Submitted to *INFORMS Journal on Computing*
manuscript JOC-2013-07-OA-119

Algorithmic approach for improved mixed-integer reformulations of convex Generalized Disjunctive Programs

Francisco Trespalacios
Carnegie Mellon University, ftrespal@andrew.cmu.edu,

Ignacio E. Grossmann
Carnegie Mellon University, grossmann@cmu.edu,

In this work, we propose an algorithmic approach to improve mixed-integer models that are originally formulated as convex Generalized Disjunctive Programs (GDP). The algorithm seeks to obtain an improved continuous relaxation of the MILP/MINLP reformulation of the GDP, while limiting the growth in the problem size. There are three main stages that form the basis of the algorithm. The first one is a pre-solve, consequence of the logic nature of GDP, which allows us to reduce the problem size, find good relaxation bounds and identify properties that help us determine where to apply a basic step. The second stage is the iterative application of basic steps, selecting where to apply them, and monitoring the improvement of the formulation. Finally, we use a hybrid reformulation of GDP that seeks to exploit both of the advantages attributed to the two common GDP-to-MILP/MINLP transformations, the Big-M and Hull reformulation. We illustrate the application of this algorithm with several examples. The results show the improvement in the problem formulations by generating models with improved relaxed solutions and relatively small growth in the number of continuous variables and constraints. The algorithm generally leads to reduction in the solution times.

Key words: generalize disjunctive programming; mixed-integer nonlinear programming

1. Introduction

Mixed-integer linear and mixed-integer nonlinear programming models (MILP/MINLP) arise in different areas, such as process design(Duran and Grossmann 1986, Floudas 1995, Turkay and Grossmann 1996), layout problems(Sawaya 2006), financial modeling and electrical power management(Stubbs and Mehrotra 1999). MILP/MINLP models can be formulated in different ways, and therefore efficiency of the algorithms to solve these problems

strongly depends on the size of the corresponding formulation and tightness of its continuous relaxation.

Generalized Disjunctive Programming (GDP) is an alternative higher-level representation of these problems(Raman and Grossmann 1994) that involves not only algebraic equations, but also disjunctions and logic propositions in terms of Boolean and continuous variables. This approach facilitates the development of the models by making the formulation process more systematic. Although there are some special techniques to solve this type of problems, such as Disjunctive Branch and Bound(Lee and Grossmann 2005) and Logic Based Outer Approximation(Turkay and Grossmann 1996), GDPs are normally reformulated as MILP/MINLP(Nemhauser and Wolsey 1988, Lee and Grossmann 2000) to exploit the developments in these solvers.

Some of the methods to solve convex MINLP problems include branch and bound(Dakin 1965, Gupta and Ravindran 1985), branch and cut(Stubbs and Mehrotra 1999), generalized Benders decomposition(Geoffrion 1972), outer approximation(Duran and Grossmann 1986, Abhishek et al. 2010), LP/NLP based branch and bound(Quesada and Grossmann 1992) and extended cutting planes(Westerlund and Pettersson 1995). A comprehensive review of MINLP techniques is given by Grossmann(Grossmann 2002), and of MILP methods and progress is given by Bixby et al.(Bixby et al. 2000, Bixby and Rothberg 2007).

The reformulation of GDP models to MILP/MINLP problems is typically done by using either the Big-M (BM) or the Hull-Reformulation (HR), where the former generates a smaller MILP/MINLP, while the latter generates a tighter one(Grossmann and Lee 2003, Vecchietti et al. 2003).

In this work, we make use of the logic structure of a GDP. Instead of directly reformulating it as an MILP/MINLP, we apply a pre-analysis, a logic operation called basic step, and a hybrid reformulation. This logic manipulation allows us to obtain an improved formulation in comparison to the one obtained by a traditional reformulation. The resulting model can be solved by a GDP or an MILP/MINLP algorithm. Figure 1 outlines the main idea of this work. We should note that the proposed pre-analysis has some similarities to the MILP "fixing variables" pre-solve technique(Savelsbergh 1994, Achterberg 2004, Lodi 2010, Mahajan 2010), but applied in the GDP space for convex nonlinear GDP models.

This paper is organized as follows. Section 2 provides an overview of GDP and relevant concepts in disjunctive programming, providing a theoretical background for the algorithm.

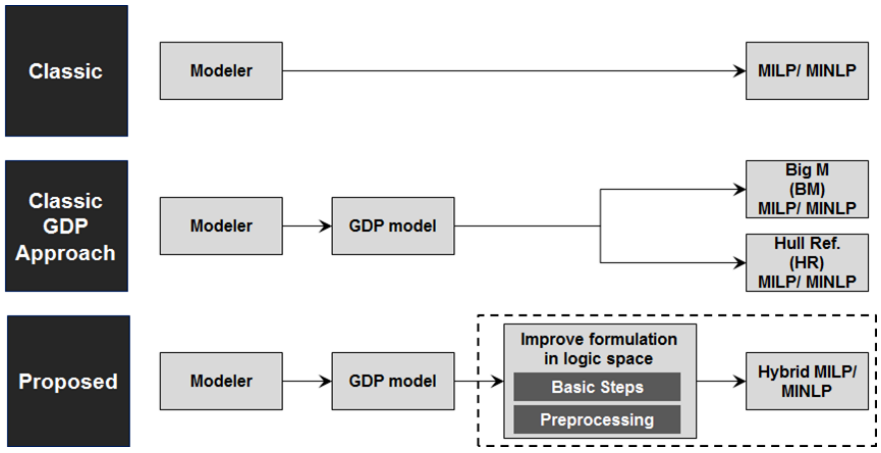

Figure 1 **Different modeling approaches**

Section 3 first presents the proposed algorithm, and second describes in detail each step of the method. Section 4 provides an example of the application of the algorithm. Section 5 presents the statistics, results and performance of different test examples.

2. Background

In order to improve problem formulations, we rely on two main tools. The first one is Generalized Disjunctive Programming, which is a higher level representation for MINLP. The second one is a logic operation called basic step that allows the generation of tighter formulations. In this section we provide the background for these two concepts.

2.1. Generalized Disjunctive Programming (GDP)

Generalized Disjunctive Programming(Raman and Grossmann 1994, Grossmann and Ruiz 2012) allows the systematic modeling of optimization problems by using algebraic equations, disjunctions and logic propositions. GDP can be considered an extension to the well-known theory of disjunctive programming developed by Balas(Balas 1979).

2.1.1. GDP formulation. The general GDP formulation can be represented as follows:

$$
\begin{aligned}
& min\, z = f(x) \\
s.t. \quad & g(x) \leq 0 \\
& \bigvee_{i \in D_k} \begin{bmatrix} Y_{ki} \\ r_{ki}(x) \leq 0 \end{bmatrix} && k \in K \\
& \veebar_{i \in D_k} Y_{ki} && k \in K \\
& \Omega(Y) = True \\
& x^{lo} \leq x \leq x^{up} \\
& x \in \mathbb{R}^n \\
& Y_{ki} \in \{True, False\} \quad k \in K, i \in D_k
\end{aligned}
\tag{GDP}
$$

As shown in (GDP), the objective is a function of the continuous variables x. The global constraints $g(x)$ must hold true regardless of the discrete decisions. Each of the disjunctions $k \in K$ contains disjunctive terms $i \in D_k$ that are linked together by an OR operator ($\vee$). For each disjunctive term in each disjunction, a Boolean variable Y_{ki} is assigned with a corresponding set of inequalities $r_{ki}(x) \leq 0$. Only one term in each disjunction can be selected $\veebar_{i \in D_k} Y_{ki}$. When a disjunctive term is active ($Y_{ki} = True$), then the corresponding inequalities are enforced. When it is not active ($Y_{ki} = False$), the constraints are ignored. $\Omega(Y) = True$ represents the logic relations between the Boolean variables. In the particular case when $f(x)$, $g(x)$ and $r_{ki}(x)$ are convex the problem becomes a convex GDP.

2.1.2. MINLP reformulation of GDP. In order to take full advantage of existing solvers(Grossmann 2002), GDP problems are normally reformulated as MILP/MINLP by using either the Big-M(Nemhauser and Wolsey 1988) (BM) or Hull Reformulation(Lee and Grossmann 2000) (HR). (BM) generates a smaller MILP/MINLP, while (HR) generates a tighter one at the expense of larger number of variables and constraints(Grossmann and Lee 2003, Vecchietti et al. 2003). The (BM) reformulation is given as follows:

$$
\begin{aligned}
& min\, z = f(x) \\
s.t. \quad & g(x) \leq 0 \\
& r_{ki}(x) \leq M^{ki}(1 - y_{ki}) && k \in K, i \in D_k \\
& \sum_{i \in D_k} y_{ki} = 1 && k \in K \\
& Hy \geq h \\
& x^{lo} \leq x \leq x^{up} \\
& x \in \mathbb{R}^n \\
& y_{ki} \in \{0,1\} && k \in K, i \in D_k
\end{aligned}
\tag{BM}
$$

In (BM) the Boolean variables Y_{ki} are transformed to binary variables y_{ki} with a one-to-one correspondence (i.e. $Y_{ki} = True$ is equivalent to $y_{ki} = 1$, while $Y_{ki} = False$ is equivalent to $y_{ki} = 0$). The transformation of logic relations ($\Omega(Y) = True$) to integer linear constraints ($Hy \geq h$) can be easily obtained(Clocksin and Mellish 1981, Williams 1985, Biegler et al. 1997, Grossmann and Trespalacios 2013). The equation $\sum_{i \in D_k} y_{ki} = 1$ guarantees that only one disjunctive term is selected per disjunction. For a selected term ($y_{ki} = 1$) the corresponding constraints $r_{ki}(x) \leq 0$ are enforced. For a term not selected ($y_{ki} = 0$) and a large enough M^{ki} the corresponding constraint $r_{ki}(x) \leq M^{ki}$ becomes redundant.

The (HR) formulation is given as follows:

$$
\begin{aligned}
& min\, z = f(x) \\
s.t. \quad & g(x) \leq 0 \\
& x = \sum_{i \in D_k} \nu^{ki} && k \in K \\
& y_{ki} r_{ki}(\nu^{ki}/y_{ki}) \leq 0 && k \in K, i \in D_k \\
& \sum_{i \in D_k} y_{ki} = 1 && k \in K \\
& Hy \geq h \\
& x^{lo} y_{ki} \leq \nu^{ki} \leq x^{up} y_{ki} && k \in K, i \in D_k \\
& x \in \mathbb{R}^n \\
& y_{ki} \in \{0,1\} && k \in K, i \in D_k
\end{aligned}
\tag{HR}
$$

In (HR), the Boolean variables Y_{ki} are also transformed into binary variables y_{ki}, and the logic relations ($\Omega(Y) = True$) into integer linear constraints ($Hy \geq h$). Here, the con-

tinuous variables x are disaggregated into variables $\nu^{ki}, i \in D_k$ ($x = \sum_{i \in D_k} \nu^{ki}$) for each of the disjunctions $k \in K$. The constraints in each term $i \in D_k$ of a disjunction $k \in K$ are represented by the perspective function $y_{ki} r_{ki}(\nu^{ki}/y_{ki})$. The constraint $x^{lo} y_{ki} \leq \nu^{ki} \leq x^{up} y_{ki}$ enforces that if a disjunction is selected ($y_{ki} = 1$), then its corresponding variables have to lie within the limits of x ($x^{lo} \leq \nu^{ki} \leq x^{up}$), and they have to satisfy their corresponding constraints ($r_{ki}(\nu^{ki}) \leq 0$). If it is not selected ($y_{ki} = 0$), then its variables $\nu^{ki} = 0$, and their corresponding constraints are trivially satisfied ($0 \leq 0$). Note that when the constraints in the disjunction are linear ($A^{ki} x \leq a^{ki}$), then $y_{ki} r_{ki}(\nu^{ki}/y_{ki}) \leq 0$ becomes $A^{ki} \nu^{ki} \leq a^{ki} y_{ki}$. For the nonlinear case, to avoid singularities, the following approximation can be used for the perspective function(Sawaya 2006):

$$y_{ki} r_{ki}(\nu^{ki}/y_{ki}) \approx ((1-\epsilon) y_{ki} + \epsilon) r_{ki}\left(\frac{\nu^{ki}}{(1-\epsilon) y_{ki} + \epsilon}\right) - \epsilon r_{ki}(0)(1 - y_{ki}) \qquad \text{(APP)}$$

where ϵ is a small finite number (e.g. 10^{-5}). This approximation yields an exact value at $y_{ki} = 0$ and $y_{ki} = 1$, and it is convex if r_{ki} is convex. In some cases, particularly for linear constraints, algebraic manipulation of the problem allows the elimination of some disaggregated variables, reducing the size of the problem(Raman and Grossmann 1994). An example of GDP reformulation can be found in the on-line supplement.

