Computers and Chemical Engineering 48 (2013) 218–233

Contents lists available at SciVerse ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

A comparative assessment of linearization methods for bilinear models

Maria Analia Rodriguez, Aldo Vecchietti*

INGAR (CONICET-UTN), Avellaneda 3657, Santa Fe 3000, Argentina

ARTICLE INFO

Article history:
Received 22 November 2011
Received in revised form
11 September 2012
Accepted 15 September 2012
Available online 3 October 2012

Keywords:
MINLP
Linearization methods
Bilinear terms
Discrete decisions

ABSTRACT

In this article, optimization problems with bilinear constraints involving one discrete variable are studied. Several industrial problems present bilinear non-convex constraints which are difficult to solve to global optimality. For this purpose models must be reformulated what in general terms increases the problem size. This article proposes two disjunctive transformation techniques which are compared to other approaches presented in the literature. An analysis is made comparing qualitative and quantitative characteristics of the methods employed. In order to implement proposed transformations, three industrial cases are studied: trim-loss in a paper mill, cutting stock in the production of carton board boxes and the purchase, inventory and delivery optimization problem. All of them are reformulated and solved using the strategies included in the paper. Several instances of each problem are evaluated and their results are analyzed comparing performance of the different methods.

1. Introduction

Several typical industrial problems such as planning, scheduling, distribution, inventory, process control and design involve discrete decisions, continuous variables and bilinear terms. In general, the original non-linear non-convex problem presents several local optima that makes obtaining the global solution a really difficult task. One attempt to overcome this issue is to use global optimization algorithms. However, as the problem size increases, the resolution time rises and the problem could become intractable. This context promotes a permanent interest in finding new approaches to solve it. One way to address it is to reformulate the non-convex problem as a linear or convex one. This transformation methodology, which allows reaching global optimality, presents two new issues that must be taken into account. In the first place, the new formulation increases the problem size in terms of variables and constraints, and as a consequence, the solution of the transformed model could be a time consuming task. In the second, the tightness of the representation employed also constrains the success of the applied method.

Several reformulations from in the literature tackle bilinear terms. These can involve continuous or discrete variables and according to that characteristic, the strategies used to transform the problem are different. The simplest case includes the product of two binary variables. Glover and Woolsey (1974) proposed a reformulation technique that converted a 0–1 polynomial programming problems into a 0–1 linear programming problems by replacing cross-product terms by continuous variables. Glover (1975) analyzed the problem of minimizing $\sum_{i,j} y_i a_{ij} y_j$ where $y_i, y_j \in \{0, 1\}$ and all the remaining constraints are linear. In this problem, the nonlinear objective function is replaced by $\sum_i z_i$ and $4 \cdot i^{\max}$ constraints are added to the formulation.

In the case of a bilinear term with one continuous and one binary variable, Petersen (1971) proposed to introduce a new variable z_i, to replace each bilinear term, and $4 \cdot i^{\max}$ linear constraints, to limit the upper and lower bounds of the continuous variable. In the same direction, Psarris and Floudas (1990) applied a similar method for the same type of problems.

Adams and Sherali (1990) and Sherali and Adams (1994) introduced a method under the name of Reformulation Linearization Technique (RLT) that proposed a hierarchy of relaxations for linear and polynomial 0–1 problems. RLT was then extended to continuous non-convex programs by Sherali and Tuncbilek (1992) which generated polynomial implied constraint in a first step and then, linearized the resulting problem by a suitable variable transformation. As it was mentioned by Chang (2000), the main drawbacks of this method are exposed in three points. First, several of the implied constraints need to be generated in a linearized form and tightening its representation step by step is a long trial-and-error process taking into account the exponential generation of new constraints and variables. Additionally, RLT algorithm always requires a huge amount of bounded constraints to be generated that on many occasions are redundant. Finally, RLT process varies according to the problem under analysis so a user needs to formulate a special RLT scheme depending on the case.

* Corresponding author. Tel.: +54 342 4535568; fax: +54 342 4553439.
E-mail address: aldovec@santafe-conicet.gov.ar (A. Vecchietti).

0098-1354/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.compchemeng.2012.09.011

When both variables are continuous, most of the approaches use some reformulation based on McCormick relaxation (McCormick, 1976).

Grossmann (2002) reported an interesting review of non-linear programming techniques where topics about bilinearities are covered. Gounaris, Misener, and Floudas (2009) presented an exhaustive comparison of several piecewise-linear relaxations for the pooling problem with the aim of improving the tightness of the original McCormick relaxation and the efficiency of the optimization procedure. Liberti and Pantelides (2006) proposed an exact graph-theoretical algorithm in the reformulation of non-convex problems that can be applied whether the variables are discrete or continuous. The basic idea behind this procedure is inherited from Sherali and Adams (1986), which is to multiply the original linear constraints by the existent variables. Bilinear terms are then replaced by their equivalent variable and eliminated from the formulation. This method is different from RLT since it applies RLT-type multiplications directly to the original NLP problem (and not its convex relaxation) deriving an exact reformulation which has fewer bilinear terms and more linear constraints, called reduced RLT (RRLT). One limitation of this technique appears when there is a variable only involved in a linear constraint which is not included in a bilinear product. In this case, when the corresponding linear constraint is multiplied by one of the variables, the procedure introduces a new bilinear term. Finally, the method could not guarantee to turn the original problem into a linear one in all cases.

The main focus of this work is placed on bilinear terms with at least one integer variable. In this subject, some works from the literature aims at solving specific bilinear problems applications while others present more general transformations. With a broad purpose, Pörn, Harjunkoski, and Westerlund (1999) proposed several convex reformulations for MINLP problems. Some linear transformations presented in this article were applied to the trim-loss problem by Harjunkoski, Westerlund, and Pörn (1999). Grossmann, Voudouris, and Ghattas (1992) presented a linear reformulation which was applied for batch process design and discrete sizing structural problems. In both articles, the authors redefined the integer variable in terms of binary variables and the bilinear product as the summation of linear terms. However, they used different strategies to reformulate the non-convex term.

In this article, due to the connection between discrete decisions and disjunctive modeling, generalized disjunctive programming is used to propose transformations which differ from the ones presented in the literature. These techniques not only offer a natural way to represent the original variables but also lead to new reformulations that can also be applied to solve the same problems. In order to present a qualitative comparison, the proposed methods and the ones from Liberti and Pantelides (2006), Harjunkoski et al. (1999) and (Grossmann et al., 1992) are analyzed in terms of the number of constraints and variables added in the model. With the same purpose, plausible links and relations between these techniques are also studied. From a quantitative perspective, the computational effort to solve some typical bilinear problems is also reported and discussed.

The remaining paper is organized as follows: in Section 2 we describe different transformations applied to convert bilinear constraints into linear ones. We also explain their characteristics and analyze the reformulation's size. All these transformation techniques proposed are used to reformulate three case studies. Results obtained with those relaxations are presented in Section 3 and an evaluation is carried out based on their efficiency to reach the solution. Final conclusions and discussions are presented in Section 4. Supplementary data includes Appendices A, B and C present the bilinear constraints reformulations for each case study.

2. Bilinear term reformulations

In this section all methods used to convert the bilinear terms into linear are presented.

2.1. Harjunkoski et al.'s reformulation

The first two linearization methods described were proposed by Harjunkoski et al. (1999). The authors applied these reformulations for the case of the trim loss problem. However, they can also be used for any case where at least one of the variables of the bilinear term is discrete.

Consider the bilinear term presented in Eq. (1):

$$m_i \cdot n_j \le D_{ij} \quad \forall i \in I,\ \forall j \in J \tag{1}$$

where m_i is an integer variable ($0 \le m_i \le m^{up}$), n_j is a positive variable ($0 \le n_j \le n^{up}$) and D_{ij} is a parameter. Note that one could also consider a different lower bound for the variables with little change in the formulation (for more details concerning this point see Pörn et al., 1999). The bilinear term is transformed using binary (β_{ik}) and slack (n_{ijk}) variables.

The first strategy, called HK1, presents an intuitive form to model the discrete variable. This variable is defined in Eq. (2) as the summation of the K possible values that m_i can take multiplied by the binary variable β_{ik}. Only one binary variable can be nonzero (Eq. (3)) in order to guarantee that only one value is given to variable m_i.

$$m_i = \sum_k k \cdot \beta_{ik} \quad \forall i \in I \tag{2}$$

$$\sum_k \beta_{ik} \le 1 \quad \text{where } K = \{1, 2, \ldots, m^{up}\}\ \forall i \in I \tag{3}$$

In the second transformation, named HK2, the discrete variable is defined using a two-based formulation as shown in Eq. (4).

$$m_i = \sum_k 2^{k-1} \cdot \beta_{ik} \quad \text{where } K = \{1, 2, \ldots, \lfloor \log_2(m^{up}) \rfloor + 1\ \forall i \in I \tag{4}$$

Eq. (4) defines the integer variable m_i as the summation of K terms. In this case, more than one binary variable (β_{ik}) can assume a positive value. The combinations of positive terms will form different possible values of m_i. As a consequence, this formulation leads to less number of 0–1 variables than the previous one.

Whether HK1 or HK2 representation is used to define variable m_i, some other constraints, Eqs. (5) and (6), must be added to replace variable n_j. For this purpose, a positive slack variable ($n_{ijk} \ge 0$) is introduced.

$$n_{ijk} - n_j \le 0 \quad \forall i \in I, \quad \forall j \in J, \quad \forall k \in K \tag{5}$$

Eq. (5) forces the value of n_{ijk} to zero when n_j is equal to zero.

$$-n_{ijk} + n_j - n^{up}(1 - \beta_{ik}) \le 0 \quad \forall i \in I, \quad \forall j \in J, \quad \forall k \in K \tag{6}$$

Note that Eq. (6) has the form of a big-M constraint. When β_{ik} is one, this constraint must be satisfied and then n_{ijk} will be equal to n_j since Eq. (5) makes n_{ijk} lower than n_j and Eq. (6) makes it greater than n_j. On the contrary, if β_{ik} is zero Eq. (6) is relaxed.

$$n_{ijk} - n^{up} \cdot \beta_{ik} \le 0 \quad \forall i \in I, \quad \forall j \in J, \quad \forall k \in K \tag{7}$$

Finally, Eq. (7) is an upper bound for n_{ijk}. When β_{ik} is one, n_{ijk} is lower than or equal to n^{up} but when β_{ik} is equal to zero, n_{ijk} is also zero.

