

Lagrangian Relaxation of the Hull-Reformulation of Linear Generalized Disjunctive Programs and its use in Disjunctive Branch and Bound

Francisco Trespalacios, Ignacio E. Grossmann*

*Department of Chemical Engineering, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213, USA*

Abstract

In this work, we present a Lagrangian relaxation of the hull-reformulation of discrete-continuous optimization problems formulated as linear generalized disjunctive programs (GDP). The proposed Lagrangian relaxation has three important properties. The first property is that it can be applied to any linear GDP. The second property is that the solution to its continuous relaxation always yields 0-1 values for the binary variables of the hull-reformulation. Finally, it is simpler to solve than the continuous relaxation of the hull-reformulation. The proposed Lagrangian relaxation can be used in different GDP solution methods. In this work, we explore its use as primal heuristic to find feasible solutions in a disjunctive branch and bound algorithm. The modified disjunctive branch and bound is tested with several instances. The results show that the proposed disjunctive branch and bound performs better than other versions of the algorithm that do not include this primal heuristic.

1. Introduction

Lagrangian relaxation of an optimization program is a widely-used and powerful tool to solve problems. The review work by Guignard[1] discusses how Lagrangian relaxation can be used in different solution methods and applications. Fisher[2] provides a theoretical background for Lagrangian relaxation of mixed-integer linear programs (MILP). The general idea in the Lagrangian relaxation is to “dualize” some of the constraints in the optimization problem (i.e. remove some constraints from the feasible region of the problem, and penalize the violation of such constraints in the objective function). This approach is particularly useful in problems with complicating constraints. Some of these problems appear in planning[3], scheduling[4], facility location[5], and stochastic programming problems[6]. In this type of problems, a Lagrangian relaxation is simpler to solve than the original problems.

A particular method that uses Lagrangian relaxation to solve MILPs is the Lagrangian relaxation based branch and bound[7]. This method follows the same general idea as the LP based branch and bound, but it

*Corresponding author

Email address: grossmann@cmu.com (Ignacio E. Grossmann)

solves the Lagrangean relaxation at every node instead of the LP relaxation. This method can be useful in problems in which, by dualizing the complicating constraints, the Lagrangean relaxation is simpler to solve than the LP relaxation. One of the main difficulties in automating this strategy, or any other method that uses Lagrangean relaxation, is identifying the complicating constraints, which can be non trivial and problem specific. Typically, the modeller needs to identify the problem structure and select the constraints to dualize, or needs to modify the model to allow such a structure[1].

Linear discrete-continuous optimization problems are typically formulated as MILPs. An alternative framework for modelling these problems is generalized disjunctive programming (GDP)[8]. GDP models discrete-continuous problems through the use of disjunctions, algebraic equations, and Boolean and continuous variables. Linear GDP problems can be reformulated as mixed-integer linear programs (MILP) and solved with existing MILP solvers. The GDP-to-MILP reformulations are the Big-M (BM)[9], multiple-parameter Big-M (MBM)[10] and Hull reformulation (HR)[11, 12]. The HR reformulations is at least as tight, and typically tighter, than the other two. The downside of the HR is that it yields a larger MILP formulation. Alternatively to the MILP reformulation, GDP problems can be solved with specialized algorithms. In the case of linear GDP problems, the disjunctive branch and bound is a powerful solution method[11, 12].

In this work we first present a Lagrangean relaxation of the HR for linear GDPs. The proposed Lagrangean relaxation is an MILP, and it has three important characteristics. The first one is that the solution to the continuous relaxation of the proposed Lagrangean relaxation always yields 0-1 values for the binary variables of the HR. The second one is that it is easier to solve than the original problem (i.e. the HR). Furthermore, it is easier to solve than the continuous relaxation of the HR. The third one is that this relaxation can be applied to any linear GDP. This means that there is no need to specify which are the complicating constraints in different problems, so automating a method that uses this Lagrangean relaxation can be achieved. We use the proposed Lagrangean relaxation to improve the performance of the disjunctive branch and bound algorithm. In particular, we evaluate the Lagrangean relaxation at every node and use its solution as heuristic for finding feasible solutions to the problem. The continuous relaxation of the Lagrangean relaxation always provides 0-1 values to the binary variables, so the value of the 0-1 variables is fixed and a small LP is solved in search of feasible solutions.