It is clear that formulation (HR) involves more variables and constraints than (BM), but it provides a tighter relaxation(Grossmann and Lee 2003, Vecchietti et al. 2003). The (HR) reformulation represents the intersection of the convex hulls of each disjunction, and it is consistent with the MINLP representations of disjunctive convex sets previously characterized(Ceria and Soares 1999, Stubbs and Mehrotra 1999).

Figure 2 illustrates, for both reformulations, the projection over $x1$ and $x2$ of the feasible region defined by two disjunctions. The first disjunction represents the selection of rectangle $A1$ or rectangle $A2$, and the second one the selection of circle $B1$ or circle $B2$. The dashed region defines the feasible region, and the shaded area represents the continuous relaxation of the (BM) and (HR). It is clear that the (HR) has a tighter relaxation than the (BM).

It is important to note that even though the (HR) is the intersection of the convex hulls of the individual disjunctions, this in general does not mean that it is the convex hull of the feasible region as can be seen in Figure 2. In order to further improve the tightness of the (HR) we will make use of the logic operation called basic step.

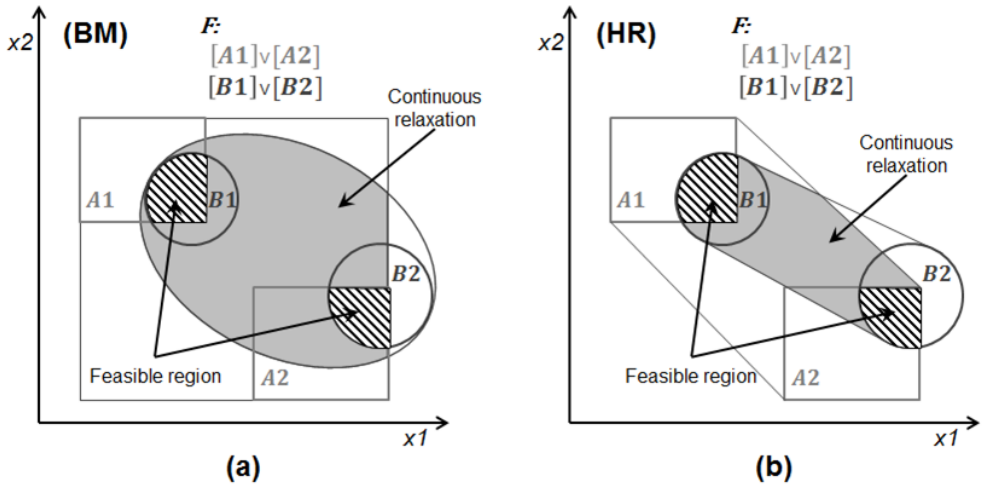

Figure 2 Illustration of (BM) and (HR) reformulations

2.2. Basic Steps

Basic step is a logic operation that allows the tightening in the formulation of disjunctive programming. It is important to review some of the basic definitions in disjunctive convex programming before describing this operation.

2.2.1. Disjunctive convex sets and equivalent forms. Disjunctive convex programming can be defined as the optimization over a disjunctive convex set. A disjunctive set can be described as the union ($\cup$) and intersection ($\cap$) of a collection of inequalities. Consider the following definitions(Balas 1985, Ruiz and Grossmann 2012):

Convex inequality (Half space in linear case): $C = \{x \in \mathbb{R}^n | \Phi(x) \leq 0\}$, where $\Phi(x) : \mathbb{R}^n \to \mathbb{R}^1$ is a convex function.

Convex set (Polyhedron in linear case): $P = \bigcap_{m \in M} C_m$

Elementary disjunctive set: $H = \bigcup_{m \in M} C_m$

Disjunction: $S_k = \bigcup_{i \in D_k} P_i = \bigcup_{i \in D_k} \bigcap_{m \in M_i} C_m$

A disjunction such that $S_k = P_i$ for some $i \in D_K$ is called improper disjunction (note that if D_k is a singleton then S_k is improper); otherwise it is called a proper disjunction.

Any disjunctive convex set can be expressed in many logically equivalent forms. There are three forms of particular interest:

Regular form (Intersection of disjunctions): $F = \bigcap_{k \in K} S_k$

Conjunctive normal form (CNF): $F = \bigcap_{k \in K} H_k$

Disjunctive normal form (DNF): $F = S = \bigcup_{i \in D} P_i$

CNF and DNF are the two extremes of a disjunctive set in regular form. Note that any GDP is in regular form(Ruiz and Grossmann 2012), where the global constraints are improper disjunctions.

2.2.2. Hierarchy of relaxations and basic steps. A basic step is a logic operation that brings any disjunctive set in regular form closer to its DNF. Theorem 2.1, which is stated and proved for the linear case(Balas 1985) , and extended for the general nonlinear convex case(Ruiz and Grossmann 2012), defines a basic step.

THEOREM 2.1. Let F be a disjunctive set in regular form. Then F can be brought to DNF by $|K|-1$ recursive applications of the following basic step which preserves regularity: For some $k,l \in K$, bring $S_k \cap S_l$ to DNF by replacing it with:

$$S_{kl} = S_k \cap S_l = \bigcup_{i \in D_k, j \in D_l} (P_i \cap P_j)$$

Any convex GDP can be proved to be equivalent to a disjunctive convex program(Ruiz and Grossmann 2012), so it is possible to use the rich theory behind disjunctive programming (including basic steps) to convex GDP. In particular, there are two main consequences of the basic steps(Balas 1985, Ruiz and Grossmann 2012): a) The continuous relaxation of the (HR) of a disjunctive set after a basic step is at least as tight, and generally tighter, than the one of the previous formulation; b) The (HR) of the DNF is the convex hull of the disjunctive convex set.

The first results indicates that we can obtain formulations with tighter continuous relaxations through basic steps. However, the problem grows exponentially in the number of constraints with the application of basic steps. It is therefore important to consider the tradeoff of tightening the problem formulation versus growing the problem size. An important consequence of the second result is that if the objective function is linear, then the optimal value to the relaxed (HR) of the DNF is the same as the optimal value to the original GDP.

Proper basic steps generate exponential growth in disjunctive terms, which leads to an exponential growth in the number of binary variables. However, Balas(Balas 1985) shows that we can obtain an equivalent formulation by including additional constraints, and representing the new terms with continuous variables between 0 and 1.

Figure 3 illustrates tightness of relaxation of the (HR) before and after the application of a basic step. The illustration shows a feasible region described by two disjunctions with

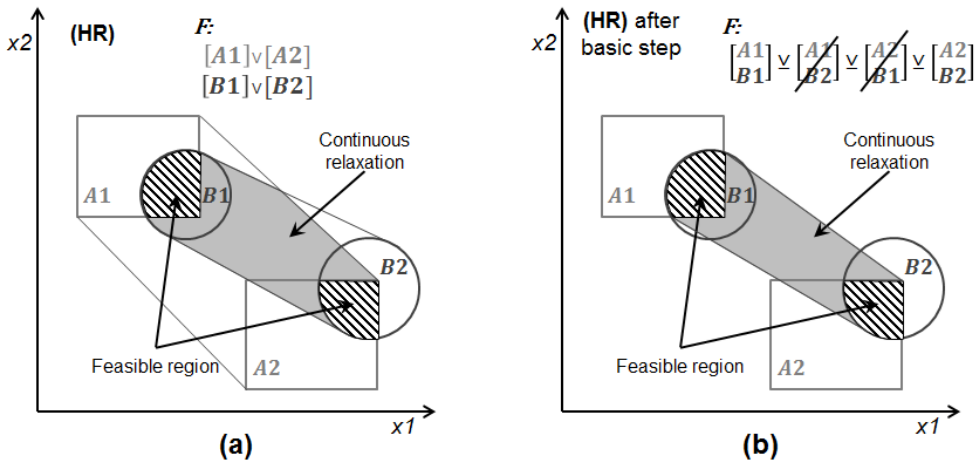

Figure 3 **Illustration of (HR) (a) before, and (b) after the application of a basic step**

two disjunctive terms each, that is $([A_1] \vee [A_2]) \wedge ([B_1] \vee [B_2])$. Figure 3.b shows that, after a basic step, the two disjunctions are intersected to form a new single disjunction $([A_1] \wedge [B_1]) \vee ([A_2] \wedge [B_2])$. Thus, the basic step not only improves the tightness of the relaxation, but that it brings the problem into DNF. The (HR) of the DNF, as expressed earlier, describes the convex hull of the feasible region. This can also be seen in 3.b. Finally, it is important to note that some of the resulting terms after the application of a basic step might become infeasible. In this example, since A1 and B2, and A2 and B1 do not intersect, the corresponding new terms are not feasible.

Therefore, the application of a basic step has the tradeoff of improving the tightness of the formulation but increasing the problem size. There has been some work to identify when it is convenient or not to apply a basic step(Balas 1985, Sawaya and Grossmann 2012, Ruiz and Grossmann 2012). Previous work provides some guidelines and case by case selection of basic steps; however, the exponential growth of the formulation is a major issue. In this paper we propose an algorithmic approach for selecting which basic steps to apply, while combining a pre-processing and a hybrid reformulation that allow us to keep the problem formulation relatively small.

3. Algorithm to improve GDP formulations

In order to improve GDP formulations, we iteratively apply basic steps. In this section we first describe the algorithm, and afterwards we explain in detail each of its steps. Figure 4 provides an outline of the algorithm, where the main idea is to first perform a preanalysis for pre-solving, then repeatedly apply basic steps over one single disjunction, and finally use the (HR) in that disjunction and (BM) in all the remaining ones.

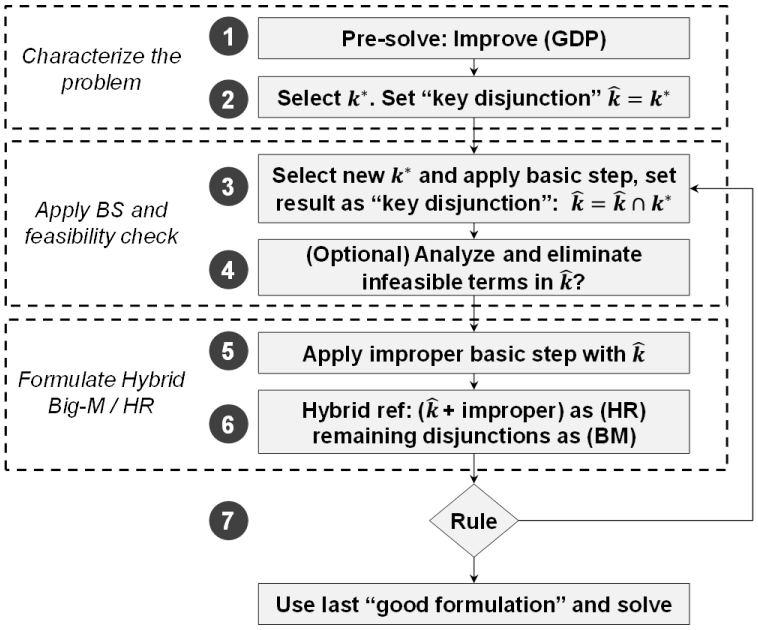

Figure 4 Outline of algorithm

3.1. Algorithm

For the description of the algorithm, we will define the global constraints as individual inequalities, such that $g(x) \leq 0$ is represented with $g_e(x) \leq 0$, $e \in E$.

Step 1. Initialize z^*, GDP^* and z^{lo} from pre-solve. Set $iter = 1$.

Goal. Use pre-solve to initialize the algorithm; improving (GDP), finding better bounds, and providing a value that characterizes each disjunction.

Step 2. Select disjunction $k^* \in K$. Set $\hat{k}_1 = k^*$; $\hat{K} = \{\hat{k}_1\}$; $\hat{D}_{k_1} = D_{k^*}$.

Goal. Select the first disjunction to which basic steps will be applied, and set this disjunction as the "key disjunction".

Step 3. Set $iter = iter + 1$. Select $k^* \in K \backslash \hat{K}$. Set $\hat{k}_{iter} = k^*$; $\hat{K} = \{\hat{k}_1, ..., \hat{k}_{iter}\}$; $\hat{D}_{k_{iter}} = D_{k^*}$; $\hat{D} = \{\hat{D}_{k_1}, ..., \hat{D}_{k_{iter}}\}$.

Goal. Select the next disjunction, and apply a basic step between this disjunction and the "key disjunction". Set the resulting disjunction of this basic step as "key disjunction".

Step 4 *(Optional).* For all $\hat{i} = (\hat{i}_1, ..., \hat{i}_{iter}) \in \hat{D}$, such that: $(GDP^*) \bigcap_{\substack{\hat{k}_s \in \hat{K} \\ s=1,...,iter}} (y_{\hat{k}_s \hat{i}_s} = 1)$ becomes infeasible, set $\hat{i} \in INEAS_{iter}$.

Goal. Identify which terms in the "key disjunction" are infeasible.

Step 5. Select global equations to which apply a basic step $\hat{E} \in E$.

Step 6. Solve the continuous relaxation of (GDPH).

$$
\begin{aligned}
& & & min\, z = f(x) & & \\
& s.t. & & g_e(x) \leq 0 & & e \in E \backslash \hat{E} \\
& & & r_{ki}(x) \leq M^{ki}(1 - y_{ki}) & & k \in K \backslash \hat{K},\, i \in D_k \\
& & & x = \sum_{\hat{i} \in \hat{D}} \nu^{\hat{i}} & & \\
& & & \hat{y}_{\hat{i}} g_e(\nu^{\hat{i}} / \hat{y}_{\hat{i}}) \leq 0 & & e \in \hat{E},\, \hat{i} \in \hat{D} \\
& & & \hat{y}_{\tilde{i}_s} r_{\hat{k}_s i}(\nu^{\tilde{i}_s} / \hat{y}_{\tilde{i}_s}) \leq 0 & & s = 1, ..., iter,\, i \in \hat{D}_{k_s},\, \tilde{i}_s \in \tilde{D}_{si} \\
& & & y_{\hat{k}_s i} = \sum_{\substack{\hat{i} \in \hat{D} \\ \hat{i}_s = i}} \hat{y}_{\hat{i}} & & s = 1, ..., iter,\, i \in \hat{D}_{k_s} \\
& & & x^{lo} \hat{y}_{\hat{i}} \leq \nu^{\hat{i}} \leq x^{up} \hat{y}_{\hat{i}} & & \hat{i} \in \hat{D} \\
& & & \sum_{i \in D_k} y_{ki} = 1 & & k \in K \\
& & & \sum_{\hat{i}} \hat{y}_{\hat{i}} = 1 & & \\
& & & Hy \geq h & & \\
& & & \hat{y}_{\hat{i}} = 0 & & \hat{i} \in INEAS_{iter} \\
& & & z^{lo} \leq z & & \\
& & & x^{lo} \leq x \leq x^{up} & & \\
& & & x \in \mathbb{R}^n & & \\
& & & y_{ki} \in \{0, 1\} & & k \in K, i \in D_k \\
& & & 0 \leq \hat{y}_{\hat{i}} \leq 1 & & \hat{i} \notin INEAS_{iter}
\end{aligned}
\tag{GDPH}
$$

Step 7. If relaxed $(GDPH) > z^*$, set $z^* = relaxation(GDPH)$, $GDP^* = GDPH$. If the relaxation has not improved after a specified maximum number of iterations, or the GDPH problem size is greater than a specified limit, solve GDP^*. Else, go back to step 3.

(GDPH) is a hybrid reformulation in which the objective function $f(x)$ is the same as in the original formulation. The global constraints that were not selected for the application of a basic step ($e \in E \backslash \hat{E}$) remain unchanged. The disjunctions that were not selected to apply basic steps are reformulated using (BM) ($r_{ki}(x) \leq M^{ki}(1 - y_{ki})$). The disjunctions that were intersected with basic steps now form a single disjunction, which we will denote "key disjunction", and that contains all terms $\hat{i} \in \hat{D}$. The corresponding variable to this new terms is $\hat{y}_{\hat{i}}$. Note that $|\hat{D}| = |\hat{D}_{k_1}| * ... * |\hat{D}_{k_{iter}}|$, which indicates an exponential growth of the disjunction with the number of iterations. The "key disjunction" is reformulated using (HR).

The equation $x = \sum_{\hat{i} \in \hat{D}} \nu^{\hat{i}}$ relates the continuous variables x, to the disaggregated variables in all terms $\nu^{\hat{i}}$. The constraint $\hat{y}_{\hat{i}} g_e(\nu^{\hat{i}}/\hat{y}_{\hat{i}})$ is the (HR) reformulation of the global constraints that were intersected with the "key disjunction" $e \in \hat{E}$. Note that these constraints are present in all terms of the "key disjunction" ($\hat{i} \in \hat{D}$). Equation $\hat{y}_{\tilde{i}_s} r_{\hat{k}_s i}(\nu^{\tilde{i}_s}/\hat{y}_{\tilde{i}_s})$ is the (HR) reformulation of all the constraints in the terms of the disjunctions to which basic steps where applied. Each term of the "key disjunction" contains *iter* sets of constraints, each one related to one of the disjunctions $\hat{k}_s \in \hat{K}$. The original set of constraints in a certain term i of a selected disjunction $\hat{k}_s$ will be present in all terms $\hat{i}$ of the "key disjunction", as long as its corresponding element $\hat{i}_s$ is equal to i ($\tilde{i}_s \in \tilde{D}_{si}$, where $\tilde{D}_{si} = \{\hat{D}_{k_1}, ..., i, ..., \hat{D}_{k_{iter}}\}$ and i is the s^{th} element of $\tilde{D}_{si}$). $\tilde{D}_{si}$ is a map that assigns the constraints in the original disjunctive terms $r_{\hat{k}_s i}$ to the terms $\hat{i}$ in the "key disjunction". Equation $y_{\hat{k}_s i} = \sum_{\substack{\hat{i} \in \hat{D} \\ \hat{i}_s = i}} \hat{y}_{\hat{i}}$ relates the original binary variables y_{ki} to the new variables $\hat{y}_{\hat{i}}$, and it allows the new variables $\hat{y}_{\hat{i}}$ to be continuous while enforcing them to always take a $\{0, 1\}$ value(Balas 1985, Ruiz and Grossmann 2012).

It is should be noted that the algorithm applies basic steps, which are valid logic operations for GDP. Also, the hybrid reformulation is a valid MILP/MINLP representation of the problem, since the (BM) and (HR) are valid reformulations for individual disjunctions. Therefore, the algorithm provides a valid MILP/MINLP representation of the original GDP.

Section 4 provides an illustration of each of the steps in the algorithm. The remaining of this section describes each of the steps of the algorithm with detail.

3.2. Step 1: Pre-solve

The pre-solve has three purposes: eliminate infeasible terms, find better bounds, and provide a value that will characterize each disjunction. This pre-solve can be performed due to the logic nature of GDP formulations. The pre-solve can be regarded as a strong branching over every disjunction, and only in the root node.

The pre-solve procedure is as follows:

0. Set $k = 1$ and $i = 1$

1. Set $Y_{ki} = True$ and, as consequence, $Y_{ki'} = False$ for all $i' \neq i, i' \in D_k$

2. Formulate MILP/MINLP by using the (HR) formulation of the remaining disjunctions $k' \neq k, k' \in K$ (note that, since $Y_{ki} = True$, the constraints associated with Y_{ki} will be

enforced as global constraints, while the ones associated with $Y_{ki'}, i' \neq i, i' \in D_k$ will be removed from the formulation).

3. Solve the continuous relaxation of this problem to optimality, and define z^{ki} as the solution of this problem.

4. Repeat this for every $k \in K$ and $i \in D_k$.

DEFINITION 3.1. For a minimization problem, we define the characteristic value of a disjunction k as follows:

$$charv_k = min\{z^{ki}\}, \forall i \in D_k$$

It is possible to reduce the problem size and find better bounds of the problem considering the following remarks:

REMARK 3.2. $charv_k$ is a lower bound of z.

Proof. Solving the (HR) reformulation of the original GDP, and relaxing all integrality constraints except the ones corresponding to the disjunction k also yields $charv_k$ (i.e. $charv_k$ is the value of the solution of a relaxation of (GDP)) $\square$.

REMARK 3.3. If z^{ki} is $infeasible$ for a disjunctive term $k \in K, i \in D_k$, then $Y_{ki} = False$ in the original formulation.

Proof. From the definition, z^{ki} is a continuous relaxation of the problem with $Y_{ki} = True$. If this relaxation is infeasible, then that particular disjunctive term can not be selected (i.e. $Y_{ki} = False$) $\square$.

As consequence of Remark 3.3, the terms and constraints associated to a $k \in K, i \in D_k$ term such that z^{ki} is $infeasible$ can be removed form the original (GDP) formulation.

REMARK 3.4. If z^{ki} is $infeasible$ for all $i \in D_k$ for any $k \in K$, then the problem is infeasible.

Proof. If for any disjunction $k \in K$, all its disjunctive terms $i \in D_k$ are infeasible, then there is no alternative in that disjunction that will make the problem feasible, and therefore, the problem is $infeasible$ $\square$.

REMARK 3.5. A lower bound to the problem z^{lo} that is at least as large as the continuous relaxation of the original (GDP) can be obtained with as follows:

$$z^{lo} = \max_{k \in K}\{charv_k\}$$

Proof. Trivially follows from 3.2 $\square$.

It is important to note that this process requires the evaluation of $\sum_{k \in K} |D_k|$ LP/NLP problems. Although this preprocessing might be expensive for only eliminating terms and

finding good bounds for the problem, the characteristic value of the disjunctions has a major role in the algorithm as will be described in section 3.3. Additionally, since most of the structure of the problem does not change while evaluating every term, it might be possible to solve the many LP/NLP problems in a more efficient manner. This issue is not addressed in this paper. The resulting (GDP) after eliminating infeasible terms is set as (GDP^*), and its continuous relaxation z^*.