At this point, we can analyze the constraints proposed by Petersen (1971) for the bilinear product of a binary variable by a positive continuous one. Consider the following bilinear product given by Eq. (8):

$$y_i \cdot n_j \quad \forall i \in I, \quad \forall j \in J \tag{8}$$

where $y_i = \{0,1\}$ and n_j continuous such that $0 \le n_j \le n^{up}$.A new variable n_{ij} is introduced in the formulation to replace the bilinear term and the following inequalities are added:

$$y_i \cdot n^{lo} \le n_{ij} \le y_i \cdot n^{up} \quad \forall i \in I, \quad \forall j \in J \tag{9}$$

$$n_j - n^{up}(1-y_i) \le n_{ij} \le n_j - n^{lo}(1-y_i) \quad \forall i \in I, \quad \forall j \in J \tag{10}$$

Since in this case, $n^{lo} = 0$, variable n_{ij} is positive ($n_{ij} \ge 0$) and lower bound constraint can be omitted. Also, if we disaggregate inequalities in Eq. (10) and change the order of the terms we get inequalities (11) to (13):

$$n_{ij} - n_j \le 0 \quad \forall i \in I, \quad \forall j \in J \tag{11}$$

$$-n_{ij} + n_j - n^{up}(1-y_i) \le 0 \quad \forall i \in I, \quad \forall j \in J \tag{12}$$

$$n_{ij} - y_i \cdot n^{up} \le 0 \quad \forall i \in I, \quad \forall j \in J \tag{13}$$

It is interesting to notice that there is a connection between Petersen's and Harjunkoski's methods. In fact, Eqs. (11)–(13) are quite similar to Eqs. (5)–(7). The difference between Petersen's and Harjunkoski's methods is given by the discrete variable in the bilinear term. Petersen's only introduces these 3 inequalities for each bilinear term ($3 \times i \times j$) while Harjunkoski's has to add these three for each new binary variable ($3 \times i \times j \times k$) to represent the discrete variable of the bilinear term. In other words, Harjunkoski's strategy disaggregates the original bilinear term into several bilinear terms formed by one binary and one continuous variable and then it applies Petersen's transformation to linearize these bilinear terms.

Then, for the first strategy, HK1, Eq. (1) can be rewritten as (14):

$$\sum_k k \cdot n_{ijk} \le D_{ij} \quad \forall i \in I, \quad \forall j \in J \tag{14}$$

For the second approach, HK2, Eq. (1) is replaced by (15):

$$\sum_k 2^{k-1} \cdot n_{ijk} \le D_{ij} \quad \forall i \in I, \quad \forall j \in J \tag{15}$$

Then, HK1 is given by (2), (3), (5)–(7), (14). This alternative introduces in the original problem:

- $2 \times i + 3 \times m^{up} \times i \times j$ constraints;
- $m^{up} \times i$ binary variables;
- $m^{up} \times i \times j$ continuous variables.

HK2 strategy is given by Eqs. (4)–(7), (15). This model adds to the original formulation:

- $i + 3 \times (\lfloor \log_2(m^{up}) \rfloor + 1) \times i \times j$ constraints;
- $(\lfloor \log_2(m^{up}) \rfloor + 1) \times i$ binary variables;
- $(\lfloor \log_2(m^{up}) \rfloor) \times i \times j$ continuous variables.

On the one side, it is remarkable that the latter approach leads to a smaller formulation because the 2-based representation needs less number of variables and constraints to transform the original model. On the other side, this formulation also allows more number of values for variable m_i. It increases in 2^{k-1} the possible values for the discrete variable when set K is augmented in one unit. Then, according to the maximum value allowed for $+(m^{up})$, 2-based reformulation (HK2) could include more possible values for m_i than really required. In those cases, integer cuts can be introduced in order to eliminate the unwanted values for the variable. Using any of these transformations, the bilinear term becomes linear and the relaxed problem can be solved to global optimality.

2.2. Voudouris, Ghattas and Grossmann's reformulation

The method studied in this section was derived from the work presented by (Grossmann et al., 1992). In this article, the authors proposed a reformulation which is applied to two examples. They showed that when one of the variables of the bilinear problem is restricted to discrete values the original nonlinear model can be reformulated as a MILP. In fact, some other authors also applied this technique for batch design applications (Moro & Pinto, 2004; Moreno & Montagna, 2009; Ierapetritou & Pistikopoulos, 1996). For further reference in this paper this method will be called VGG (Voudouris–Ghattas–Grossmann) method.

VGG redefines variable m_i in Eq. (16) in terms of binary variables β_{ik} and parameter k which indicates the possible values for this discrete variable. This method also introduces continuous variables, $n_{ijk} \ge 0$, to represent the original continuous variable n_j. Upper bound for n_{ijk} is determined in Eq. (18). Finally, variable n_j is calculated in (19) and the original bilinear constraint (1) is replaced by (20).

Note that VGG method shows some similarities and differences with respect to HK1. First, Eqs. (2) and (3) are the same as Eqs. (16) and (17) and Eq. (7) is equal to (18). The constraint that replaces the original bilinear restriction is also the same (Eqs. (14) and (20)). The difference is that VGG uses Eq. (19) instead of Eqs. (5) and (6) of HK1 which leads to a much more compact representation.

$$m_i = \sum_k k \cdot \beta_{ik} \quad \forall i \in I \tag{16}$$

$$\sum_k \beta_{ik} \le 1 \quad \text{where } K = \{1, 2, \ldots, m^{up}\} \forall i \in I \tag{17}$$

$$n_{ijk} - n^{up} \cdot \beta_{ik} \le 0 \quad \forall i \in I, \quad \forall j \in J, \quad \forall k \in K \tag{18}$$

$$n_j = \sum_k n_{ijk} \quad \forall i \in I, \forall j \in J \tag{19}$$

$$\sum_k k \cdot n_{ijk} \le D_{ij} \quad \forall i \in I, \quad \forall j \in J \tag{20}$$

This reformulation is formed by Eqs. (16)–(20) which presents a reduced number of equations compared to HK1 and HK2. Note that Eq. (20) replaces (1). In fact, this method introduces:

- $2 \times i + i \times j + m^{up} \times i \times j$ constraints;
- $m^{up} \times i$ binary variables;
- $m^{up} \times i \times j$ continuous variables.

Note that this formulation presents some similarities to the convex hull relaxation applied to disjunctive models. There is a disaggregation of the variables and upper bound constraints. We will show later some interesting conclusions connected with one of the disjunctive approach proposed in this article.

2.3. Liberti and Pantelides's method

The next method analyzed in this section was proposed by Liberti and Pantelides (2006) and it was called RRLT (Reduced Reformulation Linearization Technique). The authors reformulate non-linear problems in such a way that some of the nonlinear constraints are replaced by linear ones.

The basic idea of this method was first proposed by Sherali and Adams (1986). They create new linear constraints by multiplying existing linear constraints by problem variables under the name “reformulation-linearization technique” (RLT). The RLT uses these linear constraints to provide a lower bound to bilinear programming problems. The maximum possible number of new linear constraints is obtained by multiplying all linear constraints by all problem variables. As a consequence, the RLT could lead to excessive computational complexity in large problems. Extending these ideas, Liberti and Pantelides proposed an algorithm that identifies which set of multiplications are in fact beneficial.

Given the bilinear restriction in Eq. (1) and other linear constraints such as Eq. (21), the method multiplies the linear constraint by the other variable participating in the bilinear product, which leads to Eq. (22).

$$a_i \cdot m_i + b_i = 0 \tag{21}$$

$$a_i \cdot m_i \cdot n_j + b_i \cdot n_j = 0 \tag{22}$$

Then, we can introduce a new variable s_{ij} which replaces the bilinear product in (22):

$$a_i \cdot s_{ij} + b_i \cdot n_j = 0 \tag{23}$$

The linear constraint (23) is redundant with respect to the original constraints. Certainly, it can be used to replace the bilinear term $m_i \cdot n_j$ in the original problem without affecting the feasible region of the problem. Then, Eq. (1) can be rewritten as Eq. (24) and Eq. (21) can be eliminated and replaced by (23):

$$s_{ij} \le D_{ij} \tag{24}$$

A limitation of this technique is that all bilinear products of the type $m_i \cdot n_j$ have to exist in the problem before the linear constraint (like Eq. (23)) can be created. Otherwise, new bilinear terms would be created by this procedure. Liberti and Pantelides have concluded that such reduction constraint is valid if the substitution leads to a reduction in the overall number of bilinear constrains in comparison with the original problem.

Since the aim of this work is to compare linear transformation techniques, the application of this method will be restricted to the cases where linear formulation can be obtained. In other cases, the method will not be considered as a valid technique because it would not guarantee global optimality.

We have also studied the application of this technique to bilinear products of two discrete variables. Suppose the bilinear constraint in Eq. (25):

$$m \cdot n \le D \tag{25}$$

where both m and n are discrete variables such that:

$$0 \le m \le m^{up} \tag{26}$$

$$0 \le n \le n^{up} \tag{27}$$

Consider that the original problem also includes the following linear constraint:

$$a \cdot m + b \le 0 \tag{28}$$

when Eq. (28) is multiplied by variable n, the bilinear term generated is replaced by variable s in (29). Then, the original bilinear constraint Eq. (25) is replaced by Eq. (30).

$$a \cdot s + b \cdot n \le 0 \tag{29}$$

$$s \le D \tag{30}$$

One interesting point is that variable m will be also eliminated from the formulation. Then, even if the new variable s and n are considered discrete variables, the discrete nature of the original variable m is missed because the new model cannot guarantee that s would be exact multiple of variable n. Although the reformulation is linear, it offers a lower bound of the original problem (in the case of minimization). This is a limitation that the other methods do not present since the original variables remain in the model.

No generalization can be concluded regarding the number of variables introduced because it depends on the structure of the original problem. However, it is possible that if the problem can be reformulated as a linear model the number of variables and constraints remain constant which is the best feature of the method.

2.4. Disjunctive reformulation

In this section, we propose some reformulations using a disjunctive approach. Even though the main idea of these techniques also relies on the use of binary variables to represent the discrete ones, the way of modeling through disjunction varies from the previous methods. First, the disjunctive reformulation is more expressive and clearly addresses the integer nature of the original variables. Second, linear reformulations, applying Convex Hull and Big M relaxations, lead to different MILP models from the ones proposed by the previous authors. Two transformations are proposed using a disjunctive representation.

2.4.1. First disjunctive approach

In the first method, each term in the disjunction represents one possible value of variable m_i as shown in Eq. (31). Consequently, the disjunction has as many terms as the number of possible values of m_i. Note that each disjunction's term involve a positive variable $s_{ij} \ge 0$, which represents the bilinear product ($m_i \cdot n_j$).

$$\bigvee_{k} \begin{bmatrix} \beta_{ik} \\ m_i = k-1 \\ s_{ij} = (k-1) \cdot n_j \quad \forall j \in J \end{bmatrix} \quad \forall i \in I \tag{31}$$

In (31) $K = \{1, 2, \ldots, m^{up}+1\}$. Then, Eq. (1) can be reformulated as:

$$s_{ij} \le D_{ij} \quad \forall i \in I, \quad \forall j \in J \tag{32}$$

Note that the first equation in disjunction (31) might be not required if variable m_i is only required in the bilinear term but not in the rest of the model. As mentioned, one advantage of the disjunctive representation is its compact form where a more natural representation of the decision variables is obtained.

In order to implement this disjunctive model we present two relaxations. In the first place, Big-M reformulation is given by Eqs. (32)–(37).

$$\sum_{k} \beta_{ik} = 1 \quad \text{where}\, K = \{1, 2, \ldots, m^{up}+1\} \ \forall i \in I \tag{33}$$

$$m_i \ge (k-1) - M_{ik}^{-}(1-\beta_{ik}) \quad \forall i \in I, \ \forall k \in K \tag{34}$$

$$m_i \le (k-1) + M_{ik}^{+}(1-\beta_{ik}) \quad \forall i \in I, \ \forall k \in K \tag{35}$$

$$s_{ij} \ge (k-1) \cdot n_j - S_{ijk}^{-}(1-\beta_{ik}) \quad \forall i \in I, \ \forall j \in J, \ \forall k \in K \tag{36}$$

$$s_{ij} \le (k-1) \cdot n_j + S_{ijk}^{+}(1-\beta_{ik}) \quad \forall i \in I, \ \forall j \in J, \ \forall k \in K \tag{37}$$

A tight Big-M reformulation could be obtained with appropriate definition of parameters $M_{ik}^{-}, M_{ik}^{+}, S_{ijk}^{-}$ and S_{ijk}^{+}. Analyzing the possible values for variables m and s, these parameters are determined as follows:

$$M_{ik}^{-} = (k-1) \quad \forall i \in I, \ \forall k \in K \tag{38}$$

$$M_{ik}^{+} = m^{up} - (k-1) \quad \forall i \in I, \ \forall k \in K \tag{39}$$

$$S_{ijk}^{-} = (k-1) \cdot n^{up} \quad \forall i \in I, \ \forall j \in J, \ \forall k \in K \tag{40}$$

$$S_{ijk}^{+} = (m^{up} - (k-1)) \cdot n^{up} \quad \forall i \in I, \ \forall j \in J, \ \forall k \in K \tag{41}$$

This relaxation introduces:

- $i + 2 \times (m^{up}+1) \times i + 2 \times (m^{up}+1) \times i \times j$ constraints;
- $(m^{up}+1) \times i$ binary variables;
- $i \times j$ continuous variables.

Another strategy to reformulate disjunction (31) is given by the Convex Hull relaxation presented in Eqs. (42)–(50).

$$\sum_k \beta_{ik} = 1 \quad \text{where } K = \{1, 2, \ldots, m^{up} + 1\}\, \forall i \in I \tag{42}$$

$$m_{ik} = (k-1) \cdot \beta_{ik} \quad \forall i \in I,\ \forall k \in K \tag{43}$$

$$s_{ijk} = (k-1) \cdot n_{ijk} \quad \forall i \in I, \forall j \in J, \forall k \in K \tag{44}$$

$$m_i = \sum_k m_{ik} \quad \forall i \in I \tag{45}$$

$$s_{ij} = \sum_k s_{ijk} \quad \forall i \in I, \forall j \in J \tag{46}$$

$$n_j = \sum_k n_{ijk} \quad \forall i \in I, \forall j \in J \tag{47}$$

$$0 \le m_{ik} \le m_k^{up} \cdot \beta_{ik} \quad \forall i \in I, \forall k \in K \tag{48}$$

$$0 \le s_{ijk} \le s_k^{up} \cdot \beta_{ik} \quad \forall i \in I,\ \forall j \in J,\ \forall k \in K \tag{49}$$

$$0 \le n_{ijk} \le n_k^{up} \cdot \beta_{ik} \quad \forall i \in I, \forall j \in J, \forall k \in K \tag{50}$$

Note that Eqs. (43) and (44) redefines the constraints of the disjunctive terms using new continuous variables, Eqs. (45)–(47) disaggregates the original variables in terms of the new ones introduced and Eqs. (48)–(50) give lower and upper bounds to new variables.

Since Eqs. (43) and (44) already give the corresponding values to m_{ik} and s_{ijk}, the upper bounds of Eqs. (48) and (49) can be omitted. In this reformulation we can reduce the number of variables and constraints if we relax the equality of Eq. (42) and rewrite the original variable m_i and s_{ij} in terms of the right hand side of Eqs. (43) and (44), respectively. Then, the new linear model is determined by Eqs. (51)–(55).

$$\sum_k \beta_{ik} \le 1 \quad \text{where } K = \{1, 2, \ldots, m^{up}\}\, \forall i \in I \tag{51}$$

$$m_i = \sum_k k \cdot \beta_{ik} \quad \forall i \in I \tag{52}$$

$$s_{ij} = \sum_k k \cdot n_{ijk} \quad \forall i \in I, \forall j \in J \tag{53}$$

$$n_j = \sum_k n_{ijk} \quad \forall i \in I, \forall j \in J \tag{54}$$

$$0 \le n_{ijk} \le n^{up} \cdot \beta_{ik} \quad \forall i \in I, \forall j \in J, \forall k \in K \tag{55}$$

If we further rewrite this formulation, new variable s_{ij} is in fact not required in the formulation. We can then replace Eq. (32) by (56) and eliminate Eqs. (53).

$$\sum_k k \cdot n_{ijk} \le D_{ij} \quad \forall i \in I, \forall j \in J \tag{56}$$

Then, this Convex Hull reformulation, called DR1-CH, is given by Eqs. (51)–(52), (54)–(56), adding to the original formulation:

- $2 \times i + i \times j + m^{up} \times i \times j$ constraints;
- $m^{up} \times i$ binary variables;
- $m^{up} \times i \times j$ continuous variables.

Comparing this reformulation, called DR1-CH, to the one obtained applying the VGG method we can conclude the later is the convex hull relaxation of the disjunctive representation given in Eq. (31). This conclusion was not self-evident when writing the disjunction to replace the bilinear term, because a mathematical simplification was made to arrive at the final form of DR1-CH, equivalent to VGG. This is a very interesting result of this study since Voudouris, Ghattas and Grossmann's approach was extensively used to solve batch design applications with excellent results. The strength of these methods is given by the facts that: the Convex Hull reformulation gives tight bounds to the relaxed problem and, in this case, the reformulation provides a compact model because many superfluous variables and constraints were eliminated from its original form.

2.4.2. Second disjunctive approach

In the second technique a two terms disjunction is used instead of the several terms one (Eq. (31)) applied in the first representation to calculate the value of m_i. Now, the discrete variable m_i is calculated outside the disjunctions as the summation of K terms, defined by the positive variable m_{ik}, given in each disjunction (i,k). In this case, if the Boolean variable β_{ik} is true, a positive value is assigned to m_{ik}, otherwise m_{ik} is zero. This new formulation is given by Eqs. (57)–(59).

$$\begin{bmatrix} \beta_{ik} \\ m_{ik} = 2^{k-1} \\ s_{ijk} = 2^{k-1} \cdot n_j \quad \forall j \in J \end{bmatrix} \vee \begin{bmatrix} \neg\beta_{ik} \\ m_{ik} = 0 \\ s_{ijk} = 0 \quad \forall j \in J \end{bmatrix} \quad \forall i \in I, \forall k \in K \tag{57}$$

where $k = \{1, 2, \ldots, \lfloor \log_2(m^{up}) \rfloor\}$.

Variables m_{ik} and s_{ijk} are added to the original model to represent the integer variable m_i and the bilinear term, respectively. The definition of m_i is now written as shown in (58).

$$m_i = \sum_k m_{ik} \quad \forall i \in I \tag{58}$$

Eq. (1) is now rewritten as (59):

$$\sum_k s_{ijk} \le D_{ij} \quad \forall i \in I, \forall j \in J \tag{59}$$

Note that if variable m_i is only used in Eq. (1), then there is no need to define m_i in the disjunction (57) and constraint (58) can be omitted. When the number of possible values of variable m_i is large, this representation could be more attractive than the first one since it requires less number of Boolean variables due to the 2-based representation of the discrete variable. As it was also pointed out for the HK2 method, the disadvantage is that it could include more possible values for m_i than really required.

In order to implement this model as a MILP, this disjunctive representation is also reformulated using Big-M and Convex Hull relaxations. The first one, called DR2-BM, is presented in Eqs. (60)–(67).

$$m_{ik} \ge 2^{k-1} - M1_{ik}^{-} \cdot (1 - \beta_{ik}) \quad \forall i \in I, \forall k \in K \tag{60}$$

$$m_{ik} \le 2^{k-1} + M1_{ik}^{+} \cdot (1 - \beta_{ik}) \quad \forall i \in I,\ \forall k \in K \tag{61}$$

$$s_{ijk} \ge 2^{k-1} \cdot n_j - S1_{ijk}^{-} \cdot (1 - \beta_{ik}) \quad \forall i \in I,\ \forall j \in J,\ \forall k \in K \tag{62}$$

$$s_{ijk} \le 2^{k-1} \cdot n_j + S1_{ijk}^{+} \cdot (1 - \beta_{ik}) \quad \forall i \in I,\ \forall j \in J,\ \forall k \in K \tag{63}$$

$$m_{ik} \ge -M2_{ik}^{-} \cdot \beta_{ik} \quad \forall i \in I,\ \forall k \in K \tag{64}$$

$$m_{ik} \le M2_{ik}^{+} \cdot \beta_{ik} \quad \forall i \in I,\ \forall k \in K \tag{65}$$

$$s_{ijk} \ge -S2_{ijk}^{-} \cdot \beta_{ik} \quad \forall i \in I,\ \forall j \in J,\ \forall k \in K \tag{66}$$

$$s_{ijk} \le S2_{ijk}^{+} \cdot \beta_{ik} \quad \forall i \in I,\ \forall j \in J,\ \forall k \in K \tag{67}$$

Note that Eqs. (60)–(63) represent the first term of disjunction (57) while Eqs. (64)–(67) redefine the second term.

As it was shown in the first disjunctive method, suitable values of Big-M parameters lead to tight representations of the original problem. Then, for variable m_{ik}, appropriate values for these parameters are calculated in (68) to (71).

$$M1^-_{ik} = 2^{k-1} \quad \forall i \in I,\ \forall k \in K \tag{68}$$

$$M1^+_{ik} = 0 \quad \forall i \in I,\ \forall k \in K \tag{69}$$

$$M2^-_{ik} = 0 \quad \forall i \in I,\ \forall k \in K \tag{70}$$

$$M2^+_{ik} = 2^{k-1} \quad \forall i \in I,\ \forall k \in K \tag{71}$$

If we replace these parameters in the corresponding equations we obtain the following. Note that $m_{ik} \geq 0$ is omitted because the variable is non-negative by definition.

$$m_{ik} \geq 2^{k-1} - 2^{k-1} \cdot (1 - \beta_{ik}) \quad \forall i \in I,\ \forall k \in K \tag{72}$$

$$m_{ik} \leq 2^{k-1} \quad \forall i \in I,\ \forall k \in K \tag{73}$$

$$m_{ik} \leq 2^{k-1} \cdot \beta_k \quad \forall i \in I,\ \forall k \in K \tag{74}$$

Note that Eqs. (73) is the upper bound for variable m_{ik}. Eq. (72) is activated when $\beta_k = 1$ and makes m_{ik} greater than or equal to 2^{k-1}. On the contrary, Eq. (74) applies when $\beta_k = 0$ and forces m_{ik} to take its lower bound (zero) in that case. In fact, Eqs. (73) and (74) can be eliminated from the formulation since bounds are already included in the Big-M constraints (72) and (74). Then, if we further rewrite Eq. (72) we get the following:

$$m_{ik} \geq 2^{k-1}(1 - (1 - \beta_{ik})) \quad \forall i \in I,\ \forall k \in K \tag{75}$$

Then, Eq. (80) is finally given by (76).

$$m_{ik} \geq 2^{k-1}\beta_{ik} \quad \forall i \in I,\ \forall k \in K \tag{76}$$

Now, we analyze Big-M parameters for variable s_{ijk}.

$$S1^-_{ijk} = 2^{k-1} \cdot n^{up}_j \quad \forall i \in I,\ \forall j \in J,\ \forall k \in K \tag{77}$$

$$S1^+_{ijk} = 0 \quad \forall i \in I,\ \forall j \in J,\ \forall k \in K \tag{78}$$

$$S2^-_{ijk} = 0 \quad \forall i \in I,\ \forall j \in J,\ \forall k \in K \tag{79}$$

$$S2^+_{ijk} = 2^{k-1} \cdot n^{up}_j \quad \forall i \in I,\ \forall j \in J,\ \forall k \in K \tag{80}$$

The corresponding inequalities are then transformed using Big-M parameters of Eqs. (77)–(80). Note that $s_{ijk} \geq 0$ is omitted because the variable is non-negative by definition.

$$s_{ijk} \geq 2^{k-1} \cdot n_j - (2^{k-1} \cdot n^{up}_j) \cdot (1 - \beta_{ik}) \quad \forall i \in I,\ \forall j \in J,\ \forall k \in K \tag{81}$$

$$s_{ijk} \leq 2^{k-1} \cdot n_j \quad \forall i \in I,\ \forall j \in J,\ \forall k \in K \tag{82}$$

$$s_{ijk} \leq (2^{k-1} \cdot n^{up}_j) \cdot \beta_{ik} \quad \forall i \in I,\ \forall j \in J,\ \forall k \in K \tag{83}$$

Note that in this case Eq. (82) cannot be omitted since it offers tighter bounds than (83), when it is relaxed.

In summary, when we transform the second disjunctive reformulation into its Big-M representation, called DR2-BM, Eqs. (58)–(59), (74), (76), (81)–(83) are used.

This model adds:

- $i + 2 \times (\lfloor \log_2(m^{up}) \rfloor + 1) \times i + 3 \times (\lfloor \log_2(m^{up}) \rfloor + 1) \times i \times j$ constraints;
- $(\lfloor \log_2(m^{up}) \rfloor + 1) \times i$ binary variables;
- $(\lfloor \log_2(m^{up}) \rfloor + 1) \times i + (\lfloor \log_2(m^{up}) \rfloor + 1) \times i \times j$ continuous variables.

The second approach proposed to relax this 2-based disjunctive representation is given by the Convex Hull relaxation in Eqs. (84)–(96).

$$m_{ik} = m_{ik1} + m_{ik2} \quad \forall i \in I,\ \forall k \in K \tag{84}$$

$$s_{ijk} = s_{ijk1} + s_{ijk2} \quad \forall i \in I,\ \forall j \in J,\ \forall k \in K \tag{85}$$

$$n_j = n_{ijk1} + n_{ijk2} \quad \forall i \in I,\ \forall j \in J,\ \forall k \in K \tag{86}$$

$$m_{ik1} = 2^{k-1} \cdot \beta_{ik} \quad \forall i \in I,\ \forall k \in K \tag{87}$$

$$m_{ik2} = 0 \cdot (1 - \beta_{ik}) \quad \forall i \in I,\ \forall k \in K \tag{88}$$

$$s_{ijk1} = 2^{k-1} \cdot n_{ijk1} \quad \forall i \in I,\ \forall j \in J,\ \forall k \in K \tag{89}$$

$$s_{ijk2} = 0 \cdot (1 - \beta_{ik}) \quad \forall i \in I,\ \forall j \in J,\ \forall k \in K \tag{90}$$

$$m_{ik1} \leq 2^{k-1} \cdot \beta_{ik} \quad \forall i \in I,\ \forall k \in K \tag{91}$$

$$m_{ik2} \leq 0 \cdot (1 - \beta_{ik}) \quad \forall i \in I,\ \forall k \in K \tag{92}$$

$$s_{ijk1} \leq 2^{k-1} \cdot n^{up}_j \cdot \beta_{ik} \quad \forall i \in I,\ \forall j \in J,\ \forall k \in K \tag{93}$$

$$s_{ijk2} \leq 0 \cdot (1 - \beta_{ik}) \quad \forall i \in I,\ \forall j \in J,\ \forall k \in K \tag{94}$$

$$n_{ijk1} \leq n^{up}_j \cdot \beta_{ik} \quad \forall i \in I,\ \forall j \in J,\ \forall k \in K \tag{95}$$

$$n_{ijk2} \leq n^{up}_j \cdot (1 - \beta_{ik}) \quad \forall i \in I,\ \forall j \in J,\ \forall k \in K \tag{96}$$

Note that Eqs. (84)–(86) redefine the original variables that participate in the disjunctive terms, Eqs. (87)–(90) rewrite the equations in terms of the new variables introduced and Eqs. (91)–(96) establish bounds for the new variables. It is also noteworthy that this formulation can be greatly simplified since m_{ik2} and s_{ijk2} are equal to zero. Then $m_{ik} = m_{ik1}$ and $s_{ijk} = s_{ijk1}$, as a consequence m_{ik1}, m_{ik2}, s_{ijk1} and s_{ijk2} are eliminated from the model. Additionally, since equation in the disjunction (57) establishes the values of variable m_{ik} and s_{ijk}, no bound constraints are necessary for these variables. Then, Eqs. (97)–(98) replace Eqs. (87) and (89) respectively and Eqs. (84)–(85), (88) and (90)–(94) are eliminated.

$$m_{ik} = 2^{k-1} \cdot \beta_{ik} \quad \forall i \in I,\ \forall k \in K \tag{97}$$

$$s_{ijk} = 2^{k-1} \cdot n_{ijk1} \quad \forall i \in I,\ \forall j \in J,\ \forall k \in K \tag{98}$$

Finally, note that we can replace m_{ik} and s_{ijk} in the model by their corresponding values given by (97) and (98). Then this formulation, called DR2-CH, is given by (99)–(103).

$$m_i = \sum_k 2^{k-1} \cdot \beta_{ik} \quad \forall i \in I \tag{99}$$

$$\sum_k 2^{k-1} \cdot n_{ijk1} \leq D_{ij} \quad \forall i \in I,\ \forall j \in J \tag{100}$$

$$n_j = n_{ijk1} + n_{ijk2} \quad \forall i \in I,\ \forall j \in J,\ \forall k \in K \tag{101}$$

$$n_{ijk1} \leq n^{up}_j \cdot \beta_{ik} \quad \forall i \in I,\ \forall j \in J,\ \forall k \in K \tag{102}$$

$$n_{ijk2} \leq n^{up}_j \cdot (1 - \beta_{ik}) \quad \forall i \in I,\ \forall j \in J,\ \forall k \in K \tag{103}$$

Again, the Convex Hull relaxation of the 2-based disjunctive approach leads to a compact reformulation adding to the original problem:

- $i + 3 \times (\lfloor \log_2(m^{up}) \rfloor + 1) \times i \times j$ constraints;
- $(\lfloor \log_2(m^{up}) \rfloor + 1) \times i$ binary variables;
- $2 \times (\lfloor \log_2(m^{up}) \rfloor + 1) \times i \times j$ continuous variables.

Finally, in order to compare transformations' sizes, Table 1 presents an example with:

$$m^{up} = 8, \quad i = 4 \text{ and } j = 3.$$

According to this example, it can be concluded that Convex Hull and 2-based reformulations (HK2, VGG, DR1-CH, DR2-CH) present smaller sizes than the other techniques.

Table 1
Reformulations' sizes example.

	Constraints	Binary variables	Continuous variables
HK1	$2\times i+3\times m^{up}\times i\times j=296$	$m^{up}\times i=32$	$m^{up}\times i\times j=96$
HK2	$i+3\times(\lfloor\log_2(m^{up})\rfloor+1)\times i\times j=148$	$(\lfloor\log_2(m^{up})\rfloor+1)\times i=16$	$(\lfloor\log_2(m^{up})\rfloor+1)\times i\times j=48$
VGG	$2\times i+i\times j+m^{up}\times i\times j=106$	$m^{up}\times i=32$	$m^{up}\times i\times j=96$
DR1-BM	$i+2\times(m^{up}+1)\times i+2\times(m^{up}+1)\times i\times j=292$	$(m^{up}+1)\times i=36$	$i\times j=12$
DR1-CH	$2\times i+i\times j+m^{up}\times i\times j=106$	$m^{up}\times i=32$	$m^{up}\times i\times j=96$
DR2-BM	$i+2\times(\lfloor\log_2(m^{up})\rfloor+1)\times i+3\times(\lfloor\log_2(m^{up})\rfloor+1)\times i\times j=180$	$(\lfloor\log_2(m^{up})\rfloor+1)\times i=16$	$(\lfloor\log_2(m^{up})\rfloor+1)\times i+(\lfloor\log_2(m^{up})\rfloor+1)\times i\times j=64$
DR2-CH	$i+3\times(\lfloor\log_2(m^{up})\rfloor+1)\times i\times j=148$	$(\lfloor\log_2(m^{up})\rfloor+1)\times i=16$	$2\times(\lfloor\log_2(m^{up})\rfloor+1)\times i\times j=96$
RRLT	–	–	–

3. Case studies

Three test problems are presented in order to assess the linearization techniques presented in the previous section. Since these problems were taken from the literature, full formulations can be found in the corresponding articles. In order to illustrate them, a brief description and their bilinear constraints are presented in the next subsections. Bilinear transformations' can be found in the Appendices A, B and C for each case study, respectively (Supplementary Data).

3.1. Case Study 1: the trim-loss problem in the paper mill

This example was presented by Harjunkoski et al. (1999) consisting of cutting big paper rolls into several smaller ones. The objective is to minimize the trim loss and the number of cutting patterns. The main decision variables are the definition of the number of patterns j (m_j) and the number of orders i assigned to each pattern j (n_{ij}). The demand constraint is a bilinear inequality with the product of these two variables, as shown in Eq. (104). Fig. 1 depicts the cutting process indicating these variables and the bilinear term.

$$\sum_j m_j\cdot n_{ij}\geq D_i\quad\forall i\in I\tag{104}$$

The objective function in Eq. (105) which minimizes the trim loss presents the same bilinear term ($m_j\cdot n_{ij}$).

$$Min\sum_j m_j\cdot\left(B_{\max}-\sum_i n_{ij}\cdot B_i\right)+C_j\cdot y_j\tag{105}$$

In (105) B_{max} represents the roll width; B_i is another parameter indicating the paper roll widths of the customer orders; and C_j is the cost of producing a new pattern j. The binary variable y_j indicates the existence of pattern j.

3.2. Case Study 2: the cutting stock problem in the production of board boxes

In the production of board boxes several cutting patterns are defined to obtain the board sheets that will form the boxes. In order to produce the cutting patterns, first, several paper rolls are selected to form the board layers. In a second step, orders are assigned to define cutting pattern. Assignment decisions usually implicate binary variables leading to more difficult models and time-consuming solutions. When the board is cut into smaller pieces to produce the sheets, some waste of material always occurs due to the differences between the pattern's and the papers' widths. Then, the objective function is to minimize the cost of paper trim loss in the cutting process. The original non-convex problem was first developed by Rodriguez and Vecchietti (2008). In this case, two different bilinear terms appears in the original formulation given by Eqs. (106)–(109).

$$ta_{pk}\geq Wap_{pk}\cdot x_p\quad\forall p\in P,\forall k\in K\tag{106}$$

Eq. (106) determines the total area of pattern p in layer k, ta_{pk}, where Wap_{pk} is a discrete variable corresponding to the paper width assigned to layer k of pattern p, and x_p is a positive variable which indicates the pattern length. Then, in this case, the bilinear term corresponds to the product of one continuous variable by one integer one.

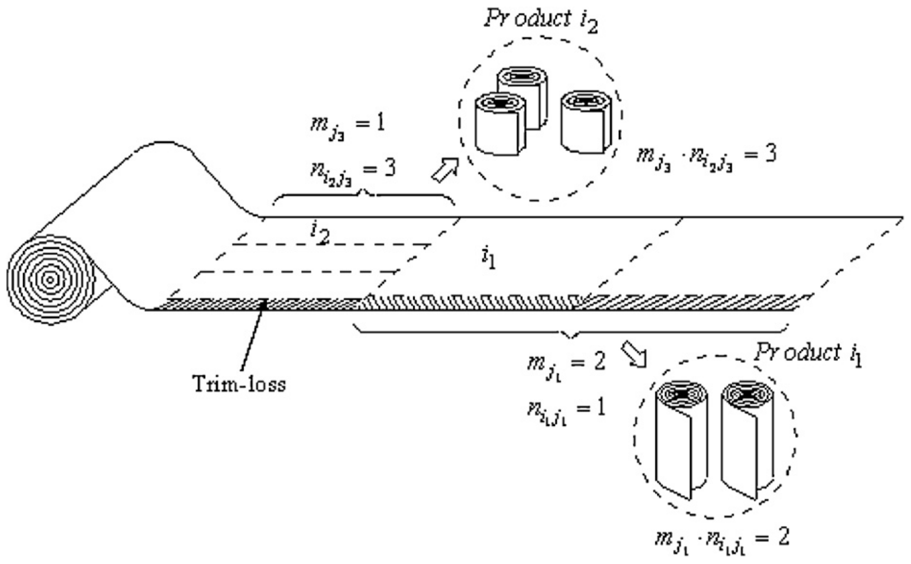
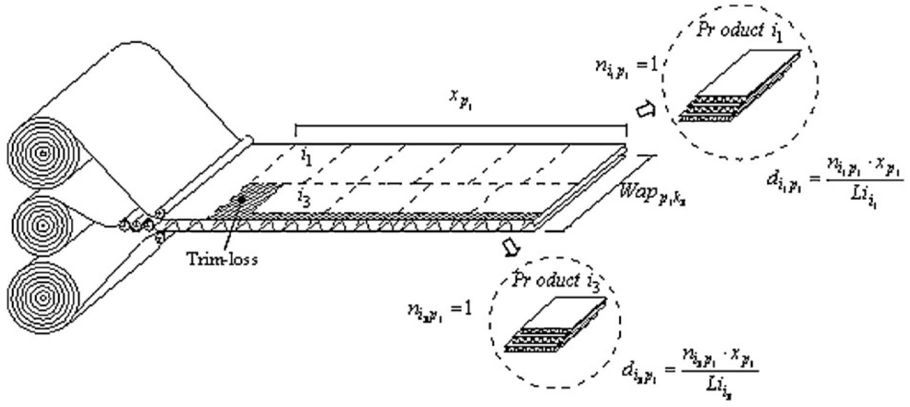


Fig. 1. Paper rolls production example, for orders i_1 and i_2 in patterns j_1 and j_3.



Pr oduct i_1
$n_{i_1p_1}=1$
x_{p_1}
i_1
i_3
$d_{i_1p_1}=\frac{n_{i_1p_1}\cdot x_{p_1}}{Li_{i_1}}$
$Wap_{p_1k_3}$
Trim-loss
Pr oduct i_3
$n_{i_3p_1}=1$
$d_{i_3p_1}=\frac{n_{i_3p_1}\cdot x_{p_1}}{Li_{i_3}}$

Fig. 2. Board sheet production example, for orders i_1 and i_3 in pattern p_1.

Another bilinear term also appears in Eq. (107) which determines a limitation according to the total amount of materials in stock. This equation establishes that paper consumption cannot be greater than the amount in stock.

$$\sum_{p\in R_{pktp}}\sum_{k\in R_{pktp}} w_{apkp}\cdot\alpha_k\cdot x_p \le S_{tpap} \quad \forall tp\in TP,\ \forall ap\in AP \tag{107}$$

It should be noted that α_k is a parameter representing the paper consumption of layer k (fluted layers consumes more than 1 m to produce 1 m length of board). Parameter $S_{tp\,ap}$ corresponds to the raw material in stock, while $w_{ap\,k\,p}$ is a binary variable which selects paper width ap of layer k in pattern p. In this inequality, Eq. (107), the bilinear product is formed by a continuous variable and a binary variable. It is noteworthy that even though the presented methods are applied to bilinear products with at least one integer variables, this would be a special case that can also be solved.

The third bilinear term in this example is given in (108).

$$oa_{ip}=n_{i\,p}\cdot Wi_i\cdot x_p \quad \forall p\in P,\ \forall i\in I \tag{108}$$

Eq. (108) calculates the area of pattern p, oa_{ip}, considering the number of order sheets i included in pattern p (n_{ip}) the width of each sheet i (Wi_i) and the length of the pattern (x_p). In this case, there is a product of the integer variable n_{ip} by the continuous variable x_p. The same bilinear term appears in Eq. (109) which is used in demand constraints.

$$d_{ip}=\frac{n_{ip}\cdot x_p}{Li_i} \quad \forall p\in P,\ \forall i\in I \tag{109}$$

Variable d_{ip} represents the number of orders i produced with pattern p, where Li_i is a parameter indicating the length of order i. This variable is used to satisfy demand and overproduction restrictions. See Fig. 2 for more details.

3.3. *Case Study 3: purchase, inventory and delivery optimization*

The main target of this problem is to optimize the inventory and delivery management integrated with material purchase decisions. Diverse materials are purchased from various manufacturers signing different contract types. This kind of commitments in general offers discounts according to the amount purchase so as a consequence they promote large purchase orders to take advantage of economy of scale. However, due to limitations in storage capacity and financial resources, the quantity purchased is distributed in several deliveries. This gives rise to non-convexities in the formulation. This problem was presented by Rodriguez and Vecchietti (2010).

The first bilinear term appears in Eq. (110). It determines that the total amount delivered of material family f from supplier j during period t must be equal to the amount bought of family f material from that supplier in that period. Note that the amount purchased of family f is in fact given by $\sum_{k\in FK_{fk}} q_{jkt}$ where k represents each material belonging to family f. This equation guarantees coherence between purchase and delivery decisions.

$$\sum_{k\in FK_{fk}} q_{jkt}=n_{jft}\cdot eoq_{jft} \quad \forall j\in J,\ \forall f\in F,\ \forall t\in T \tag{110}$$

Integer variable n_{jft} represents the number of deliveries of material family f from supplier j in period t, variable eoq_{jft} corresponds to delivery order size; while positive variable q_{jkt} indicates the quantity bought of material k from supplier j in period t. Note that the total quantity delivered of family f from supplier j during period t is given by a bilinear product of the number of deliveries (n_{jft}) by the delivery quantity (eoq_{jft}).