This paper is organized as follows. Section 2 presents a brief background on Lagrangean relaxation of MILPs, generalized disjunctive programming and the disjunctive branch and bound. Section 3 presents the proposed Lagrangean relaxation of the HR. This section presents the formulation and main properties. The proposed Lagrangean relaxation is then incorporated into a disjunctive branch and bound, which is presented in Section 4. Section 5 demonstrates the performance of the proposed disjunctive branch and bound in an illustrative example. The performance of the disjunctive branch and bound with the Lagrangean relaxation is evaluated against other versions of the disjunctive branch and bound with several instances. The results of

these instances are presented in Section 6.

2. Background

2.1. Lagrangean relaxation of mixed-integer linear programs

50 In this section we present a brief review of the Lagrangean relaxation of mixed-integer linear programs. In this work, we consider the complicating constraints to be equality constraints. We refer the reader to the work by Guignard[1] for a comprehensive review and for proofs of the Theorems and relations presented in this section. Throughout the manuscript, for any given optimization problem (Q) we denote $v(Q)$ its optimal value and $F(Q)$ its feasible region.

Without loss of generality, consider the following general mixed-integer linear program:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & Cx \leq d \\ & x \in X \end{aligned} \tag{P}$$

55 where X contains the integrality and sign restrictions on x (e.g. $X = \mathbb{R}_+^{n-q} \times \{0, 1\}^q$). Consider that $Ax = b$ are the complicating constraints (i.e. the problem becomes much simpler to solve without them). Let λ be a vector of weights, namely the Lagrangean multipliers.

The Lagrangean relaxation of (P) is:

$$\begin{aligned} \min \quad & c^T x + \lambda(Ax - b) \\ \text{s.t.} \quad & Cx \leq d \\ & x \in X \end{aligned} \tag{LR1}_\lambda$$

In $(LR1)_\lambda$, the complicating constraints $(Ax = b)$ have been “dualized” (i.e. the slacks of the complicating constraints have been added to the objective function, and the complicating constraints dropped from
60 the formulation). Note that if these constraints are inequalities $(Ax \leq b)$, then the corresponding Lagrange multipliers are non-negative.

It is easy to see that $(LR1)_\lambda$ is a relaxation of (P), since $F(P) \subseteq F(LR1)_\lambda$. Therefore, $v(LR1)_\lambda \leq v(P)$ in general. When $x \in F(P)$ then $v(LR1)_\lambda = v(P)$ (when the complicating constraints are equalities).

Theorem 2.1.

- 65 1. If $x(\lambda)$ is an optimal solution of $(LR1)_\lambda$ for some λ , then $c^T x(\lambda) + \lambda(Ax - b) \leq v(P)$.
2. If in addition $x(\lambda)$ is feasible for (P), then $x(\lambda)$ is an optimal solution of (P), and $c^T x(\lambda) = v(P)$

Theorem 2.1 states that Lagrangean relaxation always provides a lower bound for the MILP problem. The best possible lower bound that the Lagrangean relaxation provides can be obtained with the following optimization problem:

$$\max_{\lambda} v(LR1_{\lambda}) \quad (\text{LD})$$

Problem (LD) is called the Lagrangean dual of (P) with respect to the complicating constraints $Ax = b$. Let (RP) be the continuous relaxation of (P). In general, $v(RP) \leq v(\text{LD})$. In the particular case in which the Lagrangean dual has the integrality property (i.e. the extreme points of $\{x | Cx \leq d\}$ are in X), $v(RP) = v(\text{LD})$.

In this work, we present a Lagrangean relaxation applicable to the MILP reformulation of problems formulated as GDPs. In the next section, we introduce the GDP formulation and the two main GDP-to-MILP reformulations.

2.2. Linear generalized disjunctive programming

GDP is an alternative framework for modelling discrete-continuous optimization problems. In this section we present the formulation for linear GDPs, as well as the two traditional GDP-to-MILP reformulations: the BM and the HR. For a comprehensive review on formulating GDP problems, as well as the theory for general nonlinear GDPs, we refer the reader to Grossmann and Trespalcios[13].

The general linear GDP formulation can be represented as follows:

$$\begin{aligned} & \min c^T x \\ & \text{s.t.} \quad Gx \leq g \\ & \quad \bigvee_{i \in D_k} \left[\begin{array}{c} Y_{ki} \\ A^{ki}x \leq a^{ki} \end{array} \right] \quad k \in K \\ & \quad \bigvee_{i \in D_k} Y_{ki} \quad k \in K \\ & \quad \Omega(Y) = \text{True} \\ & \quad x^{lo} \leq x \leq x^{up} \\ & \quad x \in \mathbb{R}^n \\ & \quad Y_{ki} \in \{\text{True}, \text{False}\} \quad k \in K, i \in D_k \end{aligned} \quad (\text{GDP})$$

In (GDP), the objective is to minimize a linear function of the continuous variables $x \in \mathbb{R}^n$. The global constraints of the problem ($Gx \leq g$) are enforced regardless of the discrete decisions. In (GDP) there is a set of disjunctions $k \in K$. Each disjunction involves $i \in D_k$ disjunctive terms that are connected by an "or" operator (\vee). Each disjunctive term has an associated polyhedron ($A^{ki}x \leq a^{ki}$) and a Boolean variables (Y_{ki}). Exactly one disjunctive term must be selected in each disjunction ($\bigvee_{i \in D_k} Y_{ki}$). The constraints corresponding

to the selected disjunctive term ($Y_{ki} = True$) are enforced, while the ones corresponding to non-selected terms ($Y_{ki} = False$) are ignored. Finally, the $\Omega(Y) = True$ represents the logic relations between the Boolean.

The (BM) reformulation is as follows:

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

90 In (BM), the objective function and global constraints remain unchanged. The Boolean variables Y_{ki} are replaced by binary variables with a one-to-one correspondence: $Y_{ki} = True$ is equivalent to $y_{ki} = 1$ and $Y_{ki} = False$ is equivalent to $y_{ki} = 0$. Constraint $\sum_{i \in D_k} y_{ki} = 1$ enforces that exactly one disjunctive term is selected per disjunction. When a disjunctive term is selected ($y_{ki} = 1$), the corresponding constraints are enforced ($A^{ki}x \leq a^{ki}$). When a term is not selected ($y_{ki} = 0$), the corresponding constraints become
95 redundant for a large enough M^{ki} ($A^{ki}x \leq a^{ki} + M^{ki}$). The logic constraints $\Omega(Y) = True$ are easily transformed into integer linear constraints ($Hy \geq h$)[14, 13].

The (HR) formulation is given as follows:

$$\begin{aligned}
& \min c^T x \\
s.t. \quad & Gx \leq g \\
& x = \sum_{i \in D_k} \nu^{ki} \quad k \in K \\
& A^{ki}\nu^{ki} \leq a^{ki}y_{ki} \quad k \in K, i \in D_k \\
& \sum_{i \in D_k} y_{ki} = 1 \quad k \in K \\
& Hy \geq h \\
& x^{lo}y_{ki} \leq \nu^{ki} \leq x^{up}y_{ki} \quad k \in K, i \in D_k \\
& x \in \mathbb{R}^n \\
& y_{ki} \in \{0, 1\} \quad k \in K, i \in D_k
\end{aligned} \tag{HR}$$

In (HR), the objective function and global constraints remain unchanged, the Boolean variables are replaced by binary variables, and the logic relations transformed into linear constraints. In this reformulation, the continuous variables are disaggregated. The constraint $x^{lo}y_{ki} \leq \nu^{ki} \leq x^{up}y_{ki}$ enforces that when a term

is selected, the corresponding disaggregated variable must lie within the bounds. When it is not selected, the disaggregated variable becomes zero. The constraint $x = \sum_{i \in D_k} \nu^{ki}$ enforces that the variables x take the same value as the disaggregated variables of the active terms. When a term is selected ($y_{ki} = 1$) the constraint is enforced for the disaggregated variable ($A^{ki} \nu^{ki} \leq a^{ki}$). When it is not active ($y_{ki} = 0$), the constraint is trivially satisfied ($0 \leq 0$). Note that this reformulation for linear GDPs is equivalent to the convex hull representation in disjunctive programming[15].

The (BM) reformulation generates a smaller MILP, while the (HR) provides a formulation that is as tight, and generally tighter[16].

The solution of linear GDPs typically involve reformulating them as MILPs, and then solved using an LP-based branch and bound method (or one of its variants). Alternatively, the GDPs can be solved by a specialized algorithm: the disjunctive branch and bound.

2.3. Disjunctive branch and bound

The idea behind the disjunctive branch and bound[11, 12] is to branch directly on the disjunctions, while using the continuous relaxation of the BM or HR of the remaining disjunctions. Let (R-BM) and (R-HR) be the continuous relaxation of (BM) and (HR), respectively. The disjunctive branch and bound is as follows:

For a node N_p , let z^p denote the optimal value of (R-BM) or (R-HR) of the corresponding GDP_p , and (x^p, y^p) its solution. Let L be the set of nodes to be solved, and GDP_0 be the original GDP. Let z^{up} be an upper bound for the optimal value of the objective function z^* .

0. *Initialize.* Set $L = N_0$, $z^{up} = \infty$, $(x^*, y^*) = \emptyset$.

1. *Terminate.* If $L = \emptyset$, then (x^*, y^*) is optimal and algorithm terminates.

2. *Node selection.* Choose a node $N_p \in L$, and delete it from L . Go either to 3a or to 3b.

3a. *Bound.* Solve the (R-HR) of GDP_p corresponding to N_p . If it is infeasible, go to step 1. Else, let z^p be its objective function and (x^p, y^p) its solution.

3b. *Bound.* Solve the (R-BM) of GDP_p corresponding to N_p . If it is infeasible, go to step 1. Else, let z^p be its objective function and (x^p, y^p) its solution.

4. *Prune.* If $z^p \geq z^{up}$, go to step 1.

If $y^p \in \mathbb{Z}^q$ let $z^{up} = z^p$ and $(x^*, y^*) = (x^p, y^p)$. Delete all nodes $N_r \in L$ in which $z^r \geq z^{up}$, and go to step 1. Else, go to step 5.

5. *Branch.* Select a disjunction $k \in K$ such that $y_{ki} \notin \{0, 1\}$ for some $i \in D_k$. For every $i \in D_k$, construct the corresponding GDP (GDP_p^i) by setting the constraints corresponding to the disjunctive term i as global, and removing the Boolean variables and constraints corresponding to term $i' \neq i; i' \in D_k$. Add $|D_k|$ new nodes, corresponding to GDP_p^i , to L . Go to step 1.

It is easy to see that this algorithm terminates finitely, in the worst case evaluating every possible node. This algorithm can be trivially modified to consider a tolerance for termination $\epsilon > 0$. The HR disjunctive

branch and bound makes use of Step 3a at every node, while the BM disjunctive branch and bound makes use of Step 3b at every node. It is also possible to have a hybrid disjunctive branch and bound in which some nodes are solved using the BM and others using the HR. It is important to note that, as the disjunctive branch and bound progresses, the GDP problems that correspond to each node become smaller. In particular, the constraints that correspond to the disjunctive terms that were not selected are removed from the problem formulation in the subsequent nodes.

Note that the worst case involves $\prod_{k \in D_k} |D_k|$ leaf nodes, which is fewer than the worst case number of leaf nodes in a binary branch and bound algorithm (bounded by $2^{\sum_{k \in K} (|D_k| - 1)}$) except when $|D_k| = 2, \forall k \in K$ (in which case the maximum number of leaf nodes for both algorithms is $2^{|K|}$). The worst case for number of evaluated nodes in the disjunctive branch and bound depends on the sequence in which the disjunctions were branched, but it is smaller than the binary branch and bound except when $|D_k| = 2, \forall k$ (in which case it is the same: $2^{|K|+1} - 1$).

It has been shown that the disjunctive branch and bound has advantages over the binary branch and bound[11]. In this work, we improve the search of feasible solutions in the disjunctive branch and bound by solving a Lagrangean relaxation of the HR at every node.

3. Lagrangean relaxation of the hull-reformulation of GDP

In this section we present a Lagrangean relaxation of the HR of any linear GDP. This relaxation is easier to solve than the continuous relaxation of the HR. The continuous relaxation of the proposed Lagrangean relaxation can be proved to always yield 0-1 values to the binary variables of the reformulation. We first present the Lagrangean relaxation for the case in which GDP does not involve logic relations, and we then discuss its extension to the case in which it does. We note that these properties can be extended to nonlinear convex GDP. This is achieved by using the theory for convex GDPs by Ruiz and Grossmann[17], which extends part of the rich theory of disjunctive programming[15] to GDPs with convex constraints.

3.1. Lagrangean relaxation of the hull-reformulation of GDP without logic relations

For given Lagrangean multipliers (λ_{kj}) , the following Lagrangean relaxation can be applied to the hull-reformulation of any linear GDP:

$$\begin{aligned}
& \min c^T x + \sum_{k \in K} \sum_{j \in J_k} \lambda_{kj} (x_j - \sum_{i \in D_k} \nu^{ki}) \\
& s.t. \quad Gx \leq g \\
& \quad A^{ki} \nu^{ki} \leq a^{ki} y_{ki} \quad k \in K, i \in D_k \\
& \quad \sum_{i \in D_k} y_{ki} = 1 \quad k \in K \quad (LHR_\lambda) \\
& \quad x^{lo} y_{ki} \leq \nu^{ki} \leq x^{up} y_{ki} \quad k \in K, i \in D_k \\
& \quad x \in \mathbb{R}^n \\
& \quad y_{ki} \in \{0, 1\} \quad k \in K, i \in D_k
\end{aligned}$$

where J_k is the set of variables that appear in disjunction k .

Property 3.1. *The Lagrangean relaxation (LHR_λ) can be applied to any linear GDP.*

160 Property 3.1 is trivial, since the decomposition is applied to the MILP reformulation of the general form of linear GDP.

Note that in problem (LHR_λ) the variables x_j and ν^{ki} do not appear together in any constraint. Furthermore, variables ν^{ki} and $\nu^{k'i'}$ for $k' \neq k; i \in D_k; i' \in D_{k'}$ do not appear together in any constraint either. Therefore, (LHR_λ) can be decomposed into $|K| + 1$ simpler problems.

The first problem, that involves only the continuous variables and global constraints, is as follows:

$$\begin{aligned}
& \min c^T x + \sum_{k \in K} \sum_{j \in J_k} \lambda_{kj} x_j \\
& s.t. \quad Gx \leq g \\
& \quad x \in \mathbb{R}^n \quad (LHR_0)
\end{aligned}$$

The remaining $k \in K$ problems, each one containing the ν^{ki} variables corresponding to disjunction k , are as follows:

$$\begin{aligned}
& \min - \sum_{j \in J_k} \lambda_{kj} \left(\sum_{i \in D_k} \nu^{ki} \right) \\
& s.t. \quad A^{ki} \nu^{ki} \leq a^{ki} y_{ki} \quad k \in K, i \in D_k \\
& \quad \sum_{i \in D_k} y_{ki} = 1 \quad k \in K \quad (LHR_k) \\
& \quad x^{lo} y_{ki} \leq \nu^{ki} \leq x^{up} y_{ki} \quad k \in K, i \in D_k \\
& \quad y_{ki} \in \{0, 1\} \quad k \in K, i \in D_k
\end{aligned}$$

165 **Property 3.2.** *The optimal solution of (LHR_λ) can be obtained by the summation of the optimal values of $|K| + 1$ problems: $v(LHR_\lambda) = v(LHR_0) + \sum_{k \in K} v(LHR_k)$.*

Property 3.2 indicates that it is possible to solve $|K| + 1$ smaller MILPs to solve (LHR_λ) . Furthermore, (LHR_λ) and each of these problems can be solved as an LP, as is shown in Property 3.3.

Property 3.3. *Let (\hat{x}, \hat{y}) be a vertex of the continuous relaxation of (LHR_λ) . Then, for every $k \in K$ there exists an $i \in D_k$ such that $\hat{y}_{ki} = 1$, and $\hat{y}_{ki'} = 0, \forall i' \in D_k, i' \neq i$.*

Proof. The proof is presented by analyzing the decomposed problem. Problem (LHR_0) does not involve any binary variable or disaggregated as decision variable. In problems (LHR_k) , only the binary variables y_{ki} and the disaggregated variables ν_{ki} , that correspond to the disjunction $(k \in D_k)$ are optimization variables. From Corollary 2.1.2 of Balas[15], for any vertex (\bar{v}, \bar{y}) of the feasible region of the continuous relaxation of (LHR_k) there exists a $i \in D_k$ such that $\bar{y}_{ki} = 1$, and $\bar{y}_{ki'} = 0, \forall i' \in D_k, i' \neq i$. Therefore, for every $k \in K$ the vertices of their corresponding problem have $\{0, 1\}$ values for y_{ki} . Since there are no constraints linking the variables of the feasible region of each of the $|K| + 1$ problems, then all of the vertices of the feasible region of the continuous relaxation of (LHR_λ) have $\{0, 1\}$ values for y_{ki} . ■

Note that Property 3.3 indicates that every solution of (LHR_λ) yields 0-1 values for the binary variables. One of the main advantages of this property, together with 3.2, is that (LHR_λ) can be solved by solving $|K| + 1$ small LPs. The solution of (LHR_λ) can be used as a heuristic for finding “good” feasible solutions for (GDP). On the downside, this property indicates that the best possible bound that can be obtained from (LHR_λ) is the same as the optimal value of the continuous relaxation of (HR)[1].

3.2. Including logic relations in (LHR_λ)

In cases in which there are logic relations between the Boolean variables of the GDP, (LHR_λ) needs to be modified to preserve Properties 3.3 and 3.2. This can be achieved by either introducing additional variables or including the propositional logic in the solution method[18]. Using the propositional logic in the solution method enables a large set of tools that combine logic-based methods with optimization. The work by Hooker[19] has established important theories and results in this field. However, the scope of this work is to establish the properties of a general Lagrangean relaxation of (HR), as well as the basis for using this relaxation as primal heuristic. For this reason, we will present in this section a modified version of (LHR_λ) by introducing additional variables, and leave the integration of (LHR_λ) with logic-based methods for future work.

Consider the following problem, that is equivalent to (GDP):

$$\begin{aligned}
& \min c^T x \\
& s.t. \quad Gx \leq g \\
& \quad \bigvee_{i \in D_k} \begin{bmatrix} Y_{ki} \\ \bar{y}_{ki} = 1 \\ A^{ki}x \leq a^{ki} \end{bmatrix} \quad k \in K \\
& \quad \bigvee_{i \in D_k} Y_{ki} \quad k \in K \\
& \quad \sum_{i \in D_k} \bar{y}_{ki} = 1 \quad k \in K \\
& \quad H\bar{y} \geq h \\
& \quad x^{lo} \leq x \leq x^{up} \\
& \quad 0 \leq \bar{y}_{ki} \leq 1 \quad k \in K, i \in D_k \\
& \quad x \in \mathbb{R}^n \\
& \quad \bar{y}_{ki} \in \mathbb{R} \quad k \in K, i \in D_k \\
& \quad Y_{ki} \in \{True, False\} \quad k \in K, i \in D_k
\end{aligned} \tag{1}$$

Note that in (1), a continuous variable ($0 \leq \bar{y}_{ki} \leq 1$) is introduced. The linear constraints that represent the logic relations are expressed for \bar{y} . It is easy to see that (1) is equivalent to (GDP), in the sense that $Y_{ki} = True$ implies $\bar{y}_{ki} = 1$, and $Y_{ki} = False$ implies $\bar{y}_{ki} = 0$. Note that (1) also has the structure required: there are no logic relations between Boolean variables, and the global constraints and disjunctive terms involve only continuous variables. It is possible to perform the (HR) reformulation of (1). After few algebraic substitutions, the HR of (1) is as follows:

$$\begin{aligned}
& \min c^T x \\
& s.t. \quad Gx \leq g \\
& \quad y = \bar{y} \\
& \quad x = \sum_{i \in D_k} \nu^{ki} \quad k \in K \\
& \quad A^{ki} \nu^{ki} \leq a^{ki} y_{ki} \quad k \in K, i \in D_k \\
& \quad \sum_{i \in D_k} y_{ki} = 1 \quad k \in K \\
& \quad H\bar{y} \geq h \\
& \quad x^{lo} y_{ki} \leq \nu^{ki} \leq x^{up} y_{ki} \quad k \in K, i \in D_k \\
& \quad x \in \mathbb{R}^n \\
& \quad \bar{y}_{ki} \in \mathbb{R} \quad k \in K, i \in D_k \\
& \quad y_{ki} \in \{0, 1\} \quad k \in K, i \in D_k
\end{aligned} \tag{2}$$

For Lagrangean multipliers λ^1, λ^2 , the following Lagrangean relaxation of (2) is obtained:

$$\begin{aligned}
& \min c^T x + \sum_{k \in K} \left(\sum_{j \in J_k} \lambda_{kj}^1 (x_j - \sum_{i \in D_k} \nu^{ki}) + \sum_{i \in D_k} \lambda_{ki}^2 (y_{ki} - \bar{y}_{ki}) \right) \\
& s.t. \quad Gx \leq g \\
& \quad A^{ki} \nu^{ki} \leq a^{ki} y_{ki} \quad k \in K, i \in D_k \\
& \quad \sum_{i \in D_k} y_{ki} = 1 \quad k \in K \\
& \quad H\bar{y} \geq h \\
& \quad x^{lo} y_{ki} \leq \nu^{ki} \leq x^{up} y_{ki} \quad k \in K, i \in D_k \\
& \quad x \in \mathbb{R}^n \\
& \quad \bar{y}_{ki} \in \mathbb{R} \quad k \in K, i \in D_k \\
& \quad y_{ki} \in \{0, 1\} \quad k \in K, i \in D_k
\end{aligned} \tag{3}$$

which decomposes into $|K| + 1$ subproblems (where $H\bar{y} \geq h$ is a global constraint and appears in (LHR_0)).

195 In some cases, the solution of (LHR_λ) including the logic constraints for the original variables might still yield 0-1 values to the binary variables (e.g. it can still be solved as an LP). However, it might not be possible to decompose (LHR_λ) in smaller LPs (depending on the structure of the logic relations). Different methods could make use of problem (LHR_λ) to solve (GDP). For example, it can be used as heuristic in the search of feasible solutions.

200 4. Lagrangean relaxation as primal heuristic in the disjunctive branch and bound

The disjunctive branch and bound presented in Section 2.3 can be adapted to incorporate (LHR_λ) as primal heuristic. Before presenting the algorithm, consider the LP subproblem (SP) that results when the value of the Boolean variables (or binary variables in the MILP reformulation) are fixed. Given \hat{y} such that, for every $k \in K$ there is only one $i \in D_k$ for which $\hat{y}_{ki} = 1$ and $\hat{y}_{ki'} = 0, \forall i' \in D_k, i' \neq i$, the following (SP) is obtained:

$$\begin{aligned}
& \min c^T x \\
& s.t. \quad Gx \leq g \\
& \quad A^{ki} x \leq a^{ki} \quad \forall \hat{y}_{ki} = 1 \\
& \quad x^{lo} \leq x \leq x^{up} \\
& \quad x \in \mathbb{R}^n
\end{aligned} \tag{SP}$$

In (SP), the constraints corresponding to active disjunctive terms $\hat{y}_{ki} = 1$ are enforced while constraints corresponding to non-active terms $\hat{y}_{ki'} = 0$ are removed.

The modified algorithm is as follows:

0. *Initialize.* Set $L = N_0, z^{up} = \infty, (x^*, y^*) = \emptyset$. Initialize $\lambda_{kj}^0 = 0$.

205 1. *Terminate.* If $L = \emptyset$, then (x^*, y^*) is optimal and algorithm terminates.

2. *Node selection.* Choose a node $N_p \in L$, and delete it from L . Go either to 3a or to 3b.

3a. *Bound.* Solve the (R-HR) of GDP_p corresponding to N_p . If it is infeasible, go to step 1. Else, let z^p be its objective function and (x^p, y^p) its solution. Set λ_{kj}^p to be the Lagrangean multipliers corresponding to the constraints $x_j = \sum_{i \in D_k} \nu^{ki}$, $k \in K; j \in J_k$.

210 3b. *Bound.* Solve the (R-BM) of GDP_p corresponding to N_p . If it is infeasible, go to step 1. Else, let z^p be its objective function and (x^p, y^p) its solution.

4. *Prune.* If $z^p \geq z^{up}$, go to step 1.

If $y^p \in \mathbb{Z}^q$ let $z^{up} = z^p$ and $(x^*, y^*) = (x^p, y^p)$. Delete all nodes $N_r \in L$ in which $z^r \geq z^{up}$, and go to step 1. Else, go to step 5 or 6.

215 5. *Primal heuristic (optional).* Solve (LHR_λ) of GDP_p with λ_{kj}^p . Let \hat{z}^p be its objective function and \hat{y} the value of the integer variables. Let $z^p = \max\{z^p, \hat{z}^p\}$.

Solve (SP) with fixed \hat{y} and, if it is feasible, let \bar{z}^p be its objective function and (\bar{x}^p) its solution. If $\bar{z}^p \leq z^{up}$ let $z^{up} = \bar{z}^p$ and $(x^*, y^*) = (\bar{x}^p, \hat{y}^p)$. Delete all nodes $N_r \in L$ in which $z^r \geq z^{up}$. If $z^p = \bar{z}^p$, go to step 1. Else, go to step 6.

220 6. *Branch.* Select a disjunction $k \in K$ such that $y_{ki} \notin \{0, 1\}$ for some $i \in D_k$. For every $i \in D_k$, construct the corresponding GDP (GDP_p^i) by setting the constraints corresponding to the disjunctive term i as global, and removing the Boolean variables and constraints corresponