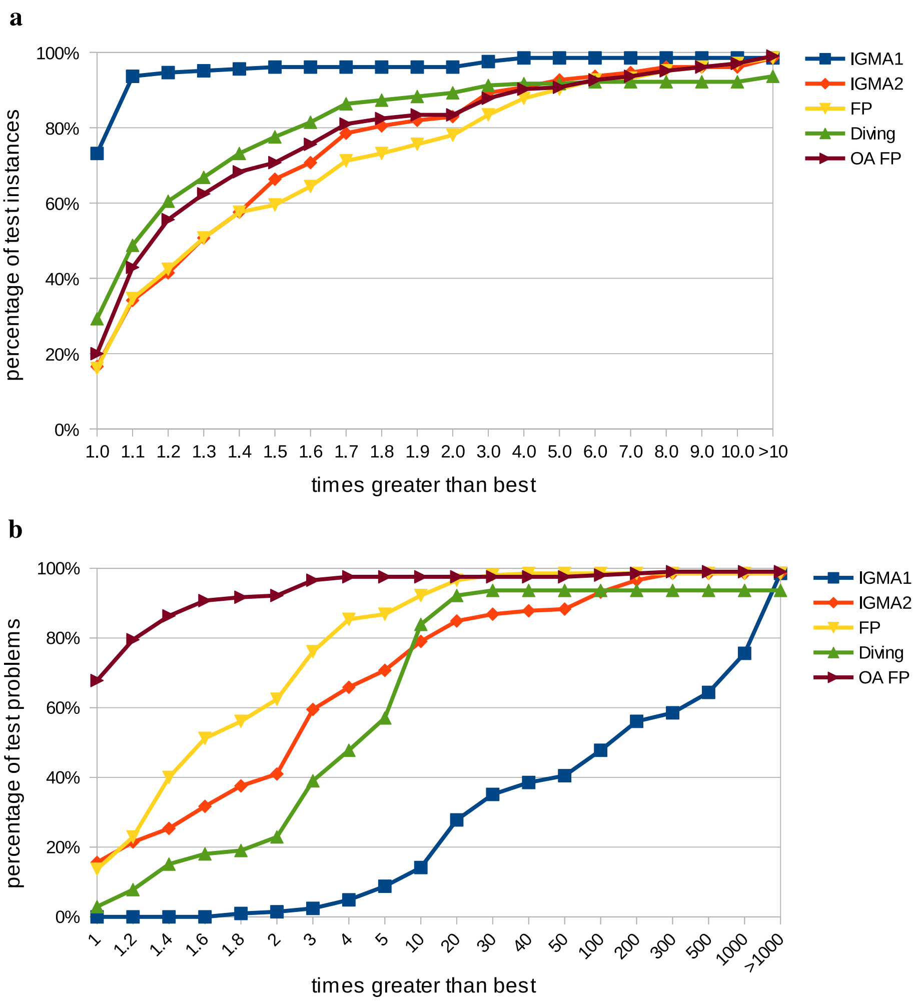


Fig. 1 Relative comparison between heuristics for convex problems; **a** best solutions found by the heuristics; **b** computational time spent by the heuristics

for which the parent node has the gap closest to zero. IGMA2, on the other hand, adopted the classic depth-first search strategy. The integer variable y_j with the most fractional value $\bar{y}_j$ was chosen as the branching variable in IGMA1. IGMA2 adopted the same strategy with respect to the value of $\tilde{y}_j$. In order to solve the gap minimization problems in IGMA1, we take random solutions as starting points. With regard to IGMA2, we use as a starting point the solution of the continuous relaxation on the partition being explored. The maximum number of iterations of `Ipopt` was set at 6000 for IGMA1. For the other algorithms, the default value of `Ipopt` (3000) was used. All these parameters were set empirically.

Since the heuristics from the literature that we have implemented were all developed for convex MINLP, we compare the results of IGMA1 and IGMA 2 with them only for the convex instances. Recall that IGMA1 and IGMA2 were also developed for convex MINLP, without no special mechanism to treat nonconvexities, and so we do not intend to compete with heuristics developed for nonconvex problems. Nevertheless, we present the results of IGMA1 and IGMA2 for nonconvex instances with the only purpose of showing their generality and robustness regarding this applicability. Feasibility heuristics specifically developed to nonconvex MINLP can be found in [2,4,13,30].

Figures 1 and 3 depict performance profile graphs generated from the application of the heuristics considered to the convex (Fig. 1) and nonconvex instances (Fig. 3), respectively. The graph on the top (a) refers to the value of the objective function on the best solution found by each algorithm, i.e., to the best upper bound to the optimal solution value of the instance, while the graph on the bottom (b) refers to the computational time used. Note that the data are normalized in relation to the best result obtained by all approaches for each instance. On the horizontal axis, the abscissa indicates the number of times the result obtained by the algorithm was greater than the best result obtained among all the algorithms. In the vertical axis, the ordinate indicates the percentage of instances achieved by each approach. More specifically, if the curve for a given algorithm passes through the point (α, τ), it indicates that for $\tau\%$ of the instances, the result obtained by the algorithm is less than or equal to α times the best result obtained among all algorithms.

For the convex instances (Fig. 1), we can state that IGMA1 presented the best results with respect to the best solution found (Fig. 1a), being able to find the best solution among all the approaches for 73% of the instances in this group. Diving, in turn, found the best solution among all approaches for 29% of the group instances, Concerning FP, OAFP and IGMA2, this number was 16%, 20% and 17%, respectively. Diving was able to find a feasible solution only for 192 test instances (93.7 %), while IGMA1, IGMA2, and FP found feasible solutions for 202 test instances (98.5%). Finally, OAFP was able to find a solution for 203 instances (99.0%). In general terms, we can say that IGMA2 had a very similar performance to FP and OAFP on this regard.

Concerning the computational time (Fig. 1b), we can notice that IGMA1 required more effort than the other approaches, and OAFP was the most efficient heuristic for the group of instances considered. IGMA2 performed well compared to FP, Diving and IGMA1. We note that one of the reasons why the computational cost of OAFP was significantly lower than of the other heuristics comes from the fact that this algorithm uses a linearization procedure for nonlinear terms, which, when applied to the convex functions, constructs an outer approximation that seems to work well in practice.

These outer approximations lead to valid linear relaxations for problem (P). When we solve these relaxations with an advanced MILP package, such as `Xpress`, we take advantage of all the maturity and efficiency achieved in the MILP area in the last decades. Thus, OAFP can find feasible solutions in small computational time for this group of instances.

Figure 2 depicts a new performance profile comparing the solutions found by each heuristic with the known optimal solutions of the convex instances. We note that IGMA1 found an optimal solution for 20% of these test instances, IGMA2 and FP found an optimal solution for 10% of these test instances, and finally, Diving and OAFP found an optimal solution for 15% and 11% of these test instances, respectively. It is remarkable that, for 79% of these test instances, IGMA1 found a solution with objective function value no more than 10% greater than the optimal solution value, and, for 88% of the test instances, IGMA1 found a solution with objective function value no more than 20% greater than the optimal value. We also point out that this result is much better than the results shown for the other heuristics, which, in general, perform similarly in this regard. The main reason for the superiority of IGMA1 here comes from the fact that it has mechanisms to improve the feasible solution found through the use of the objective level cut and continuation of the search for new solutions that satisfy this cut.

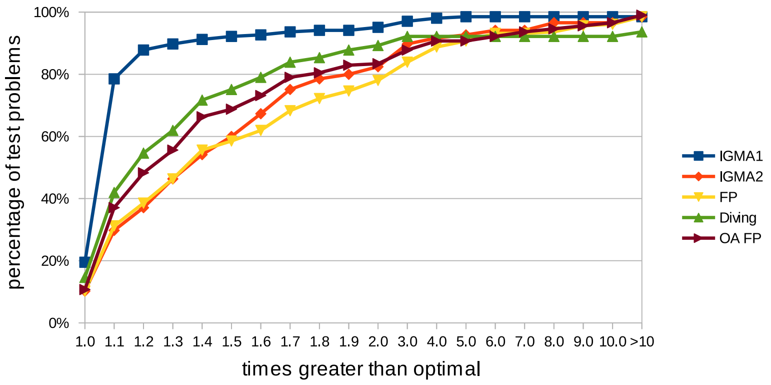


Fig. 2 Comparison between solutions found by the heuristics with known optimal solutions for convex problems

Regarding the application of IGMA1 and IGMA2 to the nonconvex instances (Fig. 3), we see that IGMA1 had better results than IGMA2 once more, when concerning the best solution found (Fig. 3a). It was able to find the best solution between both approaches for 55% of the test instances in this group. For IGMA2, this number was 40%. Furthermore, IGMA1 was able to find solutions for 81% of the test instances, whereas IGMA2 found feasible solutions for 75%.

Regarding the computational time for nonconvex instances (Fig. 3b), IGMA2 presented better results than IGMA1. However, we observe that the performance of IGMA1 is closer to IGMA2, when compared to the group of convex instances.

Figure 4 depicts one last performance profile comparing the solutions found by our heuristics with the known optimal solutions of the nonconvex instances. We note that IGMA1 and IGMA2 found an optimal solution for 14 and 13% of these test instances, respectively. It is also notable that, for 76% of these test instances, IGMA1 found a solution with objective function value no more than 20% greater than the optimal solution value.

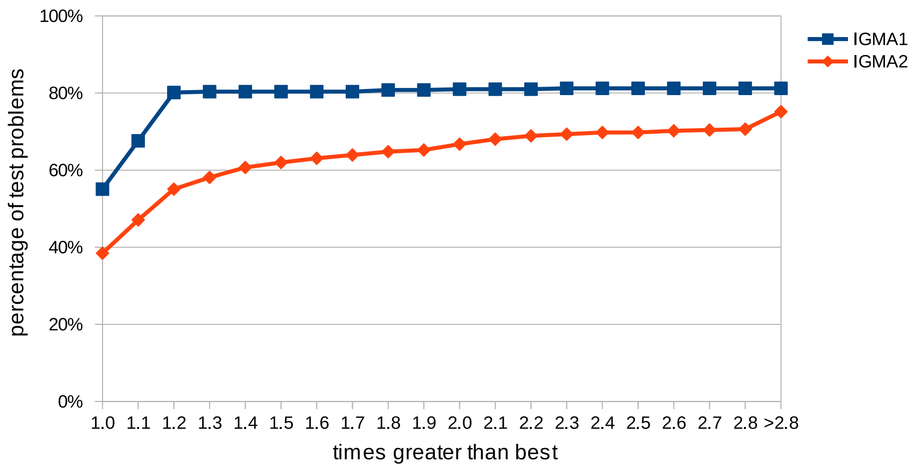
In general, we conclude that IGMA1 produces the best feasible solutions at the expense of significantly increased computational times. IGMA2, on the other hand, is a procedure of low computational cost, but not as good as IGMA1 with respect to the quality of the solutions obtained. We can thus highlight that in practical situations where it is crucial to start from good solutions without much concern about the computational time, IGMA1 may be the most appropriate choice. On the other hand, if obtaining a feasible solution in short time is the priority, IGMA2 may be more opportune.

6 Conclusions

Algorithms for binary mixed integer nonlinear programming problems based on the idea of minimizing the integrality gap were presented in this paper. From an exact algorithm, we propose two heuristics called integrality gap minimization algorithm (IGMA) versions 1 and 2.

Within the same framework, the algorithms proposed were compared to procedures from the literature and presented satisfactory results. IGMA1 in particular stood out for its ability to find, in average, better feasible solutions than the other heuristics. IGMA2, in turn, presented as a highlight the ability to find, in average, feasible solutions of similar quality to

a



b

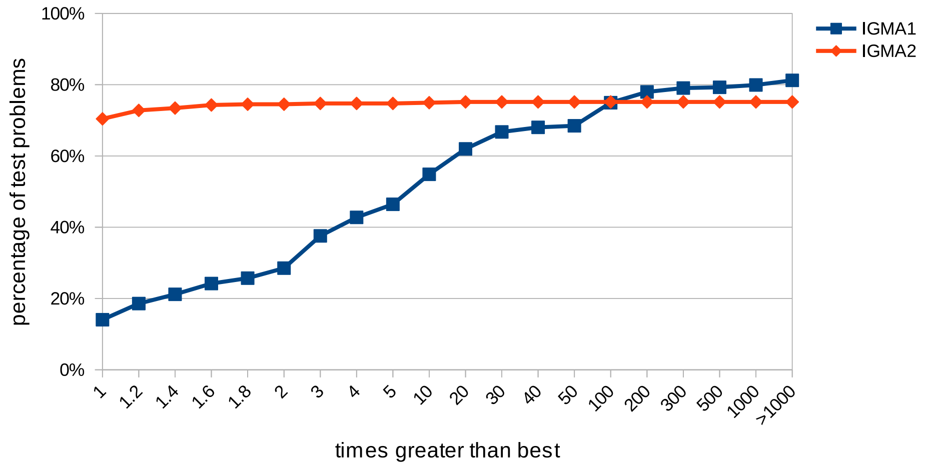


Fig. 3 Relative comparison between heuristics for nonconvex problems: **a** best solutions found by the heuristics; **b** computational time spent by the heuristics

other heuristics with lower computational cost. The results obtained from our computational experiments indicate that IGMA1 is a good option for a feasibility heuristic, when the user's focus is to find a good feasible solution, for example, in situations where exact algorithms are not able to provide any solution in a reasonable time. IGMA2, on the other hand, may be the best option when computational time is the most crucial factor in finding the solution, for example, when it is desired to use a feasibility heuristic as a subprocedure within exact algorithms.

IGMA1 and IGMA2 presented good results on both convex and nonconvex problems. IGMA1 also has the interesting characteristic of being able to be used as an exact algorithm, i.e., to effectively find the optimal solution of the problem addressed. Although preliminary results have shown that it is not currently competitive with other exact algorithms from the

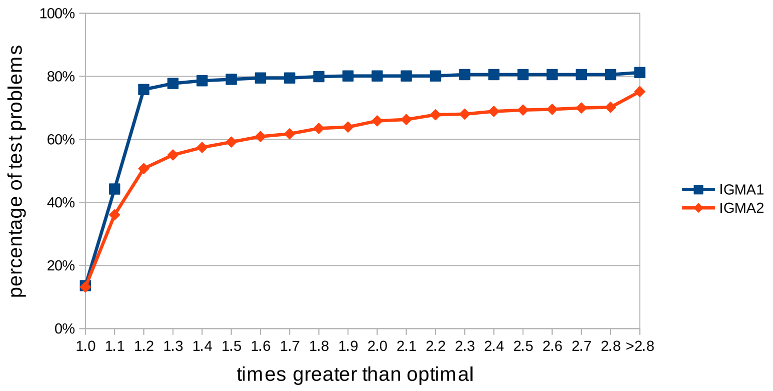


Fig. 4 Comparison between solutions found by the heuristics with known optimal solutions for nonconvex problems

literature, we point out that, as nonlinear programming solvers evolve, its performance can be improved.

We emphasize that good MINLP solvers must implement a diverse set of feasibility heuristics to effectively support the performance of exact algorithms in a wide range of problems. In this context, our work brings the contribution of two new feasibility heuristics to help improving the performance of these solvers. We point out that implementations of IGMA1 and IGMA2, as well as of the other heuristics considered in the computational experiments, will be available on the open source MINLP solver `Muriqui`, which is under development and will soon be available to the academic and technical community. All these heuristics will be used to support the algorithms described in [18,33].

References

1. Belotti, P.: Design of telecommunication networks with shared protection. Available from CyberInfrastructure for MINLP at: www.minlp.org/library/problem/index.php?i=51 (2009). Accessed 14 Oct 2016
2. Belotti, P., Berthold, T.: Three ideas for a feasibility pump for nonconvex minlp. Optim. Lett. **11**(1), 3–15 (2017)
3. Bertacco, L., Fischetti, M., Lodi, A.: A feasibility pump heuristic for general mixed-integer problems. Discrete Optim. **4**(1), 63–76 (2007)
4. Berthold, T., Gleixner, A.M.: Undercover—a primal heuristic for minlp based on sub-mips generated by set covering. Technical Report ZIB-REPORT 09-04 (2009)
5. Berthold, T.: Heuristic algorithms in global MINLP solvers. Ph.D. thesis, Technische Universität Berlin (2014)
6. Berthold, T.: Rens. Math. Program. Comput. **6**(1), 33–54 (2014)
7. Bonami, P., Cornuéjols, G., Lodi, A., Margot, F.: A feasibility pump for mixed integer nonlinear programs. Math. Program. **119**, 331–352 (2009). https://doi.org/10.1007/s10107-008-0212-2
8. Bonami, P., Gonçalves, J.: Heuristics for convex mixed integer nonlinear programs. Comput. Optim. Appl. (2008). https://doi.org/10.1007/s10589-010-9350-6
9. Bonami, P., Kilinç, M., Linderoth, J.: Algorithms and software for convex mixed integer nonlinear programs. Technical Report 1664, Computer Sciences Department, University of Wisconsin-Madison (2009)
10. Bonami, P., Lee, J., Leyffer, S., Wächter, A.: On branching rules for convex mixed-integer nonlinear optimization. J. Exp. Algorithmics **18**, 2.6:2.1–2.6:2.31 (2013)
11. Bragalli, C., D'Ambrosio, C., Lee, J., Lodi, A., Toth, P.: On the optimal design of water distribution networks: a practical minlp approach. Optim. Eng. **13**, 219–246 (2012). https://doi.org/10.1007/s11081-011-9141-7

12. Christodoulou, M., Costoulakis, C.: Nonlinear mixed integer programming for aircraft collision avoidance in free flight. In: Proceedings of the 12th IEEE Mediterranean Electrotechnical Conference, 2004. MELECON 2004, vol. 1, pp. 327–330 (2004)
13. D'Ambrosio, C., Frangioni, A., Liberti, L., Lodi, A.: A storm of feasibility pumps for nonconvex minlp. Math. Program. **136**(2), 375–402 (2012)
14. D'Ambrosio, C., Lodi, A.: Mixed integer nonlinear programming tools: a practical overview. 4OR **9**(4), 329–349 (2011)
15. Danna, E., Rothberg, E., Le Pape, C.: Exploring relaxation induced neighborhoods to improve mip solutions. Math. Program. **102**(1), 71–90 (2005)
16. Dash Optimization. Getting Started with Xpress. http://www.fico.com/xpress. Accessed 14 Oct 2016
17. Duran, M., Grossmann, I.: An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math. Program. **36**, 307–339 (1986). https://doi.org/10.1007/BF02592064
18. Fampa, M., Lee, J., Melo, W.: On global optimization with indefinite quadratics. EURO J. Comput. Optim. (2016). https://doi.org/10.1007/s13675-016-0079-6
19. Fischetti, M., Lodi, A.: Local branching. Math. Program. **98**, 23–47 (2003). https://doi.org/10.1007/s10107-003-0395-5
20. Fletcher, R., Leyffer, S.: Solving mixed integer nonlinear programs by outer approximation. Math. Program. **66**, 327–349 (1994). https://doi.org/10.1007/BF01581153
21. Fourer, R., Gay, D.M., Kernighan, B.: AMPL: a mathematical programming language. In: Wallace, S.W. (ed.) Algorithms and Model Formulations in Mathematical Programming. Nato Asi Series F, Computer And Systems Sciences, Vol. 51. Springer, New York, NY, pp. 50–151 (1989).
22. GAMS World. Minlp library 2. https://www.gamsworld.org/minlp/minlplib2/html/ (2014). Accessed 14 Oct 2016
23. Gentilini, I.: Minlp approach for the TSPN (traveling salesman problem with neighborhoods). Available from CyberInfrastructure for MINLP at: https://www.minlp.org/library/problem/index.php?i=124 (2011). Accessed 14 Oct 2016
24. Geoffrion, A.M.: Generalized benders decomposition. J. Optim. Theory Appl. **10**, 237–260 (1972). https://doi.org/10.1007/BF00934810
25. Gopalakrishnan, A., Biegler, L.: Minlp and MPCC formulations for the cascading tanks problem. Available from CyberInfrastructure for MINLP at: https://www.minlp.org/library/problem/index.php?i=140(2011). Accessed 14 Oct 2016
26. Guillen, G., Pozo, C.: Optimization of metabolic networks in biotechnology. Available from CyberInfrastructure for MINLP at: https://www.minlp.org/library/problem/index.php?i=81 (2010). Accessed 14 Oct 2016
27. Hemmecke, R., Köppe, M., Lee, J., Weismantel, R.: Nonlinear integer programming. In: Jünger, M., Liebling, T.M., Naddef, D., Nemhauser, G.L., Pulleyblank, W.R., Reinelt, G., Rinaldi, G., Wolsey, L.A. (eds.) 50 Years of Integer Programming 1958–2008, pp. 561–618. Springer, Berlin (2010). https://doi.org/10.1007/978-3-540-68279-0_15
28. Intel Corporation: Intel C++ Compiler 16.0 User and Reference Guide. https://software.intel.com/en-us/intel-cplusplus-compiler-16.0-user-and-reference-guide. Accessed 14 Oct 2016
29. Leyffer, S., Linderoth, J., Luedtke, J., Miller, A., Munson, T.: Applications and algorithms for mixed integer nonlinear programming. J. Phys. Conf. Ser. **180**(1), 012014 (2009)
30. Liberti, L., Mladenović, N., Nannicini, G.: A recipe for finding good solutions to minlps. Math. Program. Comput. **3**, 349–390 (2011). https://doi.org/10.1007/s12532-011-0031-y
31. Liu, P., Pistikopoulos, E.N., Li, Z.: Global multi-objective optimization of a nonconvex minlp problem and its application on polygeneration energy systems design. Available from CyberInfrastructure for MINLP at: https://www.minlp.org/library/problem/index.php?i=42 (2009). Accessed 14 Oct 2016
32. López, C.O., Beasley, J.E.: A note on solving minlp's using formulation space search. Optim. Lett. **8**(3), 1167–1182 (2014)
33. Melo, W., Fampa, M., Raupp, F.: Integrating nonlinear branch-and-bound and outer approximation for convex mixed integer nonlinear programming. J. Glob. Optim. **60**(2), 373–389 (2014)
34. Mouret, S., Grossmann, I.: Crude-oil operations scheduling. Available from CyberInfrastructure for MINLP at: https://www.minlp.org/library/problem/index.php?i=117 (2010). Accessed 14 Oct 2016
35. Murray, W., Ng, K.-M.: An algorithm for nonlinear optimization problems with binary variables. Comput. Optim. Appl. **47**, 257–288 (2010). https://doi.org/10.1007/s10589-008-9218-1
36. Nannicini, G., Belotti, P., Liberti, L.: A local branching heuristic for MINLPs. ArXiv e-prints (2008)
37. Quesada, I., Grossmann, I.E.: An LP/NLP based branch and bound algorithm for convex minlp optimization problems. Comput. Chem. Eng. **16**(10–11), 937–947 (1992)
38. Raghavachari, M.: On connections between zero-one integer programming and concave programming under linear constraints. Oper. Res. **17**(4), 680–684 (1969)

39. Savelsbergh, M.W.P.: Preprocessing and probing techniques for mixed integer programming problems. ORSA J. Comput. **6**(4), 445–454 (1994)
40. Science Technology Facilities Council: HSL: A collection of Fortran codes for large scale scientific computation. http://www.hsl.rl.ac.uk/. Accessed 14 Oct 2016
41. Trespalacios, F., Grossmann, I.E.: Review of mixed-integer nonlinear and generalized disjunctive programming methods. Chem. Ing. Tech. **86**(7), 991–1012 (2014)
42. Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program. **106**, 25–57 (2006). https://doi.org/10.1007/s10107-004-0559-y
43. You, F., Grossmann, I.E.: Mixed-integer nonlinear programming models and algorithms for large-scale supply chain design with stochastic inventory management. Indus. Eng. Chem. Res. **47**(20), 7802–7817 (2008)