The second bilinear term comes from total delivery cost definition (tdc_{jft}) in Eq. (111).

$$tdc_{jft}=n_{jft}\cdot dc_{jft} \quad \forall j\in J,\ \forall f\in F,\ \forall t\in T \tag{111}$$

where dc_{jft} is a positive variable that corresponds to the unit cost of each delivery of material family f from supplier j in period t. The total delivery costs in period t for a supplier j and material family f is determined by the bilinear product shown in Eq. (111).

4. Results

The main target of this section is to complete the qualitative assessment presented in Section 2 with a quantitative analysis comparing models performance. Ten instances of each problem were reformulated as linear models using all methods described in that section. Each instance was generated varying the data and parameters as well as the number of variables and constraints. In sum, 190 models were executed. The instances sizes for the trim-loss, the cutting stock and delivery problems used in this study are presented in Tables 2, 3 and 4, respectively. They were posed in GAMS system, solved using CPLEX 11.0.1 and executed over a PC with an Intel Pentium D 2.8 GHz processor and 3.5 GB of RAM.

From Tables 2–4, it can be seen that for the three problems analyzed the model sizes are quite augmented from the original MINLP problem. In the trim-loss case, the number of constraints is incremented 16 times and the number of binary variables, 7 times on average. Moreover, the number of continuous variables moves from one to 220 on average. In the cutting stock example, the number of constraints and binary variables are duplicated from the original MINLP problem while the number of continuous variables is increased by an average of almost 6 times. Similar is the case of the third problem example in which the number of constraints is

Table 2
Trim-loss problem sizes used in this study.

	Original	HK1	HK2	VGG	DR1-BM	DR2-CH	DR2-BM
Instance 1							
Constraints	22	217	106	100	217	106	124
Integer variables	15	33	21	33	36	21	21
Continuous variables	1	67	31	67	13	79	40
Instance 2							
Constraints	43	1567	481	595	1309	481	529
Integer variables	48	126	66	126	126	66	66
Continuous variables	1	511	151	511	43	295	175
Instance 3							
Constraints	29	373	177	165	353	177	201
Integer variables	24	48	32	48	52	32	32
Continuous variables	1	117	53	117	21	101	65
Instance 4							
Constraints	30	458	214	198	418	214	238
Integer variables	28	52	36	52	56	36	36
Continuous variables	1	145	65	145	35	125	77
Instance 5							
Constraints	39	1249	564	489	1034	524	564
Integer variables	50	95	65	95	100	65	65
Continuous variables	1	406	166	406	46	326	186
Instance 6							
Constraints	39	1859	773	719	1505	773	821
Integer variables	50	126	90	126	132	90	90
Continuous variables	1	607	247	607	67	487	271
Instance 7							
Constraints	40	1400	585	545	1145	585	625
Integer variables	60	110	70	110	105	70	70
Continuous variables	1	456	186	456	51	366	206
Instance 8							
Constraints	30	458	214	198	418	214	238
Integer variables	28	52	36	52	56	36	36
Continuous variables	1	145	65	145	25	125	77
Instance 9							
Constraints	38	783	358	328	683	358	388
Integer variables	50	75	55	75	80	55	55
Continuous variables	1	251	111	251	41	216	126
Instance 10							
Constraints	39	1249	524	489	1034	524	564
Integer variables	50	95	65	95	100	65	65
Continuous variables	1	406	166	406	46	326	186

2 times larger from the non-convex problem and the number of discrete variables rises from 84 to 132. Moreover, 74 continuous variables are added to the formulation.

Since the aim of this work is to compare linearization methods, RRLT is only applied when it allows a linear reformulation. Unfortunately, due to the complexity of cutting stock and delivery problems this technique introduces new bilinear terms when eliminating the original ones from them. In the case of the trim-loss problem (Table 2), it does obtain a linear representation. However, one disadvantage in this particular case is due to the integer nature of both variables of the bilinear terms. Since a new variable s is introduced to replace the bilinear term and one of the original ones is eliminated from the formulation, the new model cannot guaranteed that s will be exact multiple of the remaining variable in the problem. Then, the original integer nature of the eliminated variable is missed. As a consequence, the reformulation obtained is a relaxation of the original problem. For this reason the RRLT method is not included in the results analysis.

Tables 5–7 show the model performances in terms of CPU execution time (in seconds) and number of iterations and nodes required to reach the solutions. In those tables it is also displayed the number of the matrix non-zero elements of each example instance. In these tables the fastest method for each instance is highlighted with a rectangle and the slowest, with a black background.

From Tables 5–7 it can be seen that the execution time varies considerably from one method to the other so a more detailed analysis is required to conclude about their performance. The CPU time consumed to reach the solution is used to determine the methods' ranking, where number 1 is assigned for the fastest method and 6, for the slowest one (note that RRLT method was excluded from the comparison due to the limitations mentioned before). Note that the same ranking can be established considering the number of iterations and nodes needed to find the solution. The mean and standard deviation are calculated taking into account the ranking values for each method in all the instances of the three problems studied. These values are presented in Table 8, assuming a normal distribution, this information is also presented in Fig. 3.

According to Fig. 3 that VGG and DR2-CH methods present almost the same ranking mean, near the value of 2. However, DR2-CH mean ranking shows a higher probability of occurrence due to a lower standard deviation, which is a desirable characteristic for a well-ranked reformulation method. In third place, HK2 method appears with a mean value of 2.87 and a standard deviation of 1.01. The following mean value is obtained by DR2-BM. However, one drawback of this method is given by a large standard deviation reporting a significant variability in the method's performance. DR1-BM method is in the fifth place with a mean value of 4.6 and a 1.28 standard deviation. In the last position, HK1 presents a mean of 5.43. Due to a low dispersion in the results of this method, there is a significant probability to obtain this mean value.

Some differences are also observed from problem to problem. Then a similar analysis is presented for each problems considering all instances evaluated. According to this information, Figs. 4 and 5 show the mean and standard deviation of the methods depending on the problem evaluated.

Table 3
Cutting stock problem sizes used in this study.

	Original	HK1	HK2	VGG	DR1-BM	DR2-CH	DR2-BM
Instance 1							
Constraints	929	2141	1977	1545	2393	1769	2425
Integer variables	256	424	368	424	480	368	368
Continuous variables	97	577	521	577	353	913	633
Instance 2							
Constraints	752	1632	1536	1176	1708	1304	1792
Integer variables	208	304	272	304	336	272	272
Continuous variables	97	433	401	433	305	625	465
Instance 3							
Constraints	721	1589	1484	1148	1708	1288	1764
Integer variables	196	301	266	301	336	266	266
Continuous variables	85	421	386	421	281	631	456
Instance 4							
Constraints	1078	2638	2278	1828	2938	2008	2758
Integer variables	300	540	420	540	600	420	420
Continuous variables	121	721	601	721	421	1021	721
Instance 5							
Constraints	1077	2818	2278	1888	3178	2008	2758
Integer variables	300	600	420	600	660	420	420
Continuous variables	121	781	601	781	421	1021	721
Instance 6							
Constraints	1364	2332	2140	1684	2636	1940	2652
Integer variables	374	464	400	464	528	400	400
Continuous variables	133	625	561	625	369	1009	689
Instance 7							
Constraints	1364	2524	2140	1748	2892	1940	2652
Integer variables	374	528	400	528	592	400	400
Continuous variables	133	689	561	689	369	1009	689
Instance 8							
Constraints	1364	2524	2140	1748	2892	1940	2652
Integer variables	374	528	400	528	592	400	400
Continuous variables	133	689	561	689	369	1009	689
Instance 9							
Constraints	878	1982	1838	1430	2174	1622	2222
Integer variables	240	384	336	384	432	336	336
Continuous variables	97	529	481	529	337	817	577
Instance 10							
Constraints	878	1982	1838	1430	2174	1622	2222
Integer variables	240	384	336	384	432	336	336
Continuous variables	97	529	481	529	337	817	577

According to the method's average performance shown in Figs. 4 and 5 it can be concluded that the top three methods are VGG, DR2-CH and HK2 while DR2-BM, DR1-BM and HK1 are the bottom three. This is verified in the three problems analyzed. For cutting stock problem DR2-CH best performs since it shows the best mean ranking, really near the value of 1 and the lowest standard deviation. Similarly, VGG shows the best performance for the delivery problem while HK2 is considered the number one for the trim-loss problem. Nevertheless, these are average results and slight differences could occur for specific instances.

In order to perform an assessment of the relation between the instances' size and the methods' execution time, we group the three most efficient approaches (HK2, VGG and DR2-CH) and the least efficient (HK1, DR1-BM and DR2-BM). In this analysis, model sizes are represented by the number of non-zero elements of the problems' matrix.

In the case of trim-loss problem, Table 5 shows that for the first group of methods, the maximum model size is around 3500 non-zero elements corresponding to the longest CPU time (7000 s). In contrast, the second set in Table 5 presents larger sizes (almost 6000 non-zero matrix coefficients) with several instances taking around 10,000 s to reach a solution. Since execution time was limited to this value, the obtained solutions present a gap greater than zero (marked with (*) in Tables 5–7).

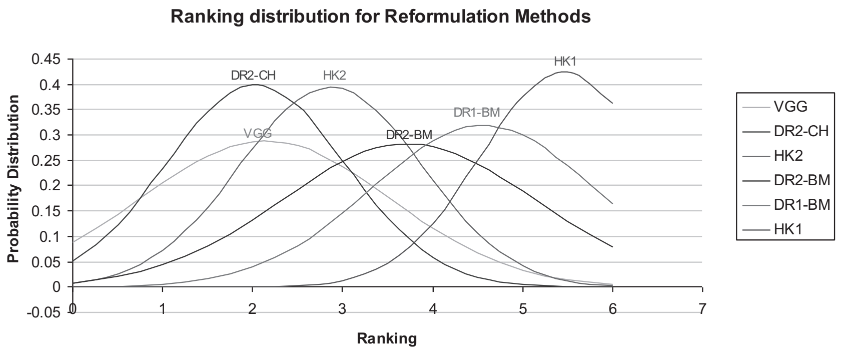


Fig. 3. Ranking distribution for reformulation methods.

Table 4
Purchase, inventory and delivery problem sizes used in this study.

	Original	HK1	HK2	VGG	DR1-BM	DR2-CH	DR2-BM
Instance 1							
Constraints	395	1259	920	893	1055	839	1151
Integer variables	162	270	216	324	297	216	216
Continuous variables	358	655	559	775	397	721	628
Instance 2							
Constraints	265	1057	619	601	853	565	769
Integer variables	108	216	144	216	234	144	144
Continuous variables	319	504	412	556	304	520	454
Instance 3							
Constraints	265	1057	619	601	853	565	769
Integer variables	108	216	144	216	234	144	144
Continuous variables	319	504	412	556	304	520	454
Instance 4							
Constraints	265	841	619	529	709	565	769
Integer variables	108	180	144	180	198	144	144
Continuous variables	319	472	412	484	304	520	454
Instance 5							
Constraints	241	601	485	421	539	449	583
Integer variables	102	138	120	138	156	120	120
Continuous variables	283	370	344	370	272	416	370
Instance 6							
Constraints	241	601	485	421	539	449	583
Integer variables	102	138	120	138	156	120	120
Continuous variables	283	370	344	370	272	416	370
Instance 7							
Constraints	323	899	682	593	763	625	836
Integer variables	144	207	171	198	225	101	171
Continuous variables	280	508	423	469	315	601	469
Instance 8							
Constraints	323	899	682	593	763	625	836
Integer variables	144	207	171	198	225	101	171
Continuous variables	280	508	423	469	315	601	469
Instance 9							
Constraints	323	899	682	593	763	625	836
Integer variables	144	207	171	198	225	101	171
Continuous variables	280	508	423	469	315	601	469
Instance 10							
Constraints	241	601	485	421	539	449	583
Integer variables	102	138	120	138	156	120	120
Continuous variables	283	370	344	370	272	416	370

A similar analysis can be done for the cutting stock problem from Table 6, the matrix maximum size includes 7500 elements in the best methods while it is 9500 in the second group. The difference in CPUs to reach the optimal solution is even more significant. HK2, VGG and DR2-CH methods do not need more than 200 s to obtain the global solution while the other three requires almost 3000 CPUs.

In the case of the delivery problem from Table 7, instances' size of the first approaches do not exceed the 3300 non-zero elements converging at less than 1400 s. Quite different is the case of methods HK1, DR1-BM and DR2-BM in which maxima sizes are around 3700 but execution time takes 10,000 s for several instances.

All these results quantify the differences between the advantageous methods (HK2, VGG and DR2-CH) and the less efficient ones (HK1, DR1-BM and DR2-BM). In general, there is a correspondence between the model sizes, defined as the quantity of non-zero elements of the matrix, and the execution time, although data dependence is observed in some instances solved. Then, reformulation methods with less number of variables and constraints are preferable than those with the opposite characteristic.

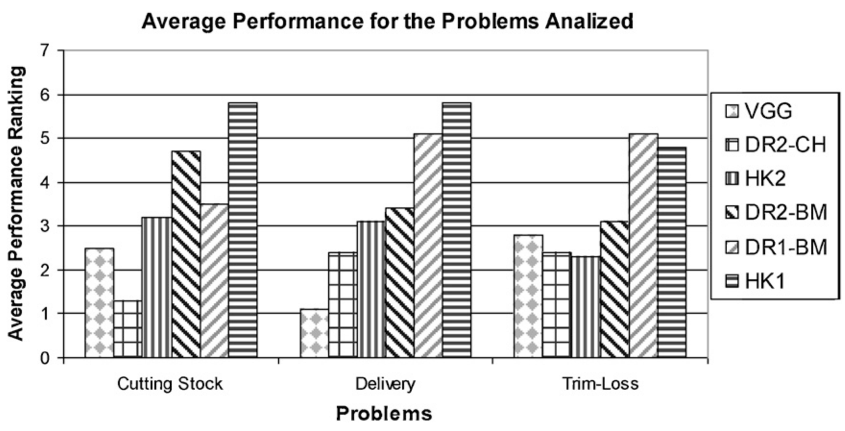


Fig. 4. Average performance ranking of methods according to problem type.

Table 5
Trim-loss results.

Instance 1	HK1	HK2	VGG≡DR1-CH	DR1-BM	DR2-CH	DR2-BM
Non-zero elements	679	322	436	595	322	358
CPUs	0.782	0.437	0.328	**2.172**	0.328	0.375
Iterations	7697	1527	1340	7446	930	1609
Nodes	1013	512	390	1254	230	502
Instance 2						
Non-zero elements	4915	1531	2935	3811	1531	1627
CPUs	118	96	59	**189**	101	169
Iterations	54220	18232	45438	41412	18954	20502
Nodes	5551	3655	5043	5621	3828	3645
Instance 3						
Non-zero elements	1173	553	741	993	553	601
CPUs	**3.3**	1.18	2.6	2.8	1.21	1.11
Iterations	54220	18232	45438	41412	18954	20502
Nodes	5551	3655	5043	5621	3828	3645
Instance 4						
Non-zero elements	1441	677	901	1193	677	725
CPUs	**8.1**	2.5	2.3	4.1	2.1	1.7
Iterations	112534	39105	28333	67131	31330	27811
Nodes	9212	8729	4389	8204	5711	3828
Instance 5						
Non-zero elements	3916	1676	2356	3036	1676	1756
CPUs	**416.3**	132.0	12.1	214.0	61.0	113.0
Iterations	6897441	1997429	26031	3953917	1032497	2200508
Nodes	177954	253083	7443	228754	115096	158728
Instance 6						
Non-zero elements	5827	2491	3487	4459	2491	2587
CPUs	**10000 (*)**	3946	7194	8958	5472	**10000 (*)**
Iterations	1061100830	34551782	40614959	98908991	29662582	261345880
Nodes	6807148	7488991	7665443	4374560	7459550	20261198
Instance 7						
Non-zero elements	4386	1876	2631	3376	1876	1956
CPUs	2630.17	2175	1878.7	**7194**	3501	3100
Iterations	12400971	12592778	2303940	34718413	43233065	20136287
Nodes	3032832	6295745	529022	12247498	3599834	7214109
Instance 8						
Non-zero elements	1441	677	901	1193	677	725
CPUs	**8.38**	3	5.3	3.4	2.14	4.1
Iterations	104228	37747	60327	54338	27768	46530
Nodes	19721	13550	14631	6421	7889	16341
Instance 9						
Non-zero elements	2471	1156	1526	1991	1156	1216
CPUs	72.92	12.43	24.72	**413.3**	14.06	10.56
Iterations	835641	136257	283245	3249018	153963	150813
Nodes	129588	44136	58002	1239948	45793	25381
Instance 10						
Non-zero elements	3916	1676	2356	3081	1676	1756
CPUs	1254.39	848	1355.29	**10000 (*)**	1150	1300.97
Iterations	6173384	6497694	10054394	178515235	6372680	5859523
Nodes	1711229	2327017	1613391	18785808	3052535	2004686

Although the analysis of MINLP algorithms is beyond the scope of this work, the original non-convex problems were also solved presenting the difficulty of finding the global solution, as it was expected. Bilinear problems present several suboptimal solutions so local optimization algorithms like DICOPT does not reach the global one in most cases. Furthermore, there is a critical influence of the initial points to get even a feasible solution. It is noteworthy that global optimization algorithm (BARON) was also used to solve the original formulation but converge was extremely slow, so this strategy was excluded from results.

Table 6
Cutting stock results.

Instance 1	HK1	HK2	VGG≡DR1-CH	DR1-BM	DR2-CH	DR2-BM
Non-zero elements	6551	6047	5383	7135	6391	7591
CPUs	975.24	220.67	78.09	479.13	80.73	328.28
Iterations	5713955	1408649	817395	3814596	513789	2195683
Nodes	142181	43943	9588	64908	10929	54582
Instance 2						
Non-zero elements	4839	4524	3971	5168	4601	5525
CPUs	14.43	4.23	4.83	5.9	2.03	4.52
Iterations	149972	44836	51702	65030	24100	47737
Nodes	2800	1019	1242	1794	549	1149
Instance 3						
Non-zero elements	5000	4721	4096	5272	4672	5656
CPUs	246.11	60.58	57.22	74	36.19	133.67
Iterations	2215446	574926	671708	607865	320791	784696
Nodes	42108	12605	11375	17551	7870	30175
Instance 4						
Non-zero elements	8081	7001	6551	8801	7271	8801
CPUs	1071.2	68.7	63.66	97.3	22.74	123.16
Iterations	5931849	492324	551422	804183	133431	975773
Nodes	122409	9373	12882	14724	2457	9051
Instance 5						
Non-zero elements	8621	7001	6911	9461	7271	8801
CPUs	2987.49	107.4	128.2	97.4	97.0	218.4
Iterations	15259631	930107	1189785	750752	471149	1314098
Nodes	329174	13928	15564	13298	6313	24971
Instance 6						
Non-zero elements	7101	6525	5845	7661	6997	8269
CPUs	592.33	117	45.91	104	11.11	252.13
Iterations	4737243	1115364	562139	826222	99200	2290451
Nodes	155947	27578	14253	37549	2560	52983
Instance 7						
Non-zero elements	7677	6225	6229	8493	6997	8397
CPUs	1469.5	25.66	51.58	49.74	10.53	64.33
Iterations	12534117	151349	361204	351735	80813	874172
Nodes	402549	5414	15318	10247	2017	30228
Instance 8						
Non-zero elements	7677	6525	6229	8493	6997	8397
CPUs	103.44	36.75	7.97	34.3	12.11	55.72
Iterations	1586869	386391	48982	327364	104151	599261
Nodes	32481	6759	2396	8794	3180	9016
Instance 9						
Non-zero elements	6050	5618	4970	6482	5934	6962
CPUs	80.23	32	10.66	8.42	9.92	30.52
Iterations	726955	415222	137707	59119	129090	266049
Nodes	21286	11464	4268	2705	2829	9727
Instance 10						
Non-zero elements	6458	5594	5234	6986	5810	7034
CPUs	21.25	15.2	6.64	24.85	3.39	32.88
Iterations	190181	142992	81315	208385	40075	381418
Nodes	6723	6292	2037	6718	958	9098

Table 7
Purchase, delivery and inventory results.

Instance 1	HK1	HK2	VGG≡DR1-CH	DR1-BM	DR2-CH	DR2-BM
Non-zero elements	3661	2632	3199	3523	2470	3256
CPUs	9320.2	361.84	9.78	**10000 (*)**	372.85	601.63
Iterations	51756453	2793048	11637	46498630	2506289	4949379
Nodes	4567500	191636	1292	5067949	215541	313794
Instance 2						
Non-zero elements	3124	1798	2176	2968	1690	2206
CPUs	3277.81	71.8	0.83	**3830.84**	31.5	63.4
Iterations	10768085	472364	1980	15070367	200007	465888
Nodes	2083696	92370	416	4368707	36669	85848
Instance 3						
Non-zero elements	3124	1798	2176	2968	1690	2206
CPUs	7843.53	1322.77	106.2	**10000**	1379.5	1839.07
Iterations	29740430	8603393	651962	43079269	7912131	11387541
Nodes	5741900	1091669	72508	7361639	1171468	1441158
Instance 4						
Non-zero elements	2476	1798	1816	2392	1690	2206
CPUs	**7274.56**	741.0	7.7	3422.56	356.92	521.69
Iterations	40281605	3983122	48481	14063118	2373003	3486273
Nodes	4744940	687209	4259	2537592	349592	471198
Instance 5						
Non-zero elements	1744	1404	1348	1686	1332	1672
CPUs	**82.39**	10.0	1.7	27.56	7.5	5.8
Iterations	564155	64293	6974	234868	51378	46606
Nodes	79408	11849	922	32729	7140	5849
Instance 6						
Non-zero elements	1744	1404	1348	1686	1332	1672
CPUs	**199.42**	38.8	3.28	48	23.47	58.88
Iterations	1274052	279438	8123	314432	168324	452104
Nodes	183446	52208	1170	51569	26822	70615
Instance 7						
Non-zero elements	2533	1925	1855	2285	1802	2341
CPUs	**428**	21.27	1.65	63.84	20.77	22.75
Iterations	2028941	102245	3794	344979	98532	106076
Nodes	296234	23946	558	60221	20794	24383
Instance 8						
Non-zero elements	2533	1925	1855	2285	1802	2341
CPUs	**4598.78**	39.2	3.95	567.25	30.05	21.27
Iterations	24825948	241955	19352	3222857	196837	93416
Nodes	2526264	41334	2526	507412	28553	22571
Instance 9						
Non-zero elements	2533	1925	1855	2285	1802	2341
CPUs	**2136.08**	6.55	3.5	809.11	2.83	8.4
Iterations	10869012	32297	14347	5083615	10729	40053
Nodes	1311079	6640	1900	572520	1852	7848
Instance 10						
Non-zero elements	1744	1404	1348	1686	1332	1672
CPUs	**107.95**	9.78	1.45	45.23	13.89	11.66
Iterations	912775	70493	4474	359455	100351	78300
Nodes	94773	10350	693	53499	14408	13048

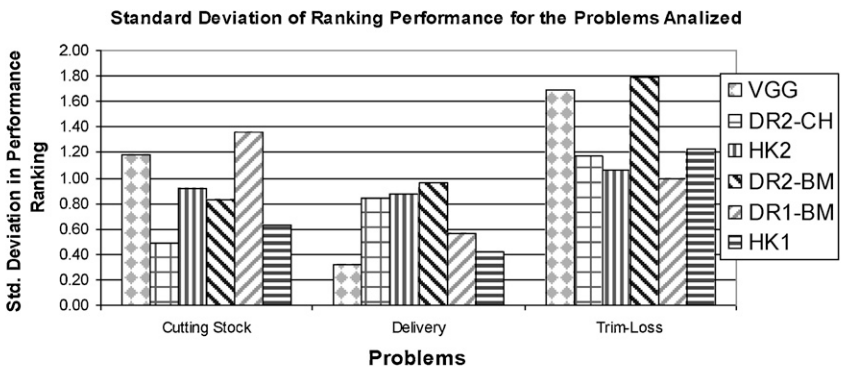


Fig. 5. Standard deviation in performance ranking of methods according to problem type.

Table 8
Ranking mean and standard deviation.

	VGG	DR2-CH	HK2	DR2-BM	DR1-BM	HK1
Mean	2.13	2.07	2.87	3.73	4.60	5.43
Std. deviation	1.38	1.01	1.01	1.41	1.28	0.94

5. Conclusions

In this article, several transformation techniques are presented for MINLP problems with bilinear terms involving at least one discrete variable. Two new disjunctive methods are proposed which are general enough to be applied to any kind of problem with this characteristic. These strategies are quantitatively and qualitatively compared to various techniques proposed by other authors.

Qualitative assessment brings the following conclusions:

- An interesting connection was found between Petersen's and Harjunkoski's methods (HK1 and HK2). The first case is applied to bilinear products of one binary variable by a continuous (or integer) one while the second one is applied to bilinear products of an integer variable by a continuous (or integer) variable. It could be observed that Petersen adds a new constraint and one variable for each bilinear term in the problem. Similarly, HK1 and HK2 methods in the first place transform the bilinear product into the sum of various products like those studied by Petersen. Then, they applied identical constraints and variables for each of these terms. This similarity has not been reported previously.
- Disjunctive methods proposed were transformed using Big-M and Convex Hull relaxations in order to be implemented as MILP problems. It was observed that these relaxations could be simplified eliminating some superfluous variables and constraints. After these simplifications, a remarkable conclusion appeared: the Convex Hull relaxation of DR1 method leads to the Voudouris, Ghattas and Grossmann technique. This also explains the good performance of this method when it was applied for batch design problems. It also gives reasons for its compact and tight representation.
- RRLT method is a potentially powerful strategy since in many cases it could reformulate the bilinear problem without introducing additional variables or constraints. However, it was also noticed that this strategy does not guarantee to get a linear model. This restricted the use of this method in some of the cases studied in this work. Another limitation comes from the case of bilinear products of two integer variables. The new variable introduced to replace the bilinear problem does not allow keeping the original integer nature of the eliminated one.
- Since model size of reformulated problems is a critical issue in order to achieve an efficient solution convergence, one important feature of this qualitative analysis is placed on the quantification of the number of constraints, continuous and binary variables introduced by each method. This is also strengthened by the performance comparison carried out in Section 4.

From the quantitative study some other conclusions are drawn:

- Three typical problems from the literature are considered and ten instances of each are executed using all transformations presented. Those models were analyzed in terms of the convergence time, number of nodes and iteration to reach the solution. In this regards, conclusions can be stated as follows:
    - Methods are ranked according to the best execution time for each run. This performance reference is equivalent to the number of nodes and iterations to reach the optimum. From this ranking, the three best techniques are VGG (equivalent to DR1-CH), DR2-CH and HK2. Analyzing these methods' characteristics, it is noteworthy that they employ: a two based representation to rewrite the integer variable and/or the convex hull relaxation. According to this, DR2-CH positions as the most robust method.
    - HK1, DR1-BM and DR2-BM methods take more time to get a solution, even though tight values were determined for Big M parameters. They also lead to larger model sizes.
    - Execution time is data dependent. Although methods maintain their performance in relative terms, solution time can vary significantly from one instance to the other even with minor changes in problem size.

In the context analyzed, linearization techniques are required to deal with this kind of problems and to obtain a global solution. With this purpose, this work establishes and compares a set of methods that could successfully be used to achieve this goal.

Acknowledgement

The authors would like to acknowledge financial support from the Consejo Nacional de Investigaciones Científicas y Técnicas under grant PIP 1817.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.compchemeng.2012.09.011.

References

Adams, W., & Sherali, H. (1990). Linearization strategies for a class of zero-one mixed integer programming problems. *Operations Research*, *38*, 217–226.
Chang, C. (2000). An efficient linearization approach for mixed-integer problems. *European Journal of Operational Research*, *123*, 652–659.
Glover, F. (1975). Improved linear integer programming formulations of nonlinear integer programs. *Management Science*, *22*, 455–460.

Glover, F., & Woolsey, E. (1974). Converting the 0–1 polynomial programming problem to a 0–1 linear program. *Operations Research*, *22*, 180–182.

Gounaris, C., Misener, R., & Floudas, C. (2009). Computational comparison of piecewise-linear relaxation for pooling problems. *Industrial Engineering and Chemistry Research*, *48*, 5742–5766.

Grossmann, I. (2002). Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and Engineering*, *3*, 227–252.

Grossmann, I., Voudouris, V., & Ghattas, O. (1992). Mixed-integer linear programming formulations of some nonlinear discrete design optimization problems. In *Recent Advances in Global Optimization* (pp. 478–512). Princeton, New Jersey, US: Princeton University Press.

Harjunkoski, I., Westerlund, T., & Pörn, R. (1999). Numerical and environmental considerations on a complex industrial mixed integer non-linear programming (MINLP) problem. *Computers and Chemical Engineering*, *23*, 1545–1561.

Ierapetritou, M., & Pistikopoulos, E. (1996). Batch plant design and operations under uncertainty. *Industrial Engineering and Chemistry Research*, *36*, 772–787.

Liberti, L., & Pantelides, C. (2006). An exact reformulation algorithm for large nonconvex NLPs involving bilinear terms. *Journal of Global Optimization*, *36*, 161–189.

McCormick, G. (1976). Computability of global solutions to factorable nonconvex programs: Part I—Convex underestimating problems. *Mathematical Programming*, *10*, 147–175.

Moreno, M., & Montagna, J. (2009). A multiperiod model for production planning and design in a multiproduct batch environment. *Mathematical and Computer Modelling*, *49*, 1372–1385.

Moro, L., & Pinto, J. (2004). Mixed-integer programming approach for short-term crude oil scheduling. *Industrial Engineering and Chemistry Research*, *43*, 85–94.

Petersen, C. (1971). *A note on transforming the product of variables to linear form in linear programs*. West Lafayette, IN: Purdue University.

Pörn, R., Harjunkoski, I., & Westerlund, T. (1999). Convexification of different classes of non-convex MINLP problems. *Computers and Chemical Engineering*, *23*, 439–448.

Psarris, P., & Floudas, C. (1990). Improving dynamic operability in MIMO systems with time delays. *Chemical Engineering Science*, *45*, 3505–3524.

Rodriguez, M., & Vecchietti, A. (2008). Enterprise optimization for solving an assignment and trim-loss non-convex problem. *Computers and Chemical Engineering*, *32*, 2812–2822.

Rodriguez, M., & Vecchietti, A. (2010). Inventory and delivery optimization under seasonal demand in the supply chain. *Computers and Chemical Engineering*, 1705–1718.

Sherali, H., & Adams, W. (1986). A tight linearization and an algorithm for 0–1 quadratic programming problems. *Management Science*, *32*, 1274–1290.

Sherali, H., & Adams, W. (1994). A hierarchy of relaxations and convex hull characterizations for mixed-integer zero-one programming problems. *Discrete Applied Mathematics*, *52*, 83–106.

Sherali, H., & Tuncbilek, C. (1992). A global optimization algorithm for polynomial programming problems using a reformulation-linearization technique. *Journal of Global Optimization*, *2*, 101–112.