3.2.1. Illustration of pre-solve. In order to illustrate the pre-solve procedure, consider the formulation shown in (1).

$$
\begin{aligned}
& min\, z = x_1 + x_2 \\
s.t.\ & \\
& \begin{bmatrix} Y_{11} \\ x_2 \geq 8 + x_1 \\ x_2 = 12 - x_1 \end{bmatrix} \vee \begin{bmatrix} Y_{12} \\ x_1 \leq 5 \\ x_2 \geq 6 \\ x_2 \leq x_1 + 5 \end{bmatrix} \vee \begin{bmatrix} Y_{13} \\ x_1 \geq 9 \\ x_2 \leq 5 \\ x_2 \geq x_1 - 8 \end{bmatrix} \\
& \begin{bmatrix} Y_{21} \\ 4 \leq x_1 \leq 7 \\ 7 \leq x_2 \leq 8 \end{bmatrix} \vee \begin{bmatrix} Y_{22} \\ 7 \leq x_1 \leq 11 \\ 2 \leq x_2 \leq 4 \end{bmatrix} \\
& Y_{11} \veebar Y_{12} \veebar Y_{13} \\
& Y_{21} \veebar Y_{22} \\
& x_1, x_2 \in \mathbb{R}^1 \\
& Y_{11}, Y_{12}, Y_{13}, Y_{21}, Y_{22} \in \{True, False\}
\end{aligned}
\tag{1}
$$

Problem (1) has an optimal solution of $\mathbf{z = 11}$, in which $Y_{12} = True$ and $Y_{21} = True$. The (BM) provides a relaxation of $\mathbf{z^{BM} = 3}$, and the (HR) a relaxation of $\mathbf{z^{HR} = 9.16}$. Following the pre-solve procedure described in 3.2, we first set $Y_{11} = True, Y_{12} = False, Y_{13} = False$, and then perform the (HR) reformulation of the remaining of the problem. Relaxing the integrality constraints, this yields the LP shown in (2).

$$
\begin{aligned}
& & & min\, z = x_1 + x_2 \\
& s.t. & & x_2 \geq 8 + x_1 \\
& & & x_2 = 12 - x_1 \\
& & & x_1 = (x_1)_{21} + (x_1)_{22} \\
& & & x_2 = (x_2)_{21} + (x_2)_{22} \\
& & & 4 * y_{21} \leq (x_1)_{21} \leq 7 * y_{21} \\
& & & 7 * y_{21} \leq (x_2)_{21} \leq 8 * y_{21} \\
& & & 7 * y_{22} \leq (x_1)_{22} \leq 11 * y_{22} \\
& & & 2 * y_{22} \leq (x_2)_{22} \leq 4 * y_{22} \\
& & & y_{21} + y_{22} = 1 \\
& & & x_1, x_2 \in \mathbb{R}^1 \\
& & & 0 \leq y_{21}, y_{22} \leq 1
\end{aligned}
\tag{2}
$$

Problem (2) is infeasible. Performing the same calculation for $Y_{12} = True$ yields a $z^{12} = 10.6$. Repeating this for the remaining disjunctive terms we obtain $z^{13} = 11$, $z^{21} = 11$, $z^{21} = 9.25$. With these values, we assign the characteristic values for each disjunction:

$charv_1 = min\{z^{11}, z^{12}, z^{13}\} = min\{infeas, 10.6, 11\} = 10.6$

$charv_2 = min\{z^{21}, z^{22}\} = min\{11, 9.25\} = 9.25$

With these two characteristic values, it is possible to set the new lower bound: $z^{lo} = max\{charv_1, charv_2\} = 10.6$. Also, since $z^{11} = infeas$, the term associated with Y_{11} can be eliminated from the original (GDP) formulation. Therefore, problem (1) after the pre-solve becomes (3).

$$
\begin{aligned}
& min\, z = x_1 + x_2 \\
s.t. \\
& \begin{bmatrix} Y_{12} \\ x_1 \leq 5 \\ x_2 \geq 6 \\ x_2 \leq x_1 + 5 \end{bmatrix} \vee \begin{bmatrix} Y_{13} \\ x_1 \geq 9 \\ x_2 \leq 5 \\ x_2 \geq x_1 - 8 \end{bmatrix} \\
& \begin{bmatrix} Y_{21} \\ 4 \leq x_1 \leq 7 \\ 7 \leq x_2 \leq 8 \end{bmatrix} \vee \begin{bmatrix} Y_{22} \\ 7 \leq x_1 \leq 11 \\ 2 \leq x_2 \leq 4 \end{bmatrix} \qquad (3) \\
& Y_{11} \veebar Y_{12} \veebar Y_{13} \\
& Y_{21} \veebar Y_{22} \\
& x_1, x_2 \in \mathbb{R}^1 \\
& Y_{11}, Y_{12}, Y_{13}, Y_{21}, Y_{22} \in \{True, False\}
\end{aligned}
$$

It is clear that (3) is a smaller problem than the original problem (1), and a new lower bound of $\mathbf{z^{lo} = 10.6}$ has been found. Note that the solution to the problem is $\mathbf{z = 11}$, and the (HR) of (1) provides a lower bound of $\mathbf{z^{HR} = 9.16}$, so the new lower bound $\mathbf{z^{lo} = 10.6}$ is stronger.

3.3. Step 2, 3: Selection of k^*

The algorithm selects over which disjunctions to apply basic steps (k^*). As described by Balas(Balas 1985) there are heuristics to estimate which basic step will generate the best improvement on a formulation. Furthermore, even if the best first basic step were selected, this does not necessarily imply that after a sequence of basic steps this first one is the best choice. Sawaya and Grossmann(Sawaya and Grossmann 2012) and Ruiz and Grossmann(Ruiz and Grossmann 2012) propose some heuristics as to when to apply basic steps. For the algorithm proposed in this paper, we consider three main factors for the selection of these disjunctions:

1) A consequence of basic steps described in Balas' work(Balas 1985): A basic step between two disjunctions that do not share variables in common will not improve the tightness of the formulation.

2) The number of terms in the new disjunction increases exponentially with the number of terms of the selected disjunctions ($|\hat{D}| = |\hat{D}_{k_1}| * ... * |\hat{D}_{k_{iter}}|$). Since the algorithm applies

basic steps iteratively over the same disjunction, it is desired to keep the problem size as small as possible.

3) The characteristic value of the disjunctions, obtained in the first step of the algorithm, provides a heuristic on the "tightness" associated which each disjunction. Therefore, disjunctions with higher characteristic values are preferred.

There are several heuristics that can be used for the selection of k^*. In our experience, the best performance was achieved by applying many basic steps with small growth in size. Therefore, the heuristic we found to work the best was to select disjunctions with fewest terms that share variables in common with many other disjunctions (preferably also small ones). This evaluation is done as follows:

0. Initialize $W^k = 0, \forall k \in K$. Set $m = 1$, $n = 2$.

1. If disjunction m shares a variable in common with disjunction n, then $W^m = W^m + \frac{1}{(|D_m|)*(|D_n|)}$; $W^n = W^n + \frac{1}{(|D_m|)*(|D_n|)}$.

2. Set $n = n + 1$. If $n \leq |K|$ go back to 1, else go to 3.

3. Set $m = m + 1$. If $m \leq |K| - 1$ set $n = m + 1$ and go back to 1, else go to 4.

4. Select the disjunction k^* with largest W^k value. If there is a tie, choose the disjunction with the highest characteristic value.

Note that this algorithm gives priority to the number of basic steps that can be applied to a certain disjunction, giving more weight to the basic steps with smaller increase in number of disjunctive terms. If there is a tie with this parameter, it selects the one with highest $charv_k$.

The method for the selection of k^* in step 3 is almost the same as the one for the selection of k^* in step 2. The difference is that a basic step will be applied between this disjunction k^* and "key disjunction". Therefore, if a disjunction m does not share a variable in common with any $\hat{k}_s \in \hat{K}$ then $W^m = 0$.

3.4. Step 4: Analyze and eliminate resulting disjunctive terms

As shown in Section 3.1, the basic steps will be applied iteratively over the same disjunction. This means that the number of terms of such a disjunction will grow exponentially after each basic step. Eliminating terms after each basic step helps to maintain small formulations. However, there are two things to consider: first is that this step can become computationally expensive as the algorithm iterates, and second is that eliminating terms

has less impact after each iteration. There are two methods that the algorithm uses to eliminate infeasible terms.

First, the terms that result from intersecting a term that was infeasible in the previous iteration, will be infeasible. Second, in order to eliminate resulting disjunctive terms, one LP/NLP is solved for each term in the new disjunction, similarly to the pre-solve. In the initial iterations, $|\hat{D}|$ is small, so the "key disjunction" has a small number of terms. Furthermore, every term that is eliminated reduces the exponential growth of the problem size in the subsequent iterations. As the algorithm iterates and $|\hat{D}|$ becomes larger, the number of LP/NLP evaluation increases, each LP/NLP becomes more expensive, and there will not be many more basic steps, so its impact is limited. For these reasons, step 4 is only applied to the initial iterations, but dropped as the algorithm progresses.

In addition to this, a third method that identifies infeasibility implied by the logic expressions when two of the disjunctive terms are intersected can be used. Such cases are computationally cheap, but there are not necessarily many terms that can be eliminated in this way. Tools such as constraint programming can further improve the performance of this method(Hooker 2002). This third method is not addressed in this paper.

3.5. Step 5

In step 5 an improper basic step is applied between the "key disjunction" and the global constraints. Using the same concepts described in 3.3, the improper basic step is only applied with the global constraints that share at least one variable in common with any $\hat{k}_s \in \hat{K}$.

3.6. Step 6: Hybrid reformulation of (GDP)

In 2.1.2 the MILP/MINLP reformulation of the (GDP) is described through either (BM) or (HR). The reformulation, however, needs not to be strictly one of these; some disjunctions can be reformulated through (BM) while others (HR). The advantage of doing this is that, if the correct disjunctions are selected for the different reformulations, we can obtain a tight relaxation but with a smaller problem size than (HR). Note that in the hybrid reformulation the smallest possible MILP/MINLP is the complete (BM), while the tightest continuous relaxation is achieved through the (HR). Any hybrid lies between the two formulations, both in size of problem and tightness of continuous relaxation.

For the algorithm, we apply convex hull to the "key disjunction" (i.e. we formulate it using (HR)) since the tightness improvement of the basic steps does not hold true for the

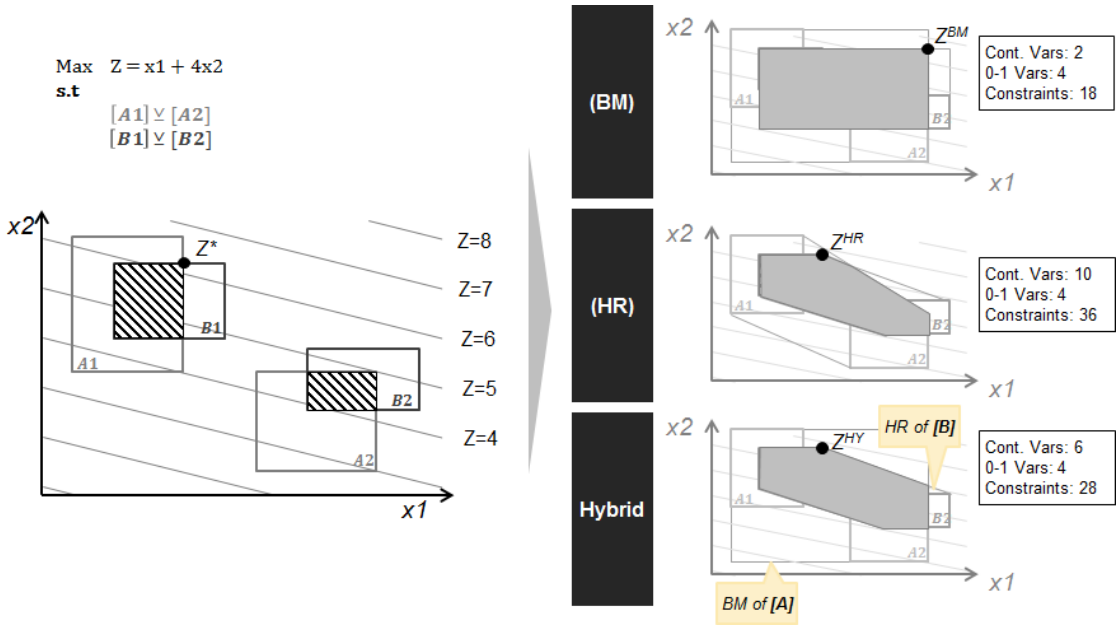

Figure 5 **Illustration of (BM), (HR) and Hybrid reformulation**

(BM). The rest of the disjunctions are formulated through (BM), considering that the tightness improvement comes from the basic steps applied in disjunctions $\hat{k}_s \in \hat{K}$.

Figure 5 illustrates the idea of the hybrid reformulation. In the (BM) reformulation the problem size is small, but the continuous relaxation (represented by the shaded region) provides a solution Z^{BM} that is far from the optimal solution Z^*. The (HR) provides a tighter continuous relaxation, and therefore the solution to the relaxation Z^{HR} is closer to the optimal solution. As expected, the problem size is considerably larger than (BM). Lastly, in the hybrid reformulation the continuous relaxation is not as tight as the (HR). However, its relaxation provides the same optimal solution as the relaxation of the (HR) $Z^{HY} = Z^{HR}$. This hybrid reformulation is larger than the (BM), but not as large as (HR).

3.7. Step 7: Rule for iterating

Many different rules can be applied to decide whether or not the algorithm keeps iterating. The most intuitive rules are size of the problem and value of continuous relaxation, since these are the two properties we are trying to improve. The last formulation that was considered to improve the GDP (GDP^*), is the one that is then solved as an MILP/MINLP.

It is also important to note that, in order to identify if the relaxation is improving or not, we check the relaxation of ($GDPH$) and (GDP^*) without using the lower bound found in the pre-solve z^{lo}. This lower bound is added in the last iteration, when (GDP^*) is solved as an MILP/MINLP

4. Illustration of algorithm

In this section we provide an illustration of the algorithm with a simple example. Consider the linear (GDP) in (4) that corresponds to a strip packing problem (see the on-line supplement):

$$
\begin{aligned}
& min\, lt \\
s.t.\quad & lt \geq x_1 + 6 \\
& lt \geq x_2 + 5 \\
& lt \geq x_3 + 4 \\
& lt \geq x_4 + 3 \\
& \begin{bmatrix} Y_{11} \\ x_1 + 6 \leq x_2 \end{bmatrix} \vee \begin{bmatrix} Y_{12} \\ x_2 + 5 \leq x_1 \end{bmatrix} \vee \begin{bmatrix} Y_{13} \\ h_1 - 6 \geq h_2 \end{bmatrix} \vee \begin{bmatrix} Y_{14} \\ h_2 - 7 \geq h_1 \end{bmatrix} \\
& \begin{bmatrix} Y_{21} \\ x_1 + 6 \leq x_3 \end{bmatrix} \vee \begin{bmatrix} Y_{22} \\ x_3 + 4 \leq x_1 \end{bmatrix} \vee \begin{bmatrix} Y_{23} \\ h_1 - 6 \geq h_3 \end{bmatrix} \vee \begin{bmatrix} Y_{24} \\ h_3 - 5 \geq h_1 \end{bmatrix} \\
& \begin{bmatrix} Y_{31} \\ x_1 + 6 \leq x_4 \end{bmatrix} \vee \begin{bmatrix} Y_{32} \\ x_4 + 3 \leq x_1 \end{bmatrix} \vee \begin{bmatrix} Y_{33} \\ h_1 - 6 \geq h_4 \end{bmatrix} \vee \begin{bmatrix} Y_{34} \\ h_4 - 3 \geq h_1 \end{bmatrix} \\
& \begin{bmatrix} Y_{41} \\ x_2 + 5 \leq x_3 \end{bmatrix} \vee \begin{bmatrix} Y_{42} \\ x_3 + 4 \leq x_2 \end{bmatrix} \vee \begin{bmatrix} Y_{43} \\ h_2 - 7 \geq h_3 \end{bmatrix} \vee \begin{bmatrix} Y_{44} \\ h_3 - 5 \geq h_2 \end{bmatrix} \\
& \begin{bmatrix} Y_{51} \\ x_2 + 5 \leq x_4 \end{bmatrix} \vee \begin{bmatrix} Y_{52} \\ x_4 + 3 \leq x_2 \end{bmatrix} \vee \begin{bmatrix} Y_{53} \\ h_2 - 7 \geq h_4 \end{bmatrix} \vee \begin{bmatrix} Y_{54} \\ h_4 - 3 \geq h_2 \end{bmatrix} \\
& \begin{bmatrix} Y_{61} \\ x_3 + 4 \leq x_4 \end{bmatrix} \vee \begin{bmatrix} Y_{62} \\ x_4 + 3 \leq x_3 \end{bmatrix} \vee \begin{bmatrix} Y_{63} \\ h_3 - 5 \geq h_4 \end{bmatrix} \vee \begin{bmatrix} Y_{64} \\ h_4 - 3 \geq h_3 \end{bmatrix} \\
& Y_{k1} \veebar Y_{k2} \veebar Y_{k3} \veebar Y_{k4} && k = 1, ..., 6 \\
& 0 \leq x_1 \leq 12;\ 0 \leq x_2 \leq 13;\ 0 \leq x_3 \leq 14;\ 0 \leq x_4 \leq 15 \\
& 6 \leq h_1 \leq 10;\ 7 \leq h_2 \leq 10;\ 5 \leq h_3 \leq 10;\ 3 \leq h_4 \leq 10 \\
& x_j, h_j \in \mathbb{R}^1 && j = 1, 2, 3, 4 \\
& Y_{ki} \in \{True, False\} && k = 1, ..., 6, i = 1, 2, 3, 4
\end{aligned}
\tag{4}
$$

This problem has an optimal solution of $\mathbf{lt = 15}$, in which $Y_{12}, Y_{21}, Y_{32}, Y_{41}, Y_{53}, Y_{63} = True$. The continuous relaxation of its (BM) reformulation has a value of $\mathbf{z^{BM} = 6.0}$, and the (HR) provides a relaxation of $\mathbf{z^{HR} = 8.3}$.

Step 1. After applying the pre-solve as described in 3.2, the terms $\{13, 14, 23, 24, 43, 44\}$ are found to be $infeasible$, so they are removed from the original (GDP) formulation. GDP^* is then (4) without terms $\{13, 14, 23, 24, 43, 44\}$. Also, the characteristic values of

Table 1 Weight parameter calculation for the disjunctions

			Accumulated weight					
m	n	common vars?	W^1	W^2	W^3	W^4	W^5	W^6
1	2	yes	0.25	0.25	0	0	0	0
1	3	yes	0.375	0.25	0.125	0	0	0
1	4	yes	0.625	0.25	0.125	0.25	0	0
1	5	yes	0.75	0.25	0.125	0.25	0.125	0
1	6	no	0.75	0.25	0.125	0.25	0.125	0
2	3	yes	0.75	0.375	0.125	0.25	0.125	0
2	4	yes	0.75	0.625	0.25	0.5	0.125	0
2	5	no	0.75	0.625	0.25	0.5	0.125	0
2	6	yes	0.75	0.75	0.25	0.5	0.125	0.125
3	4	no	0.75	0.75	0.25	0.5	0.125	0.125
3	5	yes	0.75	0.75	0.3125	0.5	0.1875	0.125
3	6	yes	0.75	0.75	0.375	0.5	0.1875	0.1875
4	5	yes	0.75	0.75	0.375	0.625	0.3125	0.1875
4	6	yes	0.75	0.75	0.375	0.75	0.3125	0.3125
5	6	yes	0.75	0.75	0.375	0.75	0.375	0.375

the disjunctions $k = 1, ..., 6$ are, respectively: $charv_k = (11, 10, 8.3, 9.6, 8.3, 8.3)$. A new lower bound z^{lo} is also found, since $max(charv_k) = 11$, which is larger than the relaxation of the (HR) $z^* = 8.3$. Set $iter = 1$.

Step 2. The selection of k^* is performed by assigning a weight to each disjunction as described in 3.3. In this case $W = (0.75, 0.75, 0.38, 0.75, 0.38, 0.38)$. Table 1 shows the iterations to obtain W, following the procedure described in section 3.3. Disjunctions 1, 2 and 4 have the same weight, but the first disjunction has the highest $charv_k$. Therefore, disjunction 1 is chosen as k^*. $\hat{k_1} = 1$; $\hat{K} = \{1\}$; $\hat{D}_1 = \{1, 2\}$.

Step 3. $iter = 2$. To find new k^*, weighting factors W^k for $k \neq 1$ are also assigned $W = (1, 0.5, 1, 0.5, 0)$. Disjunction 2 and 4 have the same W^k, but since $charv_2 > charv_4$, disjunction 2 is selected as k^*. Also note that $W^6 = 0$ since disjunction 6 and disjunction 1 do not share variables in common.

Set $\hat{k}_2 = 2$; $\hat{K} = \{1, 2\}$; $\hat{D}_2 = \{1, 2\}$; $\hat{D} = \{(1, 1), (1, 2), (2, 1), (2, 2)\}$.

For this step, it is possible to represent the "key disjunction" as follows:

$$\begin{bmatrix} \hat{Y}_{1,1} \\ x_1+6\leq x_2 \\ x_1+6\leq x_3 \end{bmatrix} \vee \begin{bmatrix} \hat{Y}_{1,2} \\ x_1+6\leq x_2 \\ x_3+4\leq x_1 \end{bmatrix} \vee \begin{bmatrix} \hat{Y}_{2,1} \\ x_2+5\leq x_1 \\ x_1+6\leq x_3 \end{bmatrix} \vee \begin{bmatrix} \hat{Y}_{2,2} \\ x_2+5\leq x_1 \\ x_3+4\leq x_1 \end{bmatrix} \tag{5}$$

The additional constraints that are added so that the new disjunctive variables $\hat{y}_{\hat{i}}$ can be continuous are shown in (6):

$$\begin{aligned} & y_{11}=\hat{y}_{1,1}+\hat{y}_{1,2} \\ & y_{12}=\hat{y}_{2,1}+\hat{y}_{2,2} \\ & y_{21}=\hat{y}_{1,1}+\hat{y}_{2,1} \\ & y_{22}=\hat{y}_{1,2}+\hat{y}_{2,2} \\ & y_{11},y_{12},y_{21},y_{22}\in\{0,1\} \\ & 0\leq \hat{y}_{1,1},\hat{y}_{1,2},\hat{y}_{2,1},\hat{y}_{2,2}\leq 1 \end{aligned} \tag{6}$$

Step 4. All the resulting terms in disjunction (5) are feasible, so $INFEAS_2=\emptyset$.

Step 5. Select global equations to which apply a basic step $\hat{E}\in E$. In the example, the first three global constraints share a variable in common with the "key disjunction". It is possible to represent the resulting disjunction after the application of the improper basic step as shown in (7).

$$\begin{bmatrix} \hat{Y}_{1,1} \\ lt\geq x_1+6 \\ lt\geq x_2+5 \\ lt\geq x_3+4 \\ x_1+6\leq x_2 \\ x_1+6\leq x_3 \end{bmatrix} \vee \begin{bmatrix} \hat{Y}_{1,2} \\ lt\geq x_1+6 \\ lt\geq x_2+5 \\ lt\geq x_3+4 \\ x_1+6\leq x_2 \\ x_3+4\leq x_1 \end{bmatrix} \vee \begin{bmatrix} \hat{Y}_{2,1} \\ lt\geq x_1+6 \\ lt\geq x_2+5 \\ lt\geq x_3+4 \\ x_2+5\leq x_1 \\ x_1+6\leq x_3 \end{bmatrix} \vee \begin{bmatrix} \hat{Y}_{2,2} \\ lt\geq x_1+6 \\ lt\geq x_2+5 \\ lt\geq x_3+4 \\ x_2+5\leq x_1 \\ x_3+4\leq x_1 \end{bmatrix} \tag{7}$$

Step 6. The hybrid reformulation of this example is performed by applying (HR) to the "key disjunction" and (BM) in the remaining disjunctions. This MILP can be found in the on-line supplement, formulation (4H).

Step 7. The relaxed solution of (4H) is 11. Since relaxed (4H)$>$ 8.3, set $z^*=11$, GDP^*=(4H). Note that, as explained in section 3.7, the lower bound found in the pre-solve ($z^{lo}=11$) is not used to evaluate the improvement in the formulation. This lower bound will be added only in the last iteration when (GDP) is solved as MILP/MINLP.

Since it improved, the algorithm proceeds to the next iteration.

In the next iteration, a basic step between the "key disjunction" and disjunction 4 is applied. The resulting disjunction is illustrated in (8).

$$\begin{bmatrix} \hat{Y}_{1,1,1} \\ lt \geq x_1 + 6 \\ lt \geq x_2 + 5 \\ lt \geq x_3 + 4 \\ x_1 + 6 \leq x_2 \\ x_1 + 6 \leq x_3 \\ x_2 + 5 \leq x_3 \end{bmatrix} \vee \begin{bmatrix} \hat{Y}_{1,1,2} \\ lt \geq x_1 + 6 \\ lt \geq x_2 + 5 \\ lt \geq x_3 + 4 \\ x_1 + 6 \leq x_2 \\ x_1 + 6 \leq x_3 \\ x_3 + 4 \leq x_2 \end{bmatrix} \vee \begin{bmatrix} \hat{Y}_{1,2,1} \\ lt \geq x_1 + 6 \\ lt \geq x_2 + 5 \\ lt \geq x_3 + 4 \\ x_1 + 6 \leq x_2 \\ x_3 + 4 \leq x_1 \\ x_2 + 5 \leq x_3 \end{bmatrix} \vee \begin{bmatrix} \hat{Y}_{1,2,2} \\ lt \geq x_1 + 6 \\ lt \geq x_2 + 5 \\ lt \geq x_3 + 4 \\ x_1 + 6 \leq x_2 \\ x_3 + 4 \leq x_1 \\ x_3 + 4 \leq x_2 \end{bmatrix} \vee$$
$$\vee \begin{bmatrix} \hat{Y}_{2,1,1} \\ lt \geq x_1 + 6 \\ lt \geq x_2 + 5 \\ lt \geq x_3 + 4 \\ x_2 + 5 \leq x_1 \\ x_1 + 6 \leq x_3 \\ x_2 + 5 \leq x_3 \end{bmatrix} \vee \begin{bmatrix} \hat{Y}_{2,1,2} \\ lt \geq x_1 + 6 \\ lt \geq x_2 + 5 \\ lt \geq x_3 + 4 \\ x_2 + 5 \leq x_1 \\ x_1 + 6 \leq x_3 \\ x_3 + 4 \leq x_2 \end{bmatrix} \vee \begin{bmatrix} \hat{Y}_{2,2,1} \\ lt \geq x_1 + 6 \\ lt \geq x_2 + 5 \\ lt \geq x_3 + 4 \\ x_2 + 5 \leq x_1 \\ x_3 + 4 \leq x_1 \\ x_2 + 5 \leq x_3 \end{bmatrix} \vee \begin{bmatrix} \hat{Y}_{2,2,2} \\ lt \geq x_1 + 6 \\ lt \geq x_2 + 5 \\ lt \geq x_3 + 4 \\ x_2 + 5 \leq x_1 \\ x_3 + 4 \leq x_1 \\ x_3 + 4 \leq x_2 \end{bmatrix} \tag{8}$$

In this case, the term associated with $\hat{Y}_{1,2,1}$ and $\hat{Y}_{2,1,2}$ are infeasible, so $INFEAS_2 = \{(1,2,1),(2,1,2)\}$.

Also, additional constraints need to be added in the MILP reformulation to avoid the increase in binary variables, as described in earlier. These constraints (after setting $\hat{y}_{1,2,1} = \hat{y}_{2,1,2} = 0$;) are shown in (9).

$$\begin{aligned} y_{11} &= \hat{y}_{1,1,1} + \hat{y}_{1,1,2} + \hat{y}_{1,2,2} \\ y_{12} &= \hat{y}_{2,1,1} + \hat{y}_{2,2,1} + \hat{y}_{2,2,2} \\ y_{21} &= \hat{y}_{1,1,1} + \hat{y}_{1,1,2} + \hat{y}_{2,1,1} \\ y_{22} &= \hat{y}_{1,2,2} + \hat{y}_{2,2,1} + \hat{y}_{2,2,2} \\ y_{41} &= \hat{y}_{1,1,1} + \hat{y}_{2,1,1} + \hat{y}_{2,2,1} \\ y_{42} &= \hat{y}_{1,1,2} + \hat{y}_{1,2,2} + \hat{y}_{2,2,2} \\ y_{11}, y_{12}, y_{21}, y_{22}, y_{41}, y_{42} &\in \{0,1\} \\ 0 \leq \hat{y}_{1,1,1}, \hat{y}_{1,1,2}, \hat{y}_{1,2,2}, \hat{y}_{2,1,1}, \hat{y}_{2,2,1}, \hat{y}_{2,2,2} &\leq 1 \end{aligned} \tag{9}$$

The continuous relaxation of the hybrid MILP reformulation of this iteration is $\mathbf{z^{iter=2} = 15}$. Since it improved, the algorithm proceeds to another iteration.

The third iteration involves a basic step between the "key disjunction" and disjunction 5. This results in a disjunction with 32 terms, of which 8 of them are infeasible (the 8 terms that result from intersecting terms $\{(1,2,1),(2,1,2)\}$ with the 4 terms of disjunction 5). This ($GDPH$) also has a relaxation of $\mathbf{z^{iter=3} = 15}$. Since there is no improvement, the formulation obtained in iteration 2 is selected as GDP^* and solved as MILP. Note that the continuous relaxation of this formulation provides a lower bound that has the same value as the optimal solution of the problem $z^{iter=2} = z^* = 15$. However, the values for y_{ki} are not integer, so the MILP solver requires to evaluate some nodes to find the integer solution.

5. Results

In this section we present the computational results of applying the algorithm described in section 3 to different instances of the examples that can be found in the on-line supplement. The algorithm uses the heuristics for selecting k^* described in 3.3. There are many rules that could be set for deciding if the algorithm moves to the next iteration or not. In this study the rule that was used is as follows. If the continuous relaxation does not improve after 3 iterations, or the number of constraints is more than double than the original, or the number of disjunctive terms in D^k is larger than half of the number of original total disjunctive terms, then proceed to solve GDP^*; else keep iterating. Note that a formulation can in fact more than double its size, but that can occur only in the last iteration. Afterwards the algorithm proceeds to the next step.

Thirty six instances were solved. The instances were generated by defining problem size and randomly generating the parameters of the problems (e.g. in Stpck the number of rectangles was set, but width and length of the rectangles was randomly generated). The ϵ has a value of 10^{-4}, and the Big M parameter was estimated using the most basic solution for the given data (e.g. in the strip packing problem the most basic solution is to pack one rectangle after the other, though is not the optimal). The only exception are Batch instances, where the "optimal" Big M parameter was used, and which can be found in the CMU-IBM MINLP library(CMU and IBM 2013).

The instances are solved using branch and bound methods, Gurobi 5.5 for the linear GDP problems and SBB for the convex GDP with CONOPT as the NLP solver. Cuts and presolve were deactivated in Gurobi for all linear instances. The algorithm and models were implemented in GAMS(Brooke et al. 1998) and solved in an Intel(R) Core(TM) i7 CPU 2.93 GHz and 4 GB of RAM.

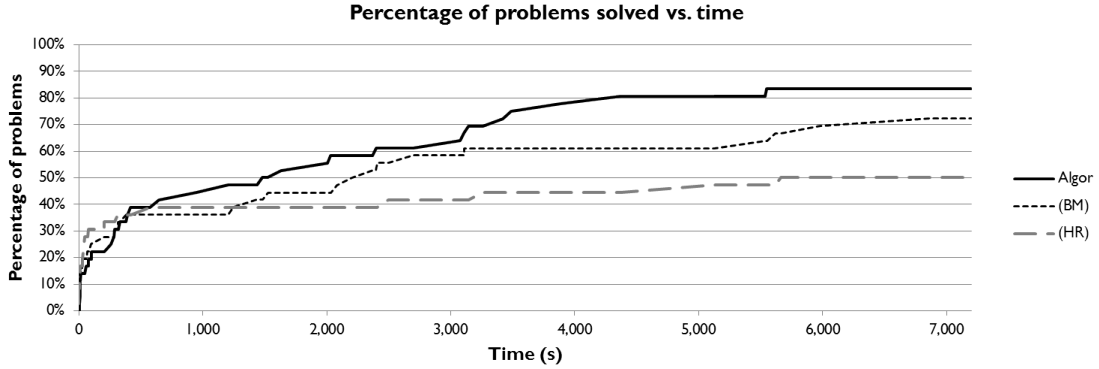

Figure 6 Percentage of problems solved vs. time for the test instances

We first present and discuss the general performance of the algorithm shown in Figures 6, 7 and 8. We then discuss some characteristics of the different instances presented in Figures 9 and 10. Finally, we describe the behaviour of the algorithm in Figure 11.

Figure 6 shows the percentage of problems solved vs. time for the (BM), (HR) and the algorithm. The time for the algorithm includes the preprocessing, the steps for improving the formulation, and the solution to the MINLP. The figure shows that the algorithm performs in general better than the (BM) and (HR) reformulations. In the smaller instances this is not true, and one of the main reasons for this is that the algorithm is programmed with a high level language (GAMS). Thus, if the algorithm is implemented in a lower lever programming language its performance is expected to improve. Other improvements to the algorithm, such as the ones mentioned earlier in the paper (taking advantage of problem structure to reduce presolving time, using constraint programming and logic concepts to eliminate infeasible terms, and improving heuristics for iterating rule and selection of basic steps) might further improve the algorithm's performance.

Figure 7 shows the number of nodes that where evaluated to find and prove optimality (within 0.1 % gap) for the 17 instances in which the three formulations did so. As expected the (HR) requires fewer nodes than the (BM) to achieve optimality. The algorithm needs fewer number of nodes than the (BM), but more than the (HR) in these 17 instances.

Figure 8 shows the percentage of problems vs. number of constraints. The figure shows that the number of constraints in the (BM) is in general smaller than (HR) and the formulation obtained through the algorithm. The MILP/MINLP obtained with the algorithm has fewer constraints than the (HR). It is important to note that for most problems the continuous relaxation improved after applying the algorithm, as shown in Figure 9.

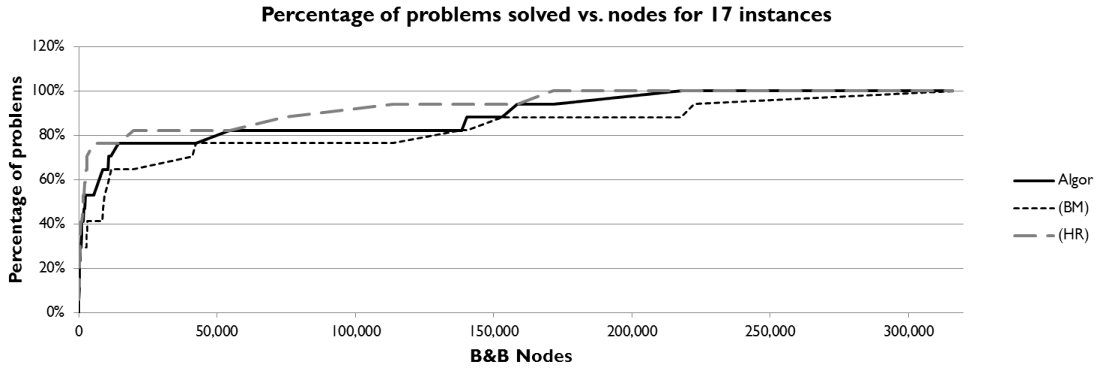

Figure 7 **Number of nodes evaluated to achieve optimum, for the 17 instances where the three formulations did so. Excludes S-Pck12 for comparison purposes, in order to avoid plotting over 1^6 nodes**

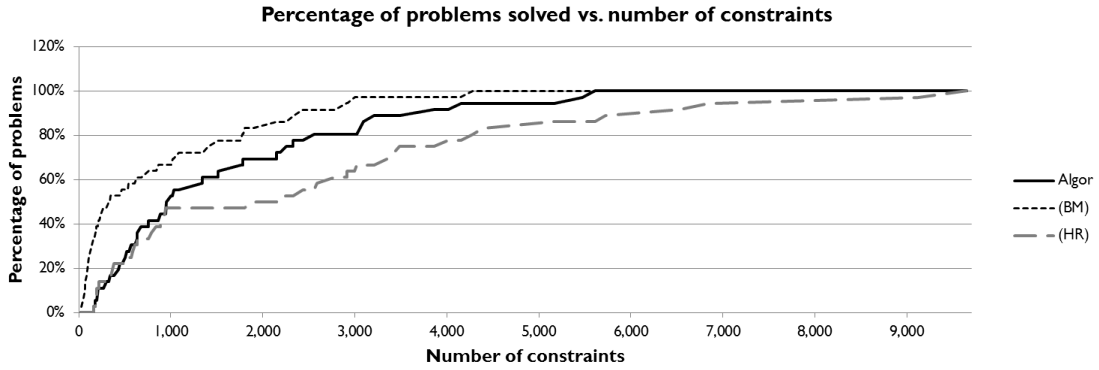

Figure 8 **Number of constraints for the different formulations**

Therefore, having not much larger or even smaller problem sizes represents an important improvement in the problem formulation.

Figure 9 shows the continuous relaxation, the number of constraints, and the number of variables for the different instances, comparing the (BM) and (HR) formulations of the original problem with the formulation obtained with the proposed algorithm. It can be seen that on 28 of the 36 instances the relaxation improves after applying the algorithm, in the other 8 is has the same value as the (HR). In few cases, such as C-Lay-3-2 (where the solution is **41,573**, the (BM) and (HR) relaxations are **0**, and the relaxation after the algorithm is **2,200**), the improvement in the relaxation is small. In most of the cases the gap improves around 20%-40%, for example C-Lay-5-2 where the solution is **11,472**, the (BM) and (HR) relaxations are **0** and the relaxation after the algorithm is **4,203**. In the extreme case of Process-8, the algorithm provides a relaxation of **1,098**, which is actually

		Relaxation			Constraints			Variables/binary		
Instance	**Solution**	**(BM)**	**(HR)**	**Algorithm**	**(BM)**	**(HR)**	**Algor**	**(BM)**	**(HR)**	**Algor**
Batch101006	769,440	734,943	745,909	**750,769**	1,020	**1,897**	1,004	279/129	**789/129**	276/129
Batch121208	1,241,126	1,202,365	1,217,603	**1,223,599**	1,512	**2,781**	1,514	407/203	**1163/203**	415/203
Batch151208	1,543,472	1,499,913	1,514,948	**1,519,320**	1,782	**3,348**	1,784	446/203	**1334/203**	454/203
Batch181210	2,042,327	2,006,860	**2,021,546**	**2,021,546**	2,148	**4,011**	2,150	533/251	**16001/251**	543/251
Batch201210	2,295,349	2,255,304	2,272,082	**2,277,093**	2,328	**4,389**	2,330	559/251	**1715/251**	569/251
C-Lay-3-2	41,573	0	0	**2,200**	57	195	**568**	32/18	92/18	**240/18**
C-Lay-3-3	26,669	0	0	**2,200**	69	219	**338**	35/21	101/21	**139/20**
C-Lay-4-2	8,469	0	0	**2,769**	93	349	**952**	54/32	166/32	**390/32**
C-Lay-4-3	9,746	0	0	**3,988**	109	381	**1,030**	58/36	178/36	**394/29**
C-Lay-5-2	11,472	0	0	**4,203**	138	530	**1,345**	82/50	262/50	**546/50**
C-Lay-5-3	20,799	0	0	**5,372**	158	588	**1,341**	87/55	277/55	**551/53**
C-Lay-5-4	10,876	0	0	**2,695**	178	608	**952**	92/60	292/60	**392/59**
C-Lay-6-2	11,914	0	0	**2,986**	192	**792**	503	116/72	**380/72**	240/67
C-Lay-6-3	14,523	0	0	**3,546**	216	**840**	518	122/78	**398/78**	246/73
C-Lay-6-4	15,084	0	0	**3,792**	240	888	**1,755**	128/84	416/84	**720/74**
C-Lay-6-5	26,521	0	0	**3,660**	264	**936**	883	134/90	**434/90**	382/82
Dice0605	12	2	**6**	**6**	2,432	**5,162**	3,090	1292/1260	**2912/1260**	2000/1051
Dice0606	12	2	**6**	**6**	1,802	**6,842**	3,090	1766/1728	**3926/1728**	2002/1051
Dice0804	24	4	**8**	**8**	2,914	**6,530**	3,860	1570/1536	**3618/1536**	2676/1249
Dice0805	21	3	**8**	**8**	4,282	**9,122**	5,468	2282/2240	**5162/2240**	3544/1881
Dice1203	56	10	**15**	**15**	2,998	**9,650**	5,610	2198/2160	**5222/2160**	3892/1690
DiceH0605	12	3	**6**	**6**	632	**2,432**	444	1292/1260	**2012/1260**	1294/1250
DiceH0606	12	3	**6**	**6**	758	**2,918**	2,253	1766/1728	**2630/1728**	2766/1718
DiceH0804	24	5	**8**	**8**	866	**3,426**	2,556	1570/1536	**2594/1536**	2706/1522
DiceH0805	21	4	**8**	**8**	1,082	**4,282**	3,205	2282/2240	**3562/2240**	3706/2226
DiceH1203	56	12	**15**	**15**	1,406	**5,726**	4,149	2198/2160	**3926/2160**	4042/2138
F-lay-3	49	31	31	**42**	27	**195**	160	28/12	**124/12**	88/12
F-lay-4	20	11	12	**15**	45	381	**632**	44/24	236/24	**336/24**
F-lay-5	63	35	35	**46**	68	**633**	179	64/40	**389/40**	121/40
F-lay-6	80	37	38	**52**	102	**942**	208	94/60	**574/60**	146/60
Process-12	1,250	0	1,227	**1,248**	121	188	**635**	71/12	151/12	**431/11**
Process-8	1,098	0	1,079	**1,098**	84	163	**201**	48/8	116/8	**120/7**
S-Pck12	35	9	13	**17**	344	**2,192**	299	290/264	**1346/264**	298/204
S-Pck13	59	10	14	**28**	327	**2,589**	675	340/312	**1588/312**	481/202
S-Pck14	48	9	13	**25**	471	**3,019**	759	394/364	**1850/364**	538/272
S-Pck15	40	10	12	**15**	542	**3,482**	433	452/420	**3482/2132**	460/296

Figure 9 Relaxation, number of constraints and variables for the (BM), (HR) and algorithm formulations

the optimal value of the objective function, (HR) provides a good relaxation of **1,079**, but still with some gap, while (BM) provides a very poor relaxation of **0**.

The algorithm produces the largest number of variables and constraints in 11 out of the 36 instances, and the size of these problems is not much larger than that of the (HR). The (HR) produces the largest formulations in the rest of the instances. The number of binary variables in (HR) and (BM) is the same as expected. However, there are fewer binary variables in some of the formulations derived from the algorithm. In these cases, the algorithm is able to eliminate some disjunctive terms during the first pre-solving step, so that the binary variables associated with these terms are removed from the formulation.

Figure 10 shows solution time, optimality gap and number of nodes evaluated in the branch and bound tree. Note that the proposed algorithm requires fewest number of nodes in 18 out of the 30 instances in which the number of nodes can be compared. The performance is measured as solution time, or gap in the cases where the models did not find and prove the optimal solution after two hours. The algorithm performs the best in 15 of the

36 instances, and worst in 12 instances. However, the algorithm performs relatively close to the best performer in all instances, except in Batch151208. For the Batch problems the (HR) formulation performs the best. The (BM) and (HR) perform similarly in the first two instances. The third one is the only instance in all test problems in which the algorithm does much worse than the (BM) and (HR). In the last two instances of the Batch problems the algorithm performs much better than the (BM). For the C-Lay problems the (BM) generally performs the best in the smaller instances (C-Lay-3-2 to C-Lay-5-4). In the larger instances the algorithm performs better than the (BM) and (HR). In Dice and DiceH the algorithm performs the best, while the (HR) performs the worst. For the F-Lay problem the (BM) performs the best, while the (HR) performs the worst. For Process (HR) is the best formulation, and the algorithm the worst, but note that the time to solve is very fast, so the presolve and rest of the algorithm takes much more time than actually solving the MINLP. For the S-Pck problems the (BM) performs the best, then the algorithm, and the (HR) the worst. Note that in C-Lay-6-2, C-Lay-6-3, and the larger instances of Dice and DiceH the (BM) and (HR) reformulations do not solve to optimality within two hours while the algorithm does. It is also important to note that in the examples related to process design (Process and Batch) the (HR) performs much better than the (BM), while in the packing (C-Lay, F-Lay and S-Pck) and Dice the (BM) is much better than the (HR).

Figure 11 shows the behaviour of the algorithm. The first three columns show the time that the algorithm spends in applying the pre-solve, performing the iterative basic steps over the "key disjunction", and how much time the solver takes to solve the resulting MILP/MINLP. The next two columns show how many iterations the algorithm performed, and what was the criteria to stop iterating. From this column it can be seen that in most of the instances the criteria to stop iterating is the resulting problem size, while only in a few is the lack of improvement in the relaxation after three iterations. The last two columns show the number of proper and improper basic steps. It is interesting to note that, in general, few proper basic steps are applied, but many improper ones are. The improvement in the relaxation with few proper basic steps, but many improper ones, is consistent with the suggestions from previous work (Balas 1985, Ruiz and Grossmann 2012).

6. Conclusion

In this paper, we have proposed an automated algorithm that improves the relaxation of GDP formulations compared to the common (BM) and (HR) reformulations. We have

		Total solution time (s)			Optimality gap (%)			Number of B&B nodes		
Instance	Type	(BM)	(HR)	Algor	(BM)	(HR)	Algor	(BM)	(HR)	Algor
Batch101006	MINLP	204	**12**	287	0.1	0.1	0.1	9,424	**300**	8,553
Batch121208	MINLP	286	**77**	391	0.1	0.1	0.1	10,551	**1,554**	6,886
Batch151208	MINLP	319	**314**	2,007	0.0	0.1	0.1	11,619	**5,400**	54,549
Batch181210	MINLP	1,523	**205**	330	0.1	0.0	0.1	41,053	3,020	**983**
Batch201210	MINLP	5,980	**44**	643	0.0	0.1	0.1	153,039	**692**	10,769
C-Lay-3-2	MINLP	**3**	6	12	0.0	0.0	0.0	282	379	**187**
C-Lay-3-3	MINLP	**4**	10	17	0.0	0.0	0.0	462	449	**389**
C-Lay-4-2	MINLP	**24**	35	61	0.0	0.0	0.0	3,039	**2,038**	2,329
C-Lay-4-3	MINLP	71	**31**	78	0.0	0.0	0.0	8,857	1,571	**1,044**
C-Lay-5-2	MINLP	**383**	574	417	0.1	0.1	0.1	42,337	19,614	**14,286**
C-Lay-5-3	MINLP	2,696	2,498	**2,031**	0.1	0.1	0.1	316,060	**74,602**	217,847
C-Lay-5-4	MINLP	**2,084**	5,124	5,549	0.0	0.1	0.1	222,736	**113,571**	158,556
C-Lay-6-2	MINLP	5,534	7,200	**3,073**	0.1	21.8	0.1	578,938	312,185	**233,685**
C-Lay-6-3	MINLP	6,854	7,200	**3,107**	0.1	30.9	0.1	712,777	277,900	**224,824**
C-Lay-6-4	MINLP	7,200	7,200	7,200	34.5	26.2	**10.8**	573,910	236,327	187,354
C-Lay-6-5	MINLP	7,200	7,200	7,200	**97.9**	300.7	107.0	546,010	54,456	182,880
Dice0605	MILP	2,370	7,200	**948**	0.0	350.0	0.0	739,393	596,342	**16,043**
Dice0606	MILP	2,405	7,200	**1,203**	0.0	414.3	0.0	363,657	446,270	**13,104**
Dice0804	MILP	**1,436**	7,200	1,480	0.0	433.3	0.0	164,227	518,659	**14,802**
Dice0805	MILP	7,200	7,200	**3,489**	50.0	540.0	0.0	1,319,930	290,636	**29,728**
Dice1203	MILP	7,200	7,200	**4,371**	12.0	380.0	0.0	570,555	391,152	**81,598**
DiceH0605	MILP	3,108	7,200	**284**	0.0	300.0	0.0	1,904,882	2,220,152	**39,232**
DiceH0606	MILP	5,619	7,200	**2,398**	0.0	414.3	0.0	2,124,442	1,621,400	**831,391**
DiceH0804	MILP	7,200	7,200	**3,879**	433.3	276.5	0.0	5,419,210	1,806,748	**843,383**
DiceH0805	MILP	7,200	7,200	**3,422**	326.7	481.8	0.0	2,884,668	970,273	**487,549**
DiceH1203	MILP	7,200	7,200	7,200	200.0	311.4	**22.2**	1,433,590	953,789	551,692
F-lay-3	MINLP	**1**	2	7	0.0	0.0	0.0	**102**	110	**102**
F-lay-4	MINLP	**27**	48	101	0.0	0.1	0.1	2,834	2,768	**1,956**
F-lay-5	MINLP	**1,247**	5,660	1,629	0.1	0.1	0.1	**138,443**	171,486	140,446
F-lay-6	MINLP	7,200	7,200	7,200	23.4	43.4	**17.5**	546,524	97,396	432,925
Process-12	MINLP	2	**1**	14	0.0	0.0	0.0	168	**48**	232
Process-8	MINLP	1	**0**	9	0.0	0.0	0.0	24	**16**	18
S-Pck12	MILP	**101**	3,257	260	0.0	0.0	0.0	**1,026,258**	2,630,563	1,239,283
S-Pck13	MILP	7,200	7,200	7,200	**13.5**	1.7	**13.5**	28,179,292	4,600,552	14,313,072
S-Pck14	MILP	**2,205**	7,200	3,142	0.0	4.3	0.0	17,864,367	2,918,216	10,168,644
S-Pck15	MILP	7,200	7,200	7,200	11.1	21.2	**8.1**	21,255,244	3,329,851	56,619,228

Figure 10 Solution time, optimality gap and B&B nodes for the (BM), (HR) and algorithm formulations

developed a pre-solve procedure that reduces problem formulations, generates stronger bounds, and provides a value that helps in the selection of disjunctions to which basic steps should be applied. We have presented an iterative procedure in which basic steps are applied over the same disjunction in order to improve the continuous relaxation of the problem. In this step of the algorithm, we have also proposed some heuristics on the selection of these disjunctions. We show that a hybrid reformulation can provide some of the advantages of both (BM) and (HR), and that the selection of disjunctions reformulated through (BM) or (HR) is simple and clear for the proposed algorithm. Finally, we have applied the proposed method to improve the formulation of different numerical examples of Generalized Disjunctive Programs. These results show that the algorithm provides better formulations, in the sense that they yield strong lower bounds without an excessive increase in problem size. Furthermore, solution times for large instances are reduced by using the algorithm. Finally, even though the algorithm shows promising results, further research in the selection of key and secondary disjunctions, as well as improvements in the presolve

	Time (s)			Algorithm		Basic Steps	
Instance	Pre	Basic S	MINLP	Iter	Criteria	Proper	Improper
Batch101006	101	2.4	184	1	Size	0	11
Batch121208	209	1.7	180	1	Size	0	13
Batch151208	235	1.5	1,771	1	Size	0	16
Batch181210	295	2.3	33.4	1	Size	0	19
Batch201210	285.2	2.1	356	1	Size	0	21
C-Lay-3-2	7.4	0.6	3.5	1	Size	1	12
C-Lay-3-3	8.6	0.5	8.1	1	Size	1	12
C-Lay-4-2	14.6	0.6	46.2	1	Size	1	24
C-Lay-4-3	18.5	4.2	55.1	3	Size	3	24
C-Lay-5-2	20.9	1.8	394	1	Size	1	36
C-Lay-5-3	30.5	0.8	2,000	1	Size	1	36
C-Lay-5-4	26.5	1.9	5,521	1	Size	1	36
C-Lay-6-2	28.3	1.8	3,043	3	Size	2	36
C-Lay-6-3	32.7	1.9	3,072	2	Size	1	36
C-Lay-6-4	40.4	1.2	7,200	1	Size	1	48
C-Lay-6-5	1,045	1.4	7,200	1	Size	1	36
Dice0605	745	13.5	189	3	Size	0	4
Dice0606	1,055	9.6	139	3	Size	0	4
Dice0804	1,072	11.7	397	3	Size	0	4
Dice0805	2,183	23.5	1,282	3	Size	0	4
Dice1203	2,207	17.3	2,147	3	Size	0	4
DiceH0605	160	14.3	110	3	Size	0	5
DiceH0606	221	9.8	2,167	3	Size	0	6
DiceH0804	285	10.9	3,583	3	Size	0	6
DiceH0805	401	22.2	2,999	3	Size	0	6
DiceH1203	613	16.1	6,587	3	Size	0	8
F-lay-3	5.2	0.5	0.9	1	Size	0	6
F-lay-4	9.6	4.1	87.6	1	Size	1	9
F-lay-5	20.7	4.7	1,603	1	Size	0	6
F-lay-6	30.0	3.7	7,200	1	Size	0	6
Process-12	11.4	1.9	0.6	2	Size	2	9
Process-8	6.5	1.6	0.5	3	Size	1	6
S-Pck12	108	9.7	142	3	Iterations	0	2
S-Pck13	148	73.7	7,052	7	Size	5	4
S-Pck14	154	51.7	2,935	7	Size	5	4
S-Pck15	232	12.4	6,968	3	Iterations	0	2

Figure 11 Behaviour of the algorithm for each instance

and infeasible term detection, is needed to achieve faster performance. Furthermore, in problems with a large number of disjunctive terms the pre-solve may not be a good option. For these cases different rules for the selection of disjunctions would have to be derived.

Acknowledgments

The authors would like to acknowledge financial support from the National Science Foundation under Grant OCI-0750826. They also thank Michael Bussieck and Alex Meeraus from GAMS, for their valuable help in the implementation of the algorithm.

References

Abhishek, K., S. Leyffer, J. Linderoth. 2010. Filmint: An outer approximation-based solver for convex mixed-integer nonlinear programs. *INFORMS Journal on Computing* **22** 555–567.

Achterberg, T. 2004. *SCIP-a framework to integrate constraint and mixed integer programming*. Konrad-Zuse-Zentrum für Informationstechnik.

Balas, E. 1979. Disjunctive programming. *Annals of Discrete Mathematics* **5** 3–51.

Balas, E. 1985. Disjunctive programming and a hierarchy of relaxations for discrete continuous optimization problems. *SIAM. Journal on Algebraic and Discrete Methods* **6** 466–486.

Biegler, L. T., I. E. Grossmann, A. W. Westerberg. 1997. *Systematic methods of chemical process design*. Prentice-Hall international series in the physical and chemical engineering sciences, Prentice Hall PTR. URL `http://books.google.com/books?id=59NTAAAAMAAJ`.

Bixby, R. E., M. Fenelon, Z. Gu, E. Rothberg, Wunderling R. 2000. Mip: Theory and practice closing the gap. *IFIP The International Federation for Information Processing* **46** 19–49.

Bixby, R. E., E. Rothberg. 2007. Progress in computational mixed integer programming a look back from the other side of the tipping point. *Annals of Operations Research* **149** 37–41.

Brooke, A., D. Kendrick, A. Meeraus, Raman. 1998. Gams, a users guide. *The Scientific Press* .

Ceria, S., J. Soares. 1999. Convex programming for disjunctive convex optimization. *Mathematical Programming* **86** 595–614.

Clocksin, W. F., C. S. Mellish. 1981. *Programming in Prolog*. Springer.

CMU, IBM. 2013. Cmu-ibm open source minlp project. URL `http://egon.cheme.cmu.edu/ibm/page.htm`.

Dakin, R. J. 1965. A tree-search algorithm for mixed programming problems. *The Computer Journal* **8** 250–255.

Duran, M., I. E. Grossmann. 1986. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming* **36** 307–339.

Floudas, C. A. 1995. *Nonlinear and mixed integer optimization: Fundamentals and applications*. Oxford University Press.

Geoffrion, A. M. 1972. Generalized benders decomposition. *Journal of Optimization Theory and Applications* **10** 237–260.

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

Grossmann, I. E., S. Lee. 2003. Generalized convex disjunctive programming: nonlinear convex hull relaxation. *Computational Optimization and Applications* **26** 83–100.

Grossmann, I. E., J. P. Ruiz. 2012. Generalized disjunctive programming: A framework for formulation and alternative algorithms for minlp optimization. *The IMA Volumes in Mathematics and its Applications* **154** 93–115. doi:10.1007/978-1-4614-1927-3_4. URL `http://dx.doi.org/10.1007/978-1-4614-1927-3_4`.

Grossmann, I. E., F. Trespalacios. 2013. Systematic modeling of discrete-continuous optimization models through generalized disjunctive programming. *AIChE Journal* **59** 3276–3295. doi:10.1002/aic.14088. URL `http://dx.doi.org/10.1002/aic.14088`.

Gupta, O. K., A. Ravindran. 1985. Branch and bound experiments in convex nonlinear integer programming. *Management Science* **31** 1533–1546.

Hooker, J. N. 2002. Logic, optimization, and constraint programming. *INFORMS Journal on Computing* **14** 295–321.

Lee, S., I. E. Grossmann. 2000. New algorithms for nonlinear generalized disjunctive programming. *Computers and Chemical Engineering* **24** 2125–2141.

Lee, S., I. E. Grossmann. 2005. Logic-based modeling and solution of nonlinear discrete/continuous optimization problems. *Annals of Operations Research* **139** 267–288.

Lodi, A. 2010. Mixed integer programming computation. *50 Years of Integer Programming 1958-2008*. 619–645.

Mahajan, A. 2010. Presolving mixed-integer linear programs. *Wiley Encyclopedia of Operations Research and Management Science* .

Nemhauser, G. L., L. A. Wolsey. 1988. *Integer and Combinatorial Optimization, Wiley-Interscience*. Wiley.

Quesada, I., I. E. Grossmann. 1992. An lp/nlp based branch and bound algorithm for convex minlp optimization problems. *Computers and Chemical Engineering* **16** 937–947.

Raman, R., I. E. Grossmann. 1994. Modeling and computational techniques for logic-based integer programming. *Computers and Chemical Engineering* **18** 563–578.

Ruiz, J. P., I. E. Grossmann. 2012. A hierarchy of relaxations for nonlinear convex generalized disjunctive programming. *European Journal of Operational Research* **218** 38–47.

Savelsbergh, M. W. P. 1994. Preprocessing and probing techniques for mixed integer programming problems. *ORSA Journal on Computing* **6** 445–454.

Sawaya, N. 2006. Reformulations, relaxations and cutting planes for generalized disjunctive programming. Ph.D. thesis, Carnegie Mellon University.

Sawaya, N., I. E. Grossmann. 2012. A hierarchy of relaxations for linear generalized disjunctive programming. *European Journal of Operational Research* **216** 70–82.

Stubbs, R., S. Mehrotra. 1999. A branch-and-cut method for 0-1 mixed convex programming. *Mathematical Programming* **86** 515–532.

Turkay, M., I. E. Grossmann. 1996. A logic-based outer-approximation algorithm for minlp optimization of process flowsheets. *Computers and Chemical Engineering* **20** 959–978.

Vecchietti, A., S. Lee, I. E. Grossmann. 2003. Modeling of discrete/continuous optimization problems: characterization and formulation of disjunctions and their relaxations. *Computers and Chemical Engineering* **27** 433–448.

Westerlund, T., F. Pettersson. 1995. An extended cutting plane method for solving convex minlp problems. *Computers and Chemical Engineering* **19** 131–136.

Williams, H. P. 1985. *Model Building in Mathematical Programming*. Wiley.