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

Francisco Trespalacios and Ignacio E. Grossmann*
Department of Chemical Engineering
Carnegie Mellon University, Pittsburgh, PA 15213

July, 2013

Abstract

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.

1 Introduction

Mixed-integer linear and mixed-integer nonlinear programming models (MILP/MINLP) arise in different areas, such as process design[1][2][3], layout problems[4], financial modeling and electrical power management[5]. 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 proposed by Raman and Grossmann[6] 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[7] and Logic Based Outer Approximation[2], GDPs are normally reformulated as MILP/MINLP[8][9] to exploit the developments in these solvers.

Some of the methods to solve convex MINLP problems include branch and bound[10][11], branch and cut[5], generalized Benders' decomposition[12], outer approximation[1][13], LP/NLP based branch and bound[14] and extended cutting planes[15]. A comprehensive review of MINLP techniques is given by Grossmann[16], and of MILP methods and progress is given by Bixby et al[17][18].

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[19][20].

In this work, we make use of the logic structure of a GDP. Instead of directly reformulating it as an MILP/MINLP, we apply

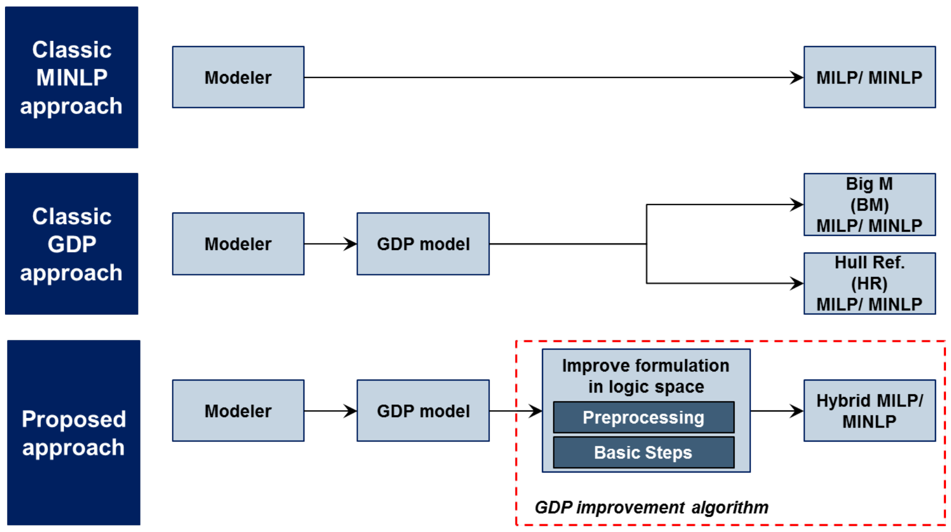


Figure 1: Different modeling approaches

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 direct transformation. 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 the proposed pre-analysis has some similarities to the MILP ”fixing variables” pre-solve technique[21][22][23][24], 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. 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 addresses the different examples used to test the algorithm, whose statistics, results and performance are reported in section 6.

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[25][6] 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[26].

2.1.1 GDP formulation

The general GDP formulation can be represented as follows:

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

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 $\underset{i\in D_k}{\veebar} Y_{ki}$. 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[16], GDP problems are normally reformulated as MILP/MINLP by using either the Big-M[9] (BM) or Hull Reformulation[8] (HR). (BM) generates a smaller MILP/MINLP, while (HR) generates a tighter one at the expense of larger number of variables and constraints[19][20]. The (BM) reformulation is given as follows:

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

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[27][28][29]. 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 continuous 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[4]:

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

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[6]. An example of GDP reformulation can be found in Appendix 1.

It is clear that formulation (HR) involves more variables and constraints than (BM), but it provides a tighter relaxation[19][20]. 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[30][5].

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

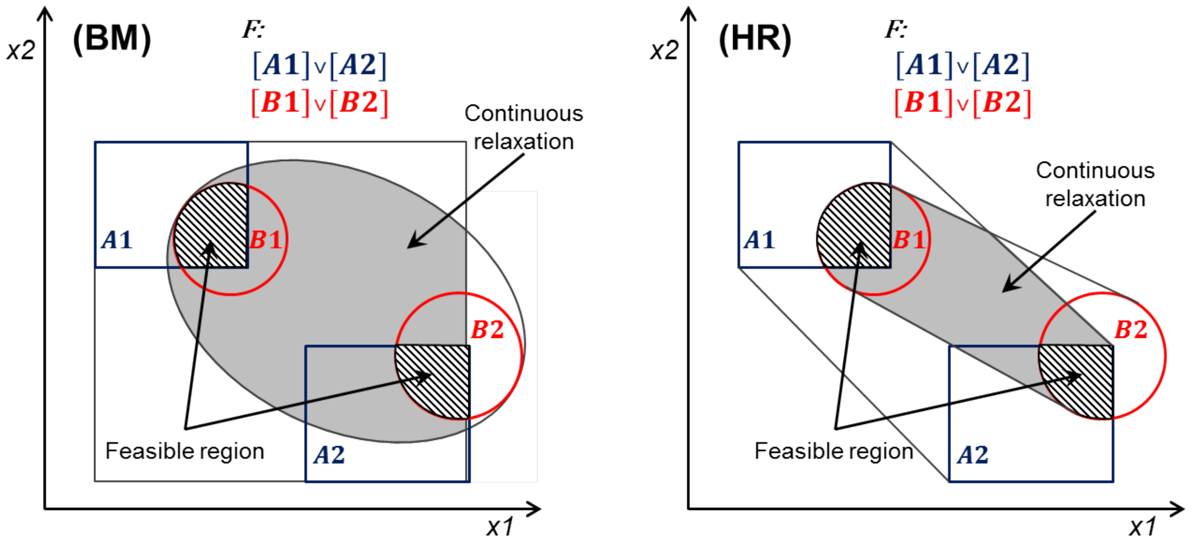


Figure 2: Illustration of (BM) and (HR) reformulations

the tightness of the (HR) we will make use of the logic operation called basic step.

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[31][32]:

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): $P = \underset{m \in M}{\cap} C_m$

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

Disjunction: $S_k = \underset{i \in D_k}{\cup} P_i = \underset{i \in D_k}{\cup} \underset{m \in M_i}{\cap} C_m$

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): $F = \underset{k \in K}{\cap} S_k$

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

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

CNF and DNF are the two extremes of a disjunctive set in regular form. Note that any GDP is in regular form[32], 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 in Balas[31] for the linear case, and extended by Ruiz and Grossmann[32] for the general nonlinear convex case, 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:

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

Ruiz and Grossmann[32] prove that any convex GDP is equivalent to a disjunctive convex program, 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[31][32]: 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 solution to the relaxed (HR) of the DNF is the same as the solution to the original GDP. Note that if the objective function is nonlinear, but it is convex in a minimization problem (or concave in a maximization problem), then it can be easily transformed into a constraint by adding a single variable in the objective (i.e. $min\ f(x)$ is transformed as $min\ \alpha$ and $f(x) \leq \alpha$ becomes a constraint). This final remark indicates that in the extreme case of DNF, it is possible to solve a continuous nonlinear program instead of a mixed-integer one[32].

Proper basic steps generate exponential growth in disjunctive terms. This would lead to an exponential growth in the number of binary variable if the (BM) or (HR) reformulations were directly performed. However, Balas[31] shows that we can obtain an equivalent formulation by including additional constraints, and representing the new terms with continuous variables between 0 and 1. These constraints relate the original binary variables to the new continuous ones, as shown in the following Theorem.

Theorem 2.2 *Consider MILP/MINLP representation of two disjunctions $k, l \in K$, whose disjunctive terms are represented by the 0-1 variables $y_{ki}, y_{lj}, i \in D_k, j \in D_l$. If a basic step is applied between disjunction k and disjunction l, the variables representing the disjunctive terms of the resulting disjunction $\hat{y}_{ij} \in \{0,1\}$ can be equivalently represented by:*

$$\begin{gathered}
y_{ki} = \sum_{j \in D_l} \hat{y}_{ij} \\
y_{lj} = \sum_{i \in D_k} \hat{y}_{ij} \\
\sum_{i \in D_k, j \in D_l} \hat{y}_{ij} = 1 \\
\sum_{i \in D_k} y_{ki} = 1 \\
\sum_{j \in D_l} y_{lj} = 1 \\
0 \leq \hat{y}_{ij} \leq 1 \\
y_{ki}, y_{lj} \in \{0,1\}
\end{gathered}$$

Proof. The proof follows from Theorem 4.4 of Balas work[31] ■.

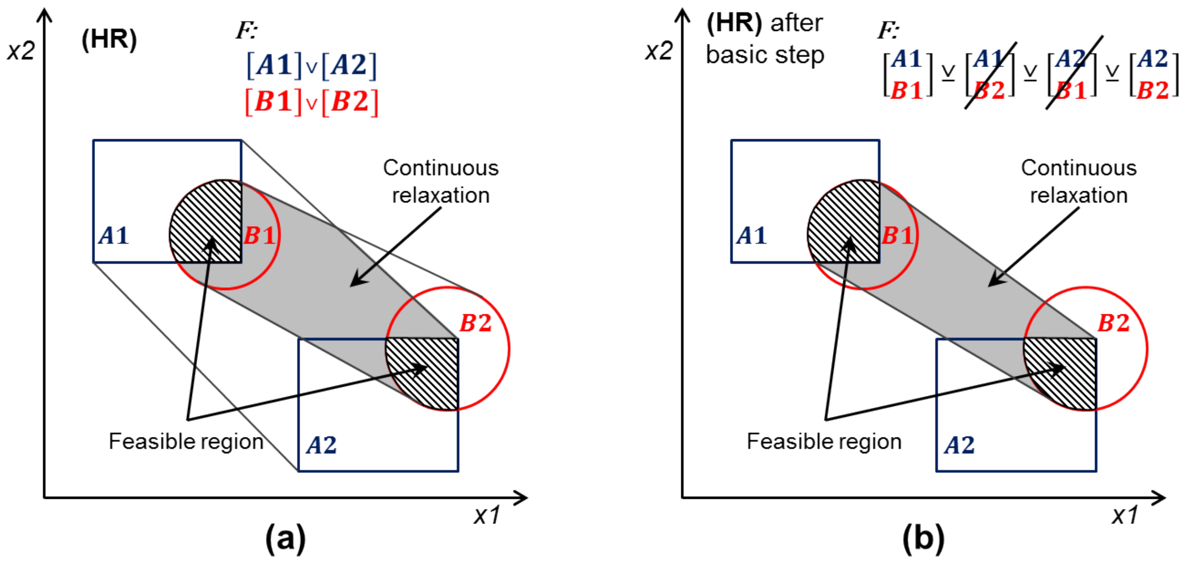


Figure 3: Illustration of (HR) (a) before, and (b) after the application of a basic step

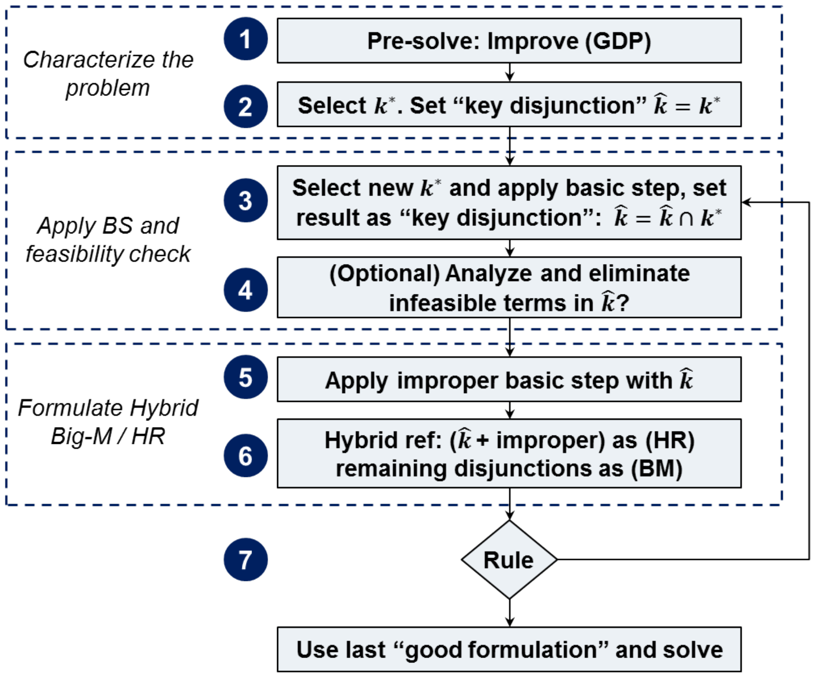
Theorem 2.2 establishes that by including additional constraints, the number of binary terms remains unchanged after the application of basic steps.

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 two disjunctive terms each, that is $([A_1] \vee [A_2]) \wedge ([B_1] \vee [B_2])$. The illustration to the right 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, we illustrate here not only that the basic step improves the tightness of the relaxation, but that it brings the problem to DNF. The (HR) of the DNF, as expressed earlier, describes the convex hull of the feasible region, as this can also be seen in the right figure. 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 is convenient or not to apply a basic step[31][33][32]. However, previous work provides mainly guidelines and case by case selection of disjunctions for basic steps, while the exponential growth of the formulation is a major issue. In this paper we propose an algorithmic approach on 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.



Characterize the problem
1
Pre-solve: Improve (GDP)
2
Select k^*. Set "key disjunction" $\hat{k} = k^*$
Apply BS and feasibility check
3
Select new k^* and apply basic step, set result as "key disjunction": $\hat{k} = \hat{k} \cap k^*$
4
(Optional) Analyze and eliminate infeasible terms in $\hat{k}$?
Formulate Hybrid Big-M / HR
5
Apply improper basic step with $\hat{k}$
6
Hybrid ref: ($\hat{k}$ + improper) as (HR) remaining disjunctions as (BM)
7
Rule
Use last "good formulation" and solve

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 will characterize each disjunction.

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

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 $\hat{k}_{iter} = k^*$; $\hat{K} = \{\hat{k}_1, ..., \hat{k}_{iter}\}$; $\hat{D}_{\hat{k}_{iter}} = D_{k^*}$; $\hat{i} = \{\hat{i}_1, ..., \hat{i}_{iter}\}$ where $\hat{i}_s \in \hat{D}_{\hat{k}_s}$ for $s = 1, ..., iter$.

Goal. Select the next disjunction and apply 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 $\hat{i}$, 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).

$$
\begin{array}{lll}
 & min\ z = f(x) & \\
s.t. & g_e(x) \le 0 & e \in E \backslash \hat{E} \\
 & r_{ki}(x) \le M^{ki}(1 - y_{ki}) & k \in K \backslash \hat{K},\ i \in D_k \\
 & x = \sum\limits_{\hat{i}} \nu^{\hat{i}} & \\
 & \hat{y}_{\hat{i}} g_e(\nu^{\hat{i}}/\hat{y}_{\hat{i}}) \le 0 & e \in \hat{E},\ \forall \hat{i} \\
 & \hat{y}_{\tilde{i}_s} r_{\hat{k}_s i}(\nu^{\tilde{i}_s}/\hat{y}_{\tilde{i}_s}) \le 0 & s = 1, ..., iter,\ i \in \hat{D}_{k_s},\ \forall \tilde{i}_s,\ where\ \tilde{i}_s = \hat{i} \cap (\hat{i}_s = i) \\
 & y_{\hat{k}_s i} = \sum\limits_{\substack{\hat{i} \\ \hat{i}_s = i}} \hat{y}_{\hat{i}} & s = 1, ..., iter,\ i \in \hat{D_{k_s}} \\
 & x^{lo} \hat{y}_{\hat{i}} \le \nu^{\hat{i}} \le x^{up} \hat{y}_{\hat{i}} & \forall \hat{i} \\
 & \sum\limits_{i \in D_k} y_{ki} = 1 & k \in K \\
 & \sum\limits_{\hat{i}} \hat{y}_{\hat{i}} = 1 & \\
 & Hy \ge h & \\
 & \hat{y}_{\hat{i}} = 0 & \hat{i} \in INEAS_{iter} \\
 & z^{lo} \le z & \\
 & x^{lo} \le x \le x^{up} & \\
 & x \in \mathbb{R}^n & \\
 & y_{ki} \in \{0, 1\} & k \in K, i \in D_k \\
 & 0 \le \hat{y}_{\hat{i}} \le 1 & \hat{i} \notin INEAS_{iter}
\end{array}
\tag{GDPH}
$$

Step 7. If relaxed $(GDPH) > z^*$, set $z^* = relaxation(GDPH)$, $GDP^* = GDPH$. If the relaxation has not improved after a specific maximum number of iterations, or the GDPH problem size is greater than 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) ($r_{ki}(x) \le M^{ki}(1 - y_{ki})$). The disjunctions that were intersected with basic steps now form a single disjunction, which we will denote "key disjunction", and that contains all terms $\hat{i}$. The corresponding variable to this new terms is $\hat{y}_{\hat{i}}$. Note that $|\hat{i}| = |\hat{D}_{k_1}| * ... * |\hat{D}_{k_{iter}}|$, which indicates an exponential growth of the disjunction with the number of iterations. The "key disjunction" is reformulated using (HR). The equation $x = \sum\limits_{\hat{i}} \nu^{\hat{i}}$ 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" ($\hat{i}$). 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 of the "key disjunction" $\hat{i}$, as long as its corresponding element $\hat{i}_s$ is equal to i ($\tilde{i}_s = \hat{i} \cap (\hat{i}_s = i)$). This allows the constraints in the original disjunctions $r_{\hat{k}_s i}$ to be assigned to the disjunctive and the disaggregated variables of the new “key disjunction” ($\hat{y}_{\tilde{i}_s}$, $\nu^{\tilde{i}_s}$). Equation $y_{\hat{k}_s i} = \sum_{\substack{\hat{i} \\ \hat{i}_s = i}} \hat{y}_{\hat{i}}$ 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[32].

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 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)) ■.

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. $Y_{ki} = False$) ■.

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 $infeasible$ ■.

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 ■.

It is important to note that this process requires the evaluation of $\sum_{k \in K} |D_k|$ LP/NLP problems. Though 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 **z = 11**, in which $Y_{12} = True$ and $Y_{21} = True$. The (BM) provides a relaxation of $\mathbf{z^{BM}} = \mathbf{3}$, and the (HR) a relaxation of $\mathbf{z^{HR}} = \mathbf{9.16}$. 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).

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

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).

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

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 **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[31] 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[33] and Ruiz and Grossmann[32] 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[31]: 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 ($|\hat{i}| = |\hat{D}_{k_1}| * ... * |\hat{D}_{k_{iter}}|$). 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 $n = 1$ 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, $\hat{i}$ 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 $\hat{i}$ 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[34]. 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 (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, 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.

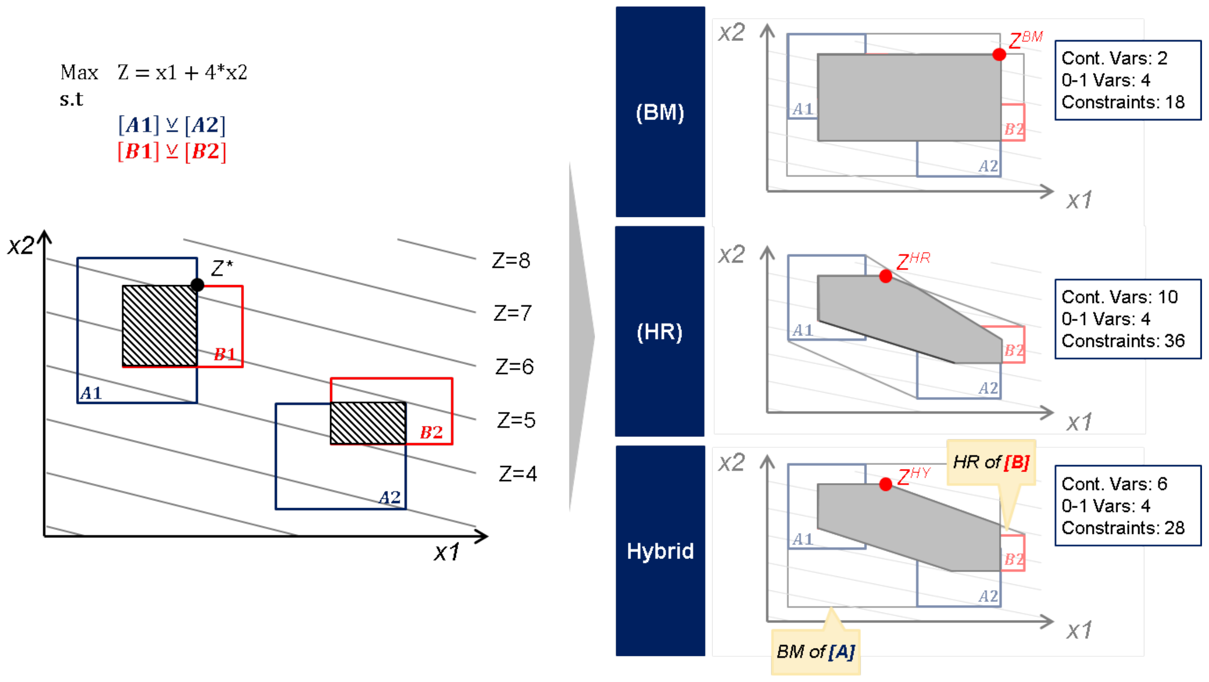


Figure 5: Illustration of (BM), (HR) and Hybrid reformulation

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 (10)):

$$
\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} \qquad\qquad 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 \qquad\qquad j = 1, 2, 3, 4 \\
& Y_{ki} \in \{True, False\} \qquad\qquad k = 1, ..., 6, i = 1, 2, 3, 4
\end{aligned}
\tag{4}
$$

This problem has an optimal solution of **lt = 15**, in which $Y_{12}, Y_{21}, Y_{32}, Y_{41}, Y_{53}, Y_{63} = True$. The continuous relaxation of its (BM) reformulation provides 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 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 $W = (0.75, 0.75, 0.38, 0.75, 0.38, 0.38)$. Table1 shows the iterations to obtain W, following the procedure described in section 3.3. Disjunctions 1, 2 and 4 have the same weighting value, 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 $\hat{k}_2 = 2$; $\hat{K} = \{1, 2\}$; $\hat{D}_2 = \{1, 2\}$; $\hat{i} = \{(1, 1), (1, 2), (2, 1), (2, 2)\}$.

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

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

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

From Theorem 2.2, the additional constraints that are added so that the new disjunctive variables $\hat{y}_{\hat{i}}$ can be continuous are shown in (6):

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

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).

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

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 is shown in Appendix 2 in equation (20).

Step 7. The relaxed solution of (20) is 11. Since relaxed $(20) > 8.3$, set $z^* = 11$, GDP^*=(20). 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).

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

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.

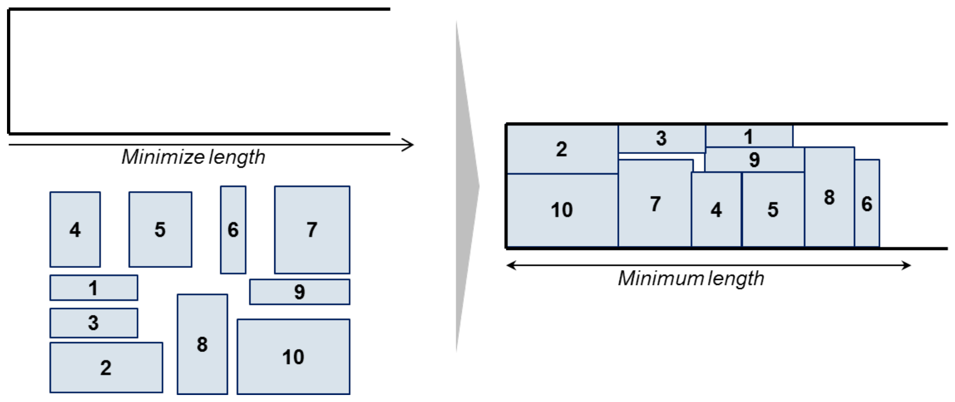


Figure 6: Illustration of strip packing problem

5 Examples

The algorithm was tested with 30 instances of five problems: strip packing, process flowsheet, farm layout, constrained layout and design of a multi-product batch plant. The first problem is a linear GDP, while all the others are convex nonlinear GDP. This section describes the problems and provides the GDP formulation.

5.1 Strip Packing (S-Pck)

In the strip packing problem a set of rectangles and a strip with fixed width are given. The objective is to pack these rectangles, without rotation and overlap, within the strip minimizing its length. Figure 6 illustrates this problem. The linear GDP formulation is as follows[33]:

$$
\begin{aligned}
& min\ lt & & \\
s.t.\quad & lt \geq x_i + L_i & & i \in N \\
& \begin{bmatrix} Y_{ij}^1 \\ x_i + L_i \leq x_j \end{bmatrix} \vee \begin{bmatrix} Y_{ij}^2 \\ x_j + L_j \leq x_i \end{bmatrix} \vee \begin{bmatrix} Y_{ij}^3 \\ y_i - H_i \geq y_j \end{bmatrix} \vee \begin{bmatrix} Y_{ij}^4 \\ y_j - H_j \geq y_i \end{bmatrix} & & i,j \in N, i<j \\
& Y_{ij}^1 \veebar Y_{ij}^2 \veebar Y_{ij}^3 \veebar Y_{ij}^4 & & i,j \in N, i<j \\
& 0 \leq x_i \leq UB - L_i & & i \in N \\
& H_i \leq y_i \leq W & & i \in N \\
& Y_{ij}^1, Y_{ij}^2, Y_{ij}^3, Y_{ij}^4 \in \{True, False\} & & i,j \in N, i<j
\end{aligned}
\tag{10}
$$

In (10) x_i and y_i represent the coordinates of the upper-left corner of each of the $i \in N$ rectangles. The global constraints indicate that the total length lt is larger than the x_i coordinate of all the rectangles plus their length L_i. There is a disjunction for each pair of rectangles (i.e. if there are three rectangles there will be three disjunctions representing the pairs: (1,2), (1,3) and (2,3)). Each term in this disjunctions represent the possible relation between two rectangles: rectangle i is either to the left, to the right, above or below rectangle j. These disjunctions ensure that there is no overlap.

5.2 Design of Nontransitive Dice (Dice)

For the design of nontransitive dice problem a number of dice are given ($n \in N$), each with the same number of faces ($f \in F$). The objective is to design a set of dice that maximizes the probability of dice n beating dice $n + 1$, and the last dice ($n = |N|$)

beating the first one ($n = 1$). For simplicity in the cyclic representation and description of the problem, we will consider the dice $|N| + 1 = 1$. Each dice n must have the same probability of beating dice $n + 1$[35]. More specifically, there are $|F| * |F|$ possible outcomes between each pair of dice n and $n + 1$. The problem seeks to maximize the number of outcomes in which dice n beats dice $n + 1$ (or equivalently, minimize the number of outcomes in which dice $n + 1$ beats dice n). Additionally, in this problem we enforce that each face in each dice has a different integer value[36][37]. Figure 7 shows an example of 3 nontransitive dice with 6 faces each. In the example the first dice ($n = 1$) beats the second one ($n = 2$) in 21 of the 36 possible outcomes ((9,5); (9,6); (9,7); (9,8); (10,5); (10,6); (10,7); (10,8); (11,5); (11,6); (11,7); (11,8); (12,5); (12,6); (12,7); (12,8); (14,5); (14,6); (14,7); (14,8); (14,13)). This means that dice $n = 1$ has has 58.3% (21/36) probability of beating the second dice. The second one has also 58.3% probability of beating the third dice. Finally, the third dice has also 58.3% probability of beating the first one. The linear GDP formulation of this problem is shown in (11), and it is formulated as minimization of the number of outcomes in which dice $n + 1$ beats dice n (instead of maximizing the number of outcomes in which dice n beats dice $n + 1$).

$$
\begin{aligned}
& min\ lost && \\
s.t.\quad & lost = \sum_{f \in F} \sum_{\hat{f} \in F} w^{lose}_{n,f,\hat{f}} && n \in N \\
& x_{n,f-1} + 1 \le x_{n,f} && n \in N, f \in F \\
& \begin{bmatrix} Y^{win}_{n,f,\hat{f}} \\ w^{lose}_{n,f,\hat{f}} = 0 \\ x_{n,f} \ge x_{n+1,\hat{f}} + 1 \end{bmatrix} \vee \begin{bmatrix} Y^{lose}_{n,f,\hat{f}} \\ w^{lose}_{n,f,\hat{f}} = 1 \\ x_{n,f} \le x_{n+1,\hat{f}} \end{bmatrix} && n \in N, f \in F, \hat{f} \in F \\
& \bigvee_{\hat{n} \in N, \hat{f} \in F} \begin{bmatrix} Z_{n,f,\hat{n},\hat{f}} \\ x_{n,f} = \hat{f} + |F|(\hat{n} - 1) \end{bmatrix} && n \in N, f \in F \\
& Y^{win}_{n,f,\hat{f}} \veebar Y^{lose}_{n,f,\hat{f}} && n \in N, f \in F, \hat{f} \in F \\
& \underset{\hat{n} \in N, \hat{f} \in F}{\veebar} Z_{n,f,\hat{n},\hat{f}} && n \in N, f \in F \\
& \underset{n \in N, f \in F}{\veebar} Z_{n,f,\hat{n},\hat{f}} && \hat{n} \in N, \hat{f} \in F \\
& 0 \le x_{n,f} \le |N| * |F| && n \in N, f \in F \\
& 0 \le w^{lose}_{|N|,f,\hat{f}} \le 1 && n \in N, f \in F, \hat{f} \in F \\
& Y^{win}_{n,f,\hat{f}}, Y^{lose}_{n,f,\hat{f}}, Z_{n,f,\hat{n},\hat{f}} \in \{True, False\} && n, \hat{n} \in N, f, \hat{f} \in F
\end{aligned}
\tag{11}
$$

In (11), variable $w^{lose}_{n,f,\hat{f}}$ represents face f of dice n loosing to face $\hat{f}$ of dice $n+1$. Variable $x_{n,f}$ represents the integer value assigned to face f of dice n (though we represent it with a continuous variable, the second disjunction enforces that $x_{n,f}$ is integer). Problem (11) seeks to minimize the number outcomes in which a dice n loses to dice $n+1$. The first global constraint ensures that all dice n have the same number of losses to dice $n + 1$. The second constraint breaks symmetry by assigning the numbers of the faces of a dice in increasing order to the face number. The first disjunction defines if a face f of a dice n wins or looses against a face $\hat{f}$ of the dice $n + 1$. The second disjunction enforces that each face in each dice is assigned an integer number. The first two logic constraints correspond to the disjunctions of the GDP formulation. The third one enforces that each number is assigned only once.

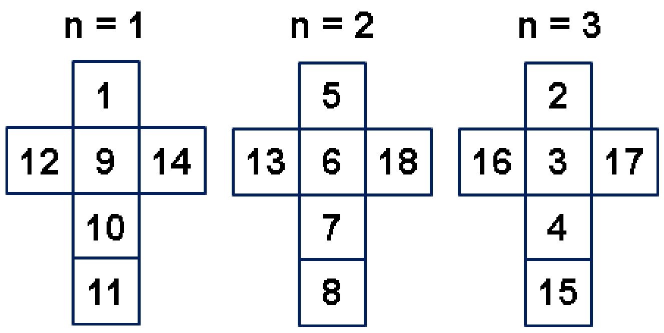


Figure 7: Illustration of 3 Nontransitive Dice with 6 faces

5.3 Design of Nontransitive Dice Hybrid GDP-MINLP (DiceH)

Problem "DiceH" is the same problem as "Dice", but using a hybrid MINLP-GDP formulation. In this hybrid reformulation, the disjunctions that assign one number to each face of each dice are formulated as (HR). After a couple of algebraic steps, the number of continuous variables and constraints can be greatly reduced. The hybrid GDP-MINLP formulation is shown in (12).

$$
\begin{aligned}
& min\ lost && \\
s.t.\quad & lost = \sum_{f \in F} \sum_{\hat{f} \in F} w^{lose}_{n,f,\hat{f}} && n \in N \\
& x_{n,f-1} + 1 \leq x_{n,f} && n \in N, f \in F \\
& x_{n,f} = \sum_{\hat{n} \in N} \sum_{\hat{f} \in F} (\hat{f} + |F|(\hat{n}-1)) * z_{n,f,\hat{n},\hat{f}} && n \in N, f \in F \\
& \sum_{n \in N} \sum_{f \in F} z_{n,f,\hat{n},\hat{f}} = 1 && \hat{n} \in N, \hat{f} \in F \\
& \begin{bmatrix} Y^{win}_{n,f,\hat{f}} \\ w^{lose}_{n,f,\hat{f}} = 0 \\ x_{n,f} \geq x_{n+1,\hat{f}} + 1 \end{bmatrix} \vee \begin{bmatrix} Y^{lose}_{n,f,\hat{f}} \\ w^{lose}_{n,f,\hat{f}} = 1 \\ x_{n,f} \leq x_{n+1,\hat{f}} \end{bmatrix} && n \in N, f \in F, \hat{f} \in F \\
& Y^{win}_{n,f,\hat{f}} \veebar Y^{lose}_{n,f,\hat{f}} && n \in N, f \in F, \hat{f} \in F \\
& 0 \leq x_{n,f} \leq |N| * |F| && n \in N, f \in F \\
& 0 \leq w^{lose}_{|N|,f,\hat{f}} \leq 1 && n \in N, f \in F, \hat{f} \in F \\
& Y^{win}_{n,f,\hat{f}}, Y^{lose}_{n,f,\hat{f}}, Z_{n,f,\hat{n},\hat{f}} \in \{True, False\} && n, \hat{n} \in N, f, \hat{f} \in F
\end{aligned}
\tag{12}
$$

Problem (12) has two main difference with (11). The first one is that the second disjunction of (11) is formulated as an MILP constraint, the third global constraint in (12). The second important difference is that the logic constraint that corresponds to the second disjunction in (11) is removed. Note that in (11) there are two logic constraints that involve the Boolean variables of the second disjunction $\left(\underset{\hat{n} \in N, \hat{f} \in F}{\veebar} Z_{n,f,\hat{n},\hat{f}} \text{ and } \underset{n \in N, f \in F}{\veebar} Z_{n,f,\hat{n},\hat{f}} \right)$, while in (12) only the second one appears $\left(\sum_{n \in N} \sum_{f \in F} z_{n,f,\hat{n},\hat{f}} = 1 \right)$. This change is valid, because constraint $\left(\sum_{n \in N} \sum_{f \in F} z_{n,f,\hat{n},\hat{f}} = 1 \right)$ enforces that each number appear exactly in one face. Since there are $|N| * |F|$ faces and numbers, this constraint also enforces that each face has exactly one of the different numbers.

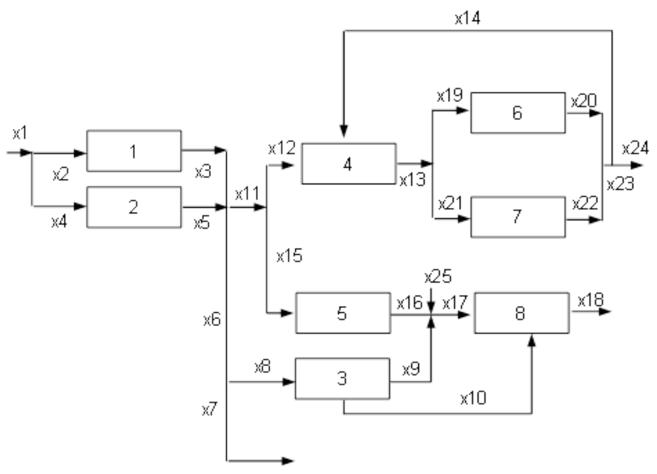


Figure 8: Superstructure illustration of an 8-equipment process network

5.4 Process Network (Process)

The process network problem "Process" is a classic optimization problem in process design. The model seeks to maximize the profit of selling a set of products taking into account the cost of raw materials and equipment. Figure 8 illustrates the superstructure for a process with potentially 8 units. The model that describes the performance of each unit is normally large and quite complex. In this example, however, the process is simplified to single input-output relations that give rise to a convex GDP[32]. The GDP problem formulation is as follows:

$$
\begin{aligned}
& min\ Z = \sum_{i \in I} c_i + \sum_{j \in J} p_j x_j + \alpha \\
s.t. \quad & \sum_{j \in J} r_{jn} x_j \leq 0 \qquad\qquad \forall n \in N \\
& \begin{bmatrix} Y_i \\ \sum_{j \in J^i} d_{ij}(e^{x_j/t_{ij}} - 1) - \sum_{j \in J^i} s_{ij} x_j \leq 0 \\ c_i = \gamma_i \end{bmatrix} \vee \begin{bmatrix} \neg Y_i \\ x_j = 0 \quad \forall j \in J^i \\ c_i = 0 \end{bmatrix} \qquad i \in I \qquad (13) \\
& \Omega(Y) = True \\
& c_i, x_j \geq 0 \\
& Y_i \in \{True, False\}
\end{aligned}
$$

In (13) c_i is the cost associated to each equipment $i \in I$. x_j represents each of the flows $j \in J$, and p_j the profit or cost associated to each one. The global constraints represent the mass balance in each of the $n \in N$ nodes, where r_{jn} is the coefficient of the mass balance for flow j. There is a disjunction for each unit i. If a unit is selected ($Y_i = True$) then the corresponding mass balance has to be satisfied, and the cost of the unit c_i takes the value associated to that equipment γ_i. If it is not selected ($Y_i = False$ or, equivalently, $\neg Y_i = True$), then all the flows $j \in J^i$ in and out that equipment become 0, and the cost c_i also becomes 0. Finally $\Omega(Y) = True$ represents the topology of the superstructure.

5.5 Farm Layout (F-Lay)

In the farm layout problem the objective is to determine the width and length of a number of rectangles with fixed area in order to minimize the total perimeter. Figure 9 illustrates this problem, which can be formulated as the following convex GDP[4]:

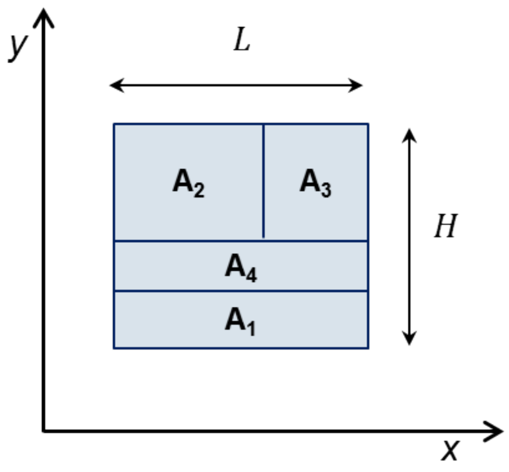


Figure 9: Illustration of farm layout problem

$$
\begin{aligned}
& && min\ Z = 2(Length + Width) \\
& s.t. && Length \geq x_i + L_i && i \in N \\
& && Width \geq y_i + W_i && i \in N \\
& && A_i/W_i - L_i \leq 0 && i \in N \\
& && \begin{bmatrix} Y_{ij}^1 \\ x_i + L_i \leq x_j \end{bmatrix} \vee \begin{bmatrix} Y_{ij}^2 \\ x_j + L_j \leq x_i \end{bmatrix} \vee \begin{bmatrix} Y_{ij}^3 \\ y_i + W_i \leq y_j \end{bmatrix} \vee \begin{bmatrix} Y_{ij}^4 \\ y_j + W_j \leq y_i \end{bmatrix} && i, j \in N, i < j \\
& && Y_{ij}^1 \veebar Y_{ij}^2 \veebar Y_{ij}^3 \veebar Y_{ij}^4 && i, j \in N, i < j \\
& && 0 \leq Length \leq Length^{up};\ \ 0 \leq Width \leq Width^{up} \\
& && L_i^{lo} \leq L_i \leq L_i^{up};\ \ W_i^{lo} \leq W_i \leq L_i^{up} && i \in N \\
& && 0 \leq x_i \leq Length^{up} - L_i^{lo};\ \ 0 \leq y_i \leq Width^{up} - L_i^{lo} && i \in N \\
& && Y_{ij}^1, Y_{ij}^2, Y_{ij}^3, Y_{ij}^4 \in \{True, False\} && i, j \in N, i < j
\end{aligned}
\tag{14}
$$

In formulation (14) the variables x_i and y_i represent the coordinates of lower-left corner of each rectangle $i \in N$, while L_i and W_i represent their corresponding length and width. $Length$ and $Width$ represent the length and width of the total area. A_i is the given area for each rectangle. Similarly to the strip packing problem (10), there is one disjunction for each pair of rectangles. Each term in the disjunction represents the possible relative position between the two rectangles: rectangle i is either to the left, or to the right, or below, or above rectangle j, respectively.

5.6 Constrained Layout (C-Lay)

The constrained layout problem is similar to the strip packing problem, but the rectangles in this case have to be packed inside a set of fixed circles. The objective function is to minimize the distance in x and y axis, with a cost associated to every pair of rectangles. Figure 10 illustrates the constrained layout problem. It can be formulated as the following convex GDP[4]:

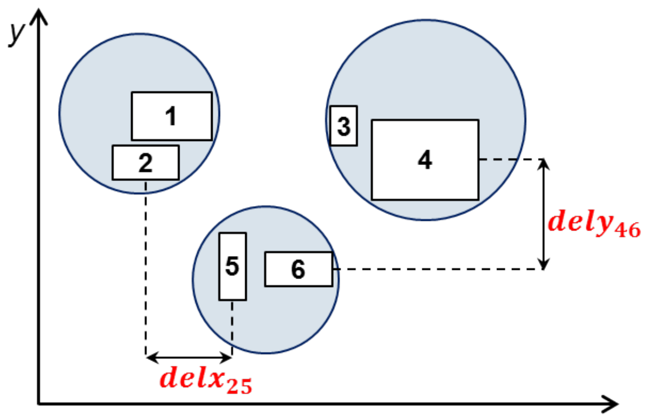


Figure 10: Illustration of constrained layout problem

$$
\begin{aligned}
& min\ Z = \sum_i \sum_j c_{ij}(delx_{ij} + dely_{ij}) & & \\
s.t. \quad & delx_{ij} \geq x_i - x_j & & i,j \in N, i < j \\
& delx_{ij} \geq x_j - x_i & & i,j \in N, i < j \\
& dely_{ij} \geq y_i - y_j & & i,j \in N, i < j \\
& dely_{ij} \geq y_j - y_i & & i,j \in N, i < j \\
& \begin{bmatrix} Y_{ij}^1 \\ x_i + L_i/2 \leq x_j - L_j/2 \end{bmatrix} \vee \begin{bmatrix} Y_{ij}^2 \\ x_j + L_j/2 \leq x_i - L_i/2 \end{bmatrix} & & \\
& \vee \begin{bmatrix} Y_{ij}^3 \\ y_i + H_i/2 \leq y_j - H_j/2 \end{bmatrix} \vee \begin{bmatrix} Y_{ij}^4 \\ y_j + H_j/2 \leq y_i - H_i/2 \end{bmatrix} & & i,j \in N, i < j \\
& \bigvee_{t \in T} \begin{bmatrix} W_{it} \\ (x_i + L_i/2 - xc_t)^2 + (y_i + H_i/2 - yc_t)^2 \leq r_t^2 \\ (x_i + L_i/2 - xc_t)^2 + (y_i - H_i/2 - yc_t)^2 \leq r_t^2 \\ (x_i - L_i/2 - xc_t)^2 + (y_i + H_i/2 - yc_t)^2 \leq r_t^2 \\ (x_i - L_i/2 - xc_t)^2 + (y_i - H_i/2 - yc_t)^2 \leq r_t^2 \end{bmatrix} & & i \in N \\
& Y_{ij}^1 \veebar Y_{ij}^2 \veebar Y_{ij}^3 \veebar Y_{ij}^4 & & i,j \in N, i < j \\
& \underset{t \in T}{\veebar}\ W_{it} & & i \in N \\
& 0 \leq x_i \leq x_i^{up} & & i \in N \\
& 0 \leq y_i \leq y_i^{up} & & i \in N \\
& Y_{ij}^1, Y_{ij}^2, Y_{ij}^3, Y_{ij}^4 \in \{True, False\} & & i,j \in N, i < j \\
& W_{it} \in \{True, False\} & & i \in N, t \in T
\end{aligned}
\tag{15}
$$

In formulation (15) x_i and y_i represent the coordinates of the centre of the rectangles $i \in N$. $delx_{ij}$ and $dely_{ij}$ represent the distance between two rectangles $i, j \in N, i < j$, and c_{ij} is the cost associated with these. The first disjunctions, similarly to strip packing and farm layout problems, ensures that there is no overlap by expressing the possible relative position between rectangles i and j. The second set of disjunctions ensure that every rectangle i is inside one of the $t \in T$ circles. For a circle t, its coordinates (xc_t,yc_t) and its radius r_t are given.

5.7 Design of multi-product batch plant (Batch)

This problem seeks to minimize the investment cost in the design of a plant with multiple units in parallel and intermediate storage tanks [38]. The design involves selecting the number of parallel units, volume of the equipment, and volume and location of the intermediate storage tanks. This problem can be convexified[39], and the formulation is as follows:

$$
\begin{aligned}
& min\ Z = \alpha_1 \sum_j \exp(n_j + m_j + \beta_1 v_j) + \alpha_2 \sum_{Tj} \exp(\beta_2 v_{Tj}) & \\
s.t. \quad & v_j \geq \ln(S_{ij}) + b_{ij} - n_j & \forall i, j \\
& e_i \geq \ln(T_{ij}) - b_{ij} - m_j & \forall i, j \\
& H \geq \sum_i (Q_i e_i) & \\
& \begin{bmatrix} YS_j & \\ v_{Tj} \geq \ln(S_j^*) + b_{ij+1} & \forall i \\ v_{Tj} \geq \ln(S_j^*) + b_{ij} & \forall i \\ b_{ij} - b_{ij+1} \leq \ln(S_{ij}^*) & \forall i \\ b_{ij} - b_{ij+1} \geq -\ln(S_{ij}^*) & \forall i \end{bmatrix} \vee \begin{bmatrix} \neg YS_j & \\ v_{Tj} = 0 & \\ b_{ij} - b_{ij+1} = 0 & \forall i \end{bmatrix} & \forall j < |J| \\
& \begin{bmatrix} YM_{j,1} \\ m_j = \ln(1) \end{bmatrix} \vee ... \vee \begin{bmatrix} YM_{j,maxp} \\ m_j = \ln(maxp) \end{bmatrix} & \forall j \\
& \begin{bmatrix} YN_{j,1} \\ n_j = \ln(1) \end{bmatrix} \vee ... \vee \begin{bmatrix} YN_{j,maxp} \\ n_j = \ln(maxp) \end{bmatrix} & \forall j \\
& YM_{j,1} \veebar ... \veebar YM_{j,maxp} & \forall j \\
& YN_{j,1} \veebar ... \veebar YN_{j,maxp} & \forall j \\
& YS_j, YM_{j,p}, YN_{j,p} \in \{True, False\} & \forall j, p = 1, ..., maxp
\end{aligned}
\qquad (16)
$$

The nomenclature of this problem is shown in Appendix 3.

6 Results

In this section we present the computational results of applying the algorithm described in section 3 to different instances of the examples presented in 5. 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 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 was used, and which can be found in the CMU-IBM MINLP library[40].

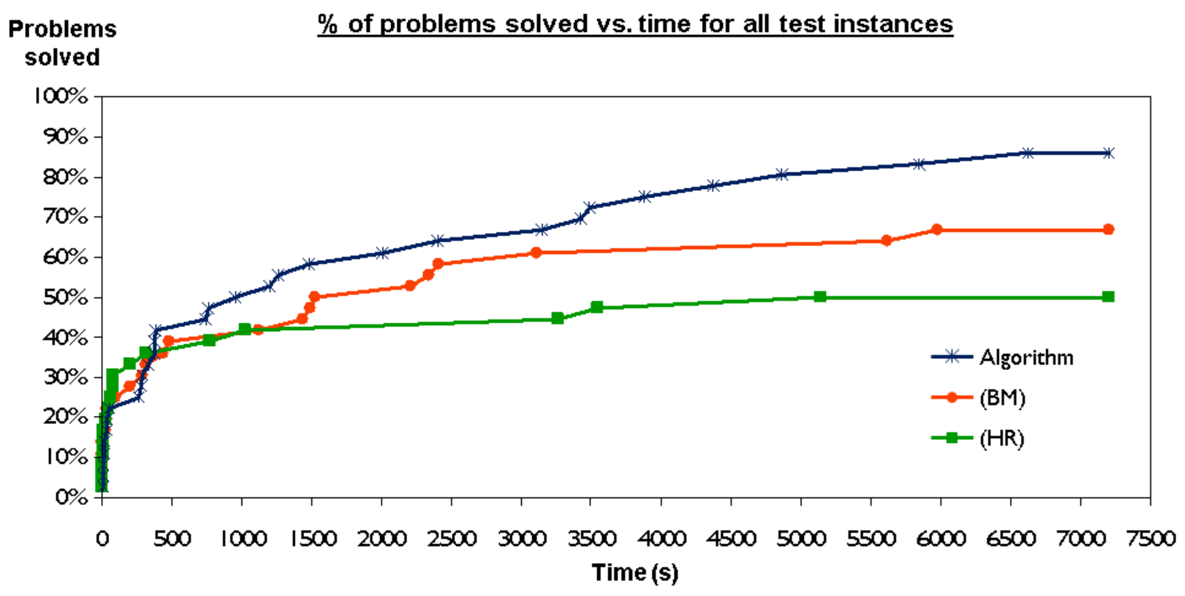


Figure 11: Percentage of problems solved vs. time for the test instances

The instances are solved using branch and bound methods, Gurobi 5.5 for the linear GDP problems and SBB for the convex GDP. Cuts and presolve were deactivated in Gurobi for all linear instances. The algorithm and models were implemented in GAMS[41] and solved in an Intel(R) Core(TM) i7 CPU 2.93 GHz and 4 GB of RAM.

We first present and discuss the general performance of the algorithm shown in Figures 11, 12, 13 and 14. We then discuss in more detail some characteristics of the different instances presented in Tables 2 and 3.

Figure 11 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 12 shows the number of nodes that where evaluated to find and prove optimality (within 0.1 % gap) for the three formulations. As expected the (HR) requires fewer nodes than the (BM) to achieve optimality. It is interesting to note that the algorithm needs fewer number of nodes than the (HR) in the larger instances. One of the possible reasons for this is that the algorithm provides a better continuous relaxation in all instances as shown in Figure 13. Figure 13 shows the continuous relaxation gap of the three formulations as percentage of the optimal solution vs. the % of problems that lie in that gap. For the formulations where the optimal solution was not found, the best solution was used to calculate this percentage. This figure shows, as expected, that the (HR) in general has a relaxation that is stronger than the (BM). It also shows that the relaxation of the formulation obtained with the algorithm is stronger than (BM) and (HR).

Figure 14 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. Therefore, having not much larger or even smaller problem sizes represents an important

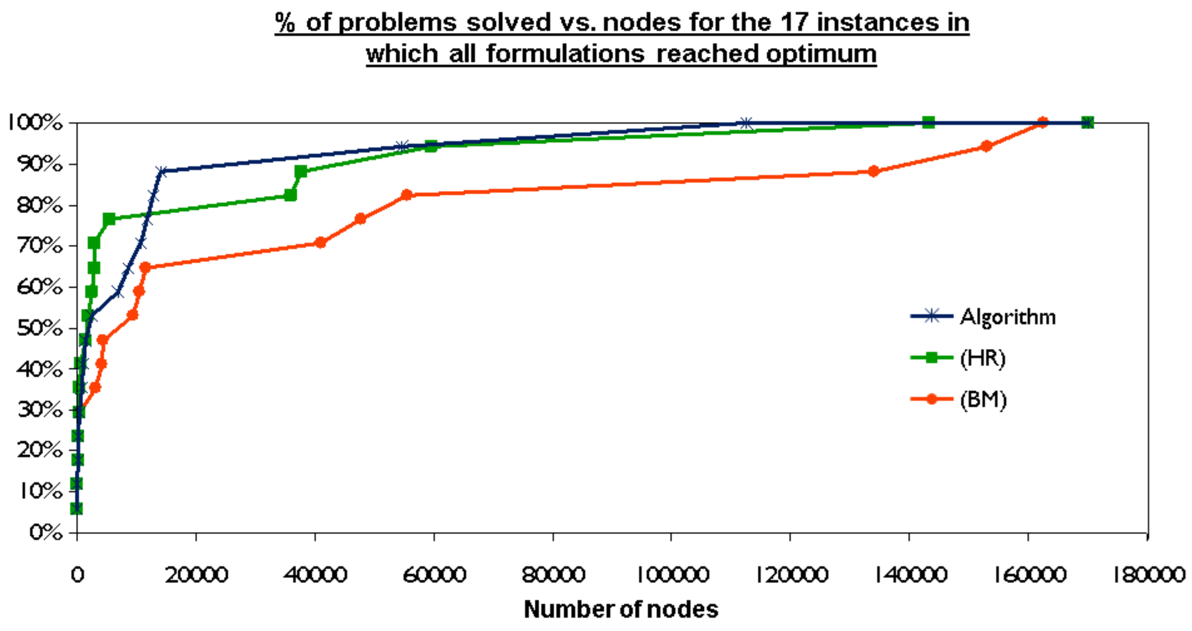


Figure 12: Number of nodes evaluated to achieve optimum, for the instances where the three formulations did so. Excludes S-Pck12 for comparison purposes, in order to avoid plotting over 1^6 nodes

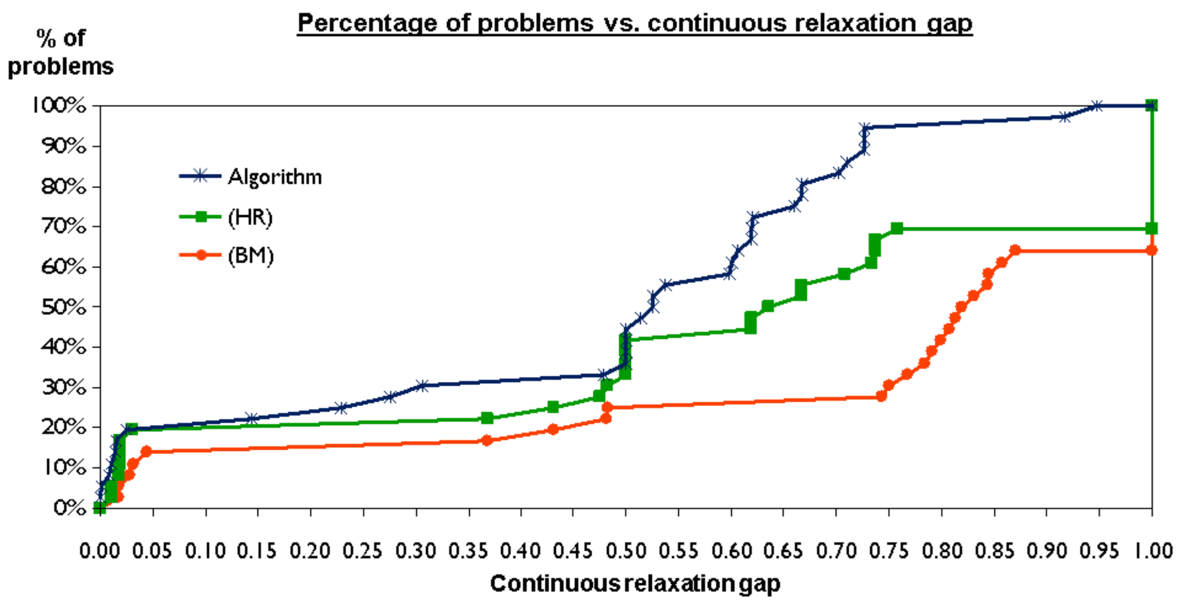


Figure 13: Relaxation gap for the different formulations

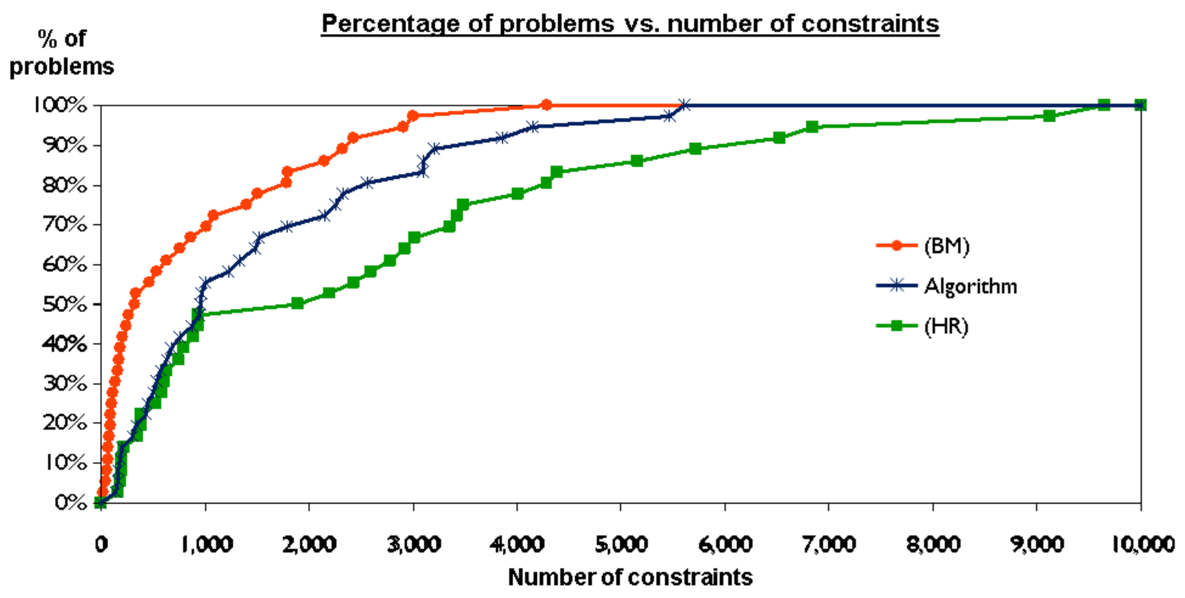


Figure 14: Number of constraints for the different formulations

improvement in the problem formulation.

Table 2 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 from Table 2 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 30%-50%, for example C-Lay-5-3 where the solution is **10,851**, the (BM) and (HR) relaxations are **0** and the relaxation after the algorithm is **4,268**. In the extreme case of Process-8, the algorithm provides a relaxation of **1,098**, which is actually the 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 10 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.

Table 3 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 21 out of the 36 instances. 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 17 of the 36 instances, and worst in 10 instances. However, the algorithm performs relatively close to the best performer in all instances, except in Batch151208. Furthermore, if we just consider the MILP/MINLP obtained with the algorithm, and not the algorithm processing time, the MILP/MINLP is best in 26 out of the 36 instances, and worst only in 2 instances. 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 (HR) is the worst formulation in general, except in the smaller instances. (BM) generally performs the best in

Table 2: Relaxation, number of constraints and number of variables for the (BM), (HR) and algorithm formulations

		Relaxation			Number of constraints			Number of variables/binary		
Instance	Solution	(BM)	(HR)	Algorithm	(BM)	(HR)	Algorithm	(BM)	(HR)	Algorithm
Batch101006	769440	734943	745909	750769	1020	1897	1004	279/129	789/129	276/120
Batch121208	1241126	1202365	1217603	1223599	1512	2781	1514	407/203	1163/203	415/203
Batch151208	1543472	1499913	1514948	1519320	1782	3348	1784	446/203	1334/203	454/203
Batch181210	2042327	2006860	2021546	2021546	2148	4011	2150	533/251	16001/251	543/251
Batch201210	2295349	2255304	2272082	2277093	2328	4389	2330	559/251	1715/251	569/251
C-Lay-3-2	41573	0	0	2200	57	195	568	32/18	92/18	240/18
C-Lay-3-3	26669	0	0	2200	69	219	338	35/21	101/21	139/20
C-Lay-4-2	6545	0	0	3025	93	349	952	54/32	166/32	390/32
C-Lay-4-3	4683	0	0	2219	109	381	968	58/36	178/36	394/36
C-Lay-5-2	17581	0	0	6685	138	530	1329	82/50	262/50	546/50
C-Lay-5-3	10851	0	0	4268	158	588	543	87/55	277/55	237/52
C-Lay-5-4	9299	0	0	2695	178	608	952	92/60	292/60	392/59
C-Lay-6-2	10262	0	0	2811	192	748	502	116/72	380/72	240/71
C-Lay-6-3	9861	0	0	2936	216	800	1478	122/78	398/78	566/69
C-Lay-6-4	17435	0	0	6998	240	888	1231	128/84	416/84	497/71
C-Lay-6-5	13227	0	0	4500	264	936	871	134/90	434/90	382/79
Dice0605	12	1.9	6	6	2432	5162	3090	1292/1260	2912/1260	2000/1051
Dice0606	12	1.6	6	6	1802	6842	3090	1766/1728	3926/1728	2002/1051
Dice0804	24	3.8	8	8	2914	6530	3860	1570/1536	3618/1536	2676/1249
Dice0805	21	3	8	8	4282	9122	5468	2282/2240	5162/2240	3544/1881
Dice1203	56	10.1	14.7	15.3	2998	9650	5610	2198/2160	5222/2160	3892/1690
DiceH0605	12	2.8	6	6	632	2432	444	1292/1260	2012/1260	1294/1250
DiceH0606	12	2.6	6	6	758	2918	2253	1766/1728	2630/1728	2766/1718
DiceH0804	24	4.8	8	8	866	3426	2556	1570/1536	2594/1536	2706/1522
DiceH0805	21	4.1	8	8	1082	4282	3205	2282/2240	3562/2240	3706/2226
DiceH1203	56	11.7	14.7	15.3	1406	5726	4149	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	54	31	31	42	45	381	141	44/24	236/24	88/24
F-lay-5	68	35.3	35.7	49.3	68	628	164	64/40	384/40	108/40
F-lay-6	67	34.6	34.6	46.5	96	936	192	88/60	568/60	132/60
Process-12	1250	0	1226.6	1248.3	121	188	635	71/12	151/12	431/11
Process-8	1098	0	1079.3	1098.2	84	163	201	48/8	116/8	120/7
S-Pck12	35	9	12.8	17	344	2192	299	290/264	1346/264	298/204
S-Pck13	59	10	14.3	28	327	2589	675	340/312	1588/312	481/202
S-Pck14	48	9	12.8	25	471	3019	759	394/364	1850/364	538/272
S-Pck15	40	10	11.7	16	542	3482	433	452/420	3482/2132	460/296

the smaller instances (C-Lay-3-2 to C-Lay-4-3). In the larger instances the algorithm performs much 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, C-Lay-6-4, 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).

Table 3: Solution time, optimality gap and B&B nodes for the (BM), (HR) and algorithm formulations

		Solution time[a] (pre-solve time)(s)			Optimality gap (%)			Number of B&B nodes[b]		
Example	Type	(BM)	(HR)	Algorithm	(BM)	(HR)	Algorithm	(BM)	(HR)	Algorithm
Batch101006	MINLP	204	12	288 (104)	0.1	0.1	0.1	9424	300	8553
Batch121208	MINLP	286	77	390 (210)	0.1	0.1	0.1	10551	1554	6886
Batch151208	MINLP	319	314	2007 (236)	0	0.1	0.1	11619	5400	54549
Batch181210	MINLP	1523	205	334 (301)	0.1	0	0.1	41053	3020	983
Batch201210	MINLP	5980	44	761 (405)	0	0.1	0.1	153039	692	10769
C-Lay-3-2	MINLP	3	6	13 (9)	0	0	0	282	379	187
C-Lay-3-3	MINLP	4	10	20 (10)	0	0	0	462	449	389
C-Lay-4-2	MINLP	33	30	39 (20)	0	0	0	4236	1880	945
C-Lay-4-3	MINLP	37	77	56 (24)	0.1	0.1	0	4341	2590	1490
C-Lay-5-2	MINLP	439	770	377 (45)	0.1	0.1	0.1	47882	37783	12791
C-Lay-5-3	MINLP	484	1022	378 (71)	0.1	0.1	0	55512	35967	14138
C-Lay-5-4	MINLP	1486	3547	740 (94)	0.1	0.1	0.1	162455	59502	11783
C-Lay-6-2	MINLP	7200	7200	4859 (96)	13.9	46.3	0.1	646748	236660	366218
C-Lay-6-3	MINLP	7200	7200	5843 (203)	55.8	40.7	0.1	550700	203360	111051
C-Lay-6-4	MINLP	7200	7200	6627 (184)	49	41.6	0	530100	96860	146571
C-Lay-6-5	MINLP	7200	7200	7200 (214)	30.4	72.1	15.4	557700	75560	304620
Dice0605	MILP	2340	7200	955 (766)	0	350	0	739393	596342	16043
Dice0606	MILP	2405	7200	1203 (1065)	0	414.3	0	363657	446270	13104
Dice0804	MILP	1436	7200	1480 (1083)	0	433.3	0	164227	518659	14802
Dice0805	MILP	7200	7200	3490 (2207)	50	540	0	1319930	290636	29728
Dice1203	MILP	7200	7200	4371 (2225)	12	380	0	570555	391152	81598
DiceH0605	MILP	3108	7200	284 (174)	0	300	0	1904882	2220152	39232
DiceH0606	MILP	5619	7200	2398 (231)	0	414.3	0	2124442	1621400	831391
DiceH0804	MILP	7200	7200	3879 (296)	433.3	276.5	0	5419210	1806748	843383
DiceH0805	MILP	7200	7200	3422 (423)	326.7	481.8	0	2884668	970273	487549
DiceH1203	MILP	7200	7200	7200 (639)	200	311.4	22.2	1433590	953789	551692
F-lay-3	MINLP	1	2	7 (5)	0	0	0	102	110	102
F-lay-4	MINLP	25	61	37 (12)	0.1	0.1	0.1	3059	2931	2546
F-lay-5	MINLP	1125	5141	1263 (28)	0.1	0.1	0.1	134026	143465	112606
F-lay-6	MINLP	7200	7200	7200 (54)	18.1	42.9	44	9037824	101775	442630
Process-12	MINLP	2	1	14 (13)	0	0	0	168	48	232
Process-8	MINLP	1	0	9 (8)	0	0	0	24	16	18
S-Pck12	MILP	100	3257	260 (118)	0	0	0	1026258	2630563	1239283
S-Pck13	MILP	7200	7200	7200 (222)	13.5	1.7	13.5	28179292	4600552	14313072
S-Pck14	MILP	2205	7200	3142 (206)	0	4.3	0	17864367	2918216	10168644
S-Pck15	MILP	7200	7200	7200 (245)	11.1	21.2	8.1	21255244	3329851	56619228

[a]Time to solve includes the processing time of the algorithm

[b]Number of nodes evaluated after two hours for the instances that reached the time limit before finding the optimal solution

7 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 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 apply basic steps. 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 and infeasible term detection, is needed to achieve faster performance. Particularly, 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.

Author Information

Corresponding Author

*E-mail: grossmann@cmu.edu.

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

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

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

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

[4] Sawaya, N. Reformulations, relaxations and cutting planes for generalized disjunctive programming. Ph.D. Thesis, Carnegie Mellon University, 2006.

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

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

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

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

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

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

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

[12] M., G. A. Generalized Benders decomposition. *Journal of Optimization Theory and Applications* **1972**, *10*, 237–260.

[13] Abhishek, K.; Leyffer, S.; Linderoth, J. FilMINT: An Outer Approximation-Based Solver for Convex Mixed-Integer Nonlinear Programs. *INFORMS Journal on Computing* **2010**, *22*, 555–567.

[14] Quesada, I.; Grossmann, I. E. An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. *Computers and Chemical Engineering* **1992**, *16*, 937–947.

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

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

[17] Bixby, R. E.; Fenelon, M.; Gu, Z.; Rothberg, E.; R., W. MIP: Theory and Practice Closing the Gap. *IFIP The International Federation for Information Processing* **2000**, *46*, 19–49.

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

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

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

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

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

[23] Lodi, A. In *50 Years of Integer Programming 1958-2008*, 2010, pp 619–645.

[24] Mahajan, A. Presolving Mixed-Integer Linear Programs. *Wiley Encyclopedia of Operations Research and Management Science* **2010**.

[25] Grossmann, I. E.; Ruiz, J. P. Generalized Disjunctive Programming: A Framework for Formulation and Alternative Algorithms for MINLP Optimization. *The IMA Volumes in Mathematics and its Applications* **2012**, *154*, 93–115.

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

[27] Clocksin, W. F.; Mellish, C. S. *Programming in Prolog*; Springer, 1981.

[28] Williams, H. P. *Model Building in Mathematical Programming*; Wiley, 1985.

[29] Biegler, L. T.; Grossmann, I. E.; Westerberg, A. W. *Systematic methods of chemical process design*; Prentice-Hall international series in the physical and chemical engineering sciences; Prentice Hall PTR, 1997.

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

[31] Balas, E. Disjunctive Programming and a Hierarchy of Relaxations for Discrete Continuous Optimization Problems. *SIAM. Journal on Algebraic and Discrete Methods* **1985**, *6*, 466–486.

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

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

[34] Hooker, J. N. Logic, Optimization, and Constraint Programming. *INFORMS Journal on Computing* **2002**, *14*, 295–321.

[35] Gardner, M. *The Colossal Book of Mathematics*; W. W. Norton, 2001.

[36] Bosh, R. A. Constructing Nontransitive Dice. *Optima, MP Society Newsletter* **2003**, *70*, 8–9.

[37] Bosh, R. A. Nontransitive Dice Revisited. *Optima, MP Society Newsletter* **2004**, *71*, 6–7.

[38] Ravemark, E. Optimization models for design and operation of chemical batch processes. Ph.D. Thesis, ETH Zurich, 1995.

[39] Vecchietti, A.; Grossmann, I. E. LOGMIP: a disjunctive 0-1 non-linear optimizer for process system models. *Computers and Chemical Engineering* **1994**, *23*, 555–565.

[40] CMU; IBM CMU-IBM Open Source MINLP Project. http://egon.cheme.cmu.edu/ibm/page.htm.

[41] Brooke, A.; Kendrick, D.; Meeraus, A.; Raman GAMS, a Users Guide. *The Scientific Press* **1998**.

Appendix

Appendix 1: Example of GDP reformulation

To illustrate a linear GDP formulation and its corresponding (HR) and (BM) reformulations, consider the optimization of a process superstructure illustrated in Figure15. In this process, either reactor 1 or reactor 2 have to be selected. For reactor 1 there are three different alternatives (R_{11}, R_{12}, R_{13}). If reactor 2 (R_2) is selected, then a choice needs to be made between using separation process 1 or 2 (S_1, S_2). A GDP representation of this problem is shown in (17).

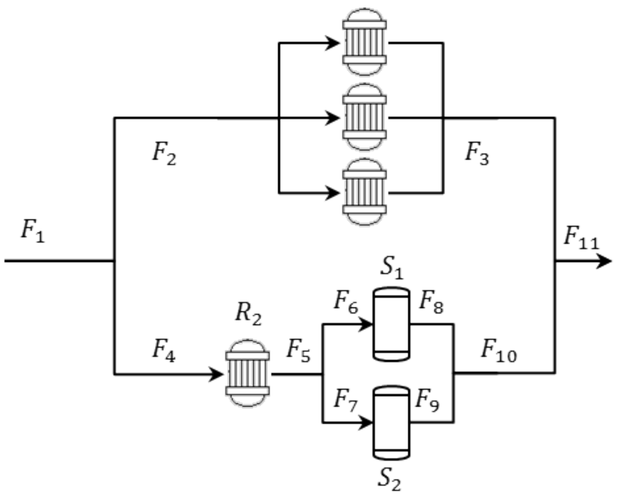


Figure 15: Process flowsheet example

$$
\begin{aligned}
& max\ z = P_{11}F_{11} - P_1F_1 - C_R - C_S \\
s.t.\quad & F_1 = F_2 + F_4 \\
& F_5 = F_6 + F_7 \\
& F_{10} = F_8 + F_9 \\
& F_{11} = F_3 + F_{10} \\
& \begin{bmatrix} Y_{R11} \\ F_3 = \beta_{R11}F_2 \\ F_{4,5,6,7,8,9,10} = 0 \\ C_R = \gamma_{R11} \end{bmatrix} \vee \begin{bmatrix} Y_{R12} \\ F_3 = \beta_{R12}F_2 \\ F_{4,5,6,7,8,9,10} = 0 \\ C_R = \gamma_{R12} \end{bmatrix} \vee \begin{bmatrix} Y_{R13} \\ F_3 = \beta_{R13}F_2 \\ F_{4,5,6,7,8,9,10} = 0 \\ C_R = \gamma_{R13} \end{bmatrix} \vee \begin{bmatrix} Y_{R2} \\ F_5 = \beta_{R2}F_4 \\ F_{2,3} = 0 \\ C_R = \gamma_{R2} \end{bmatrix} \\
& \begin{bmatrix} Y_{S1} \\ F_8 = \beta_{S1}F_6 \\ C_S = \gamma_{S1} \end{bmatrix} \vee \begin{bmatrix} Y_{S2} \\ F_9 = \beta_{S1}F_7 \\ C_S = \gamma_{S2} \end{bmatrix} \vee \begin{bmatrix} Y_{notR2} \\ C_S = 0 \end{bmatrix} \\
& Y_{R11} \veebar Y_{R12} \veebar Y_{R13} \veebar Y_{R2} \\
& Y_{S1} \veebar Y_{S2} \veebar Y_{notR2} \\
& Y_{R2} \Leftrightarrow Y_{S1} \vee Y_{S2} \\
& 0 \leq F_i \leq F_i^{up} \qquad\qquad i = 1, ..., 11 \\
& 0 \leq C_{R,S} \leq C^{up} \\
& Y_k \in \{True, False\} \qquad k \in \{R11, R12, R13, R2, S1, S2, notR2\}
\end{aligned}
\tag{17}
$$

In (17) F_i represents the flows in the superstructure. Y_k represents the selection of an equipment. C_R and C_S represent the cost of the reactor and the cost of the separation unit respectively. The global constraints represent the material balances in the nodes of the flowsheet. The first disjunction allows the selection of any of the three alternatives of reactor 1 or the selection of reactor 2. When the alternative of reactor 1 is selected, F_3 becomes a function of F_2, the flows associated with R_2 become 0, and the reactor cost takes a value that corresponds to that selection. When R_2 is selected, F_5 becomes proportional to F_4, the flows associated with rector 1 (F_2, F_3) become 0, and C_R takes a value that corresponds to the cost of reactor 2. The second disjunction represents the selection of either one of the two separation processes or no separation process (when reactor 2 is not selected). The flows after the separation process and the cost of separation unit are associated with this decision. The logic constraint forces that if and only if Y_{R2} is selected then either Y_{S1} or Y_{S2} has to be selected. Note that this constraint implies that if Y_{R11} or Y_{R12} or Y_{R13} is selected, then Y_{notR2} is selected.

The (BM) and (HR) reformulations of this example are shown in (18) and (19) respectively. Note that in (18) we are additionally relaxing some of the equality constraints as inequalities, in order to avoid additional constraints. It is easy to show that the problem with inequality relaxations is equivalent to the original problem.

$$\begin{aligned}
& max\ z = P_{11}F_{11} - P_1F_1 - C_R - C_S \\
s.t.\quad & F_1 = F_2 + F_4 \\
& F_5 = F_6 + F_7 \\
& F_{10} = F_8 + F_9 \\
& F_{11} = F_3 + F_{10} \\
& F_3 \le \beta_{R11}F_2 + M(1 - y_{R11}) \\
& F_{4,5,6,7,8,9,10} \le M(1 - y_{R11}) \\
& C_R \ge \gamma_{R11} - M(1 - y_{R11}) \\
& F_3 \le \beta_{R12}F_2 + M(1 - y_{R12}) \\
& F_{4,5,6,7,8,9,10} \le M(1 - y_{R12}) \\
& C_R \ge \gamma_{R12} - M(1 - y_{R12}) \\
& F_3 \le \beta_{R13}F_2 + M(1 - y_{R13}) \\
& F_{4,5,6,7,8,9,10} \le M(1 - y_{R13}) \\
& C_R \ge \gamma_{R13} - M(1 - y_{R13}) \\
& F_5 \le \beta_{R2}F_4 + M(1 - y_{R2}) \\
& F_{2,3} \le M(1 - y_{R2}) \\
& C_R \ge \gamma_{R2} - M(1 - y_{R2}) \\
& F_8 \le \beta_{S1}F_6 + M(1 - y_{S1}) \\
& C_S \ge \gamma_{S1} - M(1 - y_{S1}) \\
& F_9 \le \beta_{S2}F_7 + M(1 - y_{S2}) \\
& C_S \ge \gamma_{S2} - M(1 - y_{S2}) \\
& C_S \le M(1 - y_{notR2}) \\
& y_{R11} + y_{R12} + y_{R13} + y_{R2} = 1 \\
& y_{S1} + y_{S2} + y_{notR2} = 1 \\
& y_{R2} = y_{S1} + y_{S2} \\
& 0 \le F_i \le F_i^{up} \qquad\qquad i = 1, ..., 11 \\
& 0 \le C_{R,S} \le C^{up} \\
& y_k \in \{0, 1\} \qquad\qquad k \in \{R11, R12, R13, R2, S1, S2, notR2\}
\end{aligned} \tag{18}$$

$$
\begin{aligned}
& max\ z = P_{11}F_{11} - P_1F_1 - C_R - C_S \\
s.t. \quad & F_1 = F_2 + F_4 \\
& F_5 = F_6 + F_7 \\
& F_{10} = F_8 + F_9 \\
& F_{11} = F_3 + F_{10} \\
& F_i = F_i^{R11} + F_i^{R12} + F_i^{R13} + F_i^{R2} \qquad i = 1, ..., 11 \\
& F_i = F_i^{S1} + F_i^{S2} + F_i^{notR2} \qquad i = 1, ..., 11 \\
& C_R = C_R^{R11} + C_R^{R12} + C_R^{R13} + C_R^{R2} \\
& C_S = C_S^{S1} + C_S^{S2} + C_S^{notR2} \\
& F_3^{R11} = \beta_{R11}F_2^{R11} \\
& F_{4,5,6,7,8,9,10}^{R11} = 0 \\
& C_R^{R11} = \gamma_{R11} * y_{R11} \\
& F_3^{R12} = \beta_{R12}F_2^{R12} \\
& F_{4,5,6,7,8,9,10}^{R12} = 0 \\
& C_R^{R12} = \gamma_{R12} * y_{R12} \\
& F_3^{R13} = \beta_{R13}F_2^{R13} \\
& F_{4,5,6,7,8,9,10}^{R13} = 0 \\
& C_R^{R13} = \gamma_{R13} * y_{R13} \\
& F_5^{R2} = \beta_{R2}F_4^{R}2 \\
& F_{2,3}^{R2} = 0 \\
& C_R^{R2} = \gamma_{R2} * y_{R2} \\
& F_8^{S1} = \beta_{S1}F_6^{S1} \\
& C_S^{S1} = \gamma_{S1} * y_{S1} \\
& F_9^{S2} = \beta_{S2}F_7^{S2} \\
& C_S^{S2} = \gamma_{S2} * y_{S2} \\
& C_S^{notR2} = 0 \\
& y_{R11} + y_{R12} + y_{R13} + y_{R2} = 1 \\
& y_{S1} + y_{S2} + y_{notR2} = 1 \\
& y_{R2} = y_{S1} + y_{S2} \\
& 0 \leq F_i^k \leq F_i^{up} * y_k \qquad i = 1, ..., 11, k \in \{R11, R12, R13, R2, S1, S2, notR2\} \\
& 0 \leq C_{R,S}^k \leq C^{up} * y_k \qquad k \in \{R11, R12, R13, R2, S1, S2, notR2\} \\
& y_k \in \{0, 1\} \qquad k \in \{R11, R12, R13, R2, S1, S2, notR2\}
\end{aligned}
\tag{19}
$$

Appendix 2: Hybrid reformulation of algorithm example

$$
\begin{aligned}
& min\ lt \\
s.t. \quad & lt \geq x_4 + 3
\end{aligned}
$$

$$x_j = x_j^{1,1} + x_j^{1,2} + x_j^{2,1} + x_j^{2,2} \qquad j = 1,2,3$$
$$lt = lt^{1,1} + lt^{1,2} + lt^{2,1} + lt^{2,2}$$
$$lt^{1,1} \geq x_1^{1,1} + 6 * \hat{y}_{1,1}$$
$$lt^{1,1} \geq x_2^{1,1} + 5 * \hat{y}_{1,1}$$
$$lt^{1,1} \geq x_3^{1,1} + 4 * \hat{y}_{1,1}$$
$$x_1^{1,1} + 6 * \hat{y}_{1,1} \leq x_2^{1,1}$$
$$x_1^{1,1} + 6 * \hat{y}_{1,1} \leq x_3^{1,1}$$
$$lt^{1,2} \geq x_1^{1,2} + 6 * \hat{y}_{1,2}$$
$$lt^{1,2} \geq x_2^{1,2} + 5 * \hat{y}_{1,2}$$
$$lt^{1,2} \geq x_3^{1,2} + 4 * \hat{y}_{1,2}$$
$$x_1^{1,2} + 6 * \hat{y}_{1,2} \leq x_2^{1,2}$$
$$x_3^{1,2} + 4 * \hat{y}_{1,2} \leq x_1^{1,2}$$
$$lt^{2,1} \geq x_1^{2,1} + 6 * \hat{y}_{2,1}$$
$$lt^{2,1} \geq x_2^{2,1} + 5 * \hat{y}_{2,1}$$
$$lt^{2,1} \geq x_3^{2,1} + 4 * \hat{y}_{2,1}$$
$$x_2^{2,1} + 5 * \hat{y}_{2,1} \leq x_1^{2,1}$$
$$x_1^{2,1} + 6 * \hat{y}_{2,1} \leq x_3^{2,1}$$
$$lt^{2,2} \geq x_1^{2,2} + 6 * \hat{y}_{2,2}$$
$$lt^{2,2} \geq x_2^{2,2} + 5 * \hat{y}_{2,2}$$
$$lt^{2,2} \geq x_3^{2,2} + 4 * \hat{y}_{2,2}$$
$$x_2^{2,2} + 5 * \hat{y}_{2,2} \leq x_1^{2,2}$$
$$x_3^{2,2} + 4 * \hat{y}_{2,2} \leq x_1^{2,2} \qquad (20)$$
$$x_1 - x_4 + 6 \leq M * (1 - y_{31})$$
$$x_4 - x_1 + 3 \leq M * (1 - y_{32})$$
$$h_4 - h_1 + 6 \leq M * (1 - y_{33})$$
$$h_1 - h_4 + 3 \leq M * (1 - y_{34})$$
$$x_2 - x_3 + 5 \leq M * (1 - y_{41})$$
$$x_3 - x_2 + 4 \leq M * (1 - y_{42})$$
$$x_2 - x_4 + 5 \leq M * (1 - y_{51})$$
$$x_4 - x_2 + 3 \leq M * (1 - y_{52})$$
$$h_4 - h_2 + 7 \leq M * (1 - y_{53})$$
$$h_2 - h_4 + 3 \leq M * (1 - y_{54})$$
$$x_3 - x_4 + 4 \leq M * (1 - y_{61})$$
$$x_4 - x_3 + 3 \leq M * (1 - y_{62})$$
$$h_4 - h_3 + 5 \leq M * (1 - y_{63})$$
$$h_3 - h_4 + 3 \leq M * (1 - y_{64})$$
$$y_{k1} + y_{k2} + y_{k3} + y_{k4} = 1 \qquad k = 3,5,6$$
$$y_{k1} + y_{k2} = 1 \qquad k = 1,2,4$$
$$y_{11} = \hat{y}_{1,1} + \hat{y}_{1,2}$$

$$y_{12} = \hat{y}_{2,1} + \hat{y}_{2,2}$$

$$y_{21} = \hat{y}_{1,1} + \hat{y}_{2,1}$$

$$y_{22} = \hat{y}_{1,2} + \hat{y}_{2,2}$$

$$0 \le x_1^{\hat{i}} \le 12\hat{y}_{\hat{i}} \qquad \hat{i} = \{(1,1),(1,2),(2,1),(2,2)\}$$

$$0 \le x_2^{\hat{i}} \le 13\hat{y}_{\hat{i}} \qquad \hat{i} = \{(1,1),(1,2),(2,1),(2,2)\}$$

$$0 \le x_3^{\hat{i}} \le 14\hat{y}_{\hat{i}} \qquad \hat{i} = \{(1,1),(1,2),(2,1),(2,2)\}$$

$$0 \le x_3^{\hat{i}} \le 15\hat{y}_{\hat{i}} \qquad \hat{i} = \{(1,1),(1,2),(2,1),(2,2)\}$$

$$0 \le lt^{\hat{i}} \le 20\hat{y}_{\hat{i}} \qquad \hat{i} = \{(1,1),(1,2),(2,1),(2,2)\}$$

$$0 \le lt \le 20;\ 0 \le x_1 \le 12;\ 0 \le x_2 \le 13;\ 0 \le x_3 \le 14;\ 0 \le x_4 \le 15$$

$$6 \le h_1 \le 10;\ 7 \le h_2 \le 10;\ 5 \le h_3 \le 10;\ 3 \le h_4 \le 10$$

$$y_{ki} \in \{0,1\} \qquad k = 1,...,6,\ i \in D_k$$

$$0 \le \hat{y}_{1,1}, \hat{y}_{1,2}, \hat{y}_{2,1}, \hat{y}_{2,2} \le 1$$

$$x_j, h_j \in \mathbb{R}^1 \qquad j = 1,2,3,4$$

Appendix 3: Nomenclature for design of a multi-product batch plant example

Given:

$\alpha_1, \alpha_2, \beta_1, \beta_2$: Coefficients for the capital cost of the units and intermediate storage tanks.

$i \in I$: products.

$j \in J$: stages.

H: horizon time.

Q_i: production rate of product i.

T_{ij}: processing time of product i at stage j.

S_{ij}: size factor of product i at stage j.

S_j^*: size factor for intermediate storage tank.

S_{ij}^*: size factor for stages.

Determine:

B_{ij}: batch size product i at stage j.

E_i: inverse production rate for product i.

M_j: number of units in parallel out-of-phase at stage j.

N_j: number of units in parallel in phase at stage j.

V_j: Unit size of stage j.

V_{Tj}: size of intermediate storage tank between stage j and $j+1$.

In order to convexify the problem, the following variables are introduced:

$b_{ij} = \ln(B_{ij})$

$e_i = \ln(E_i)$

$m_j = \ln(M_j)$

$n_j = \ln(N_j)$

$v_j = \ln(V_j)$

$v_{Tj} = \ln(V_{Tj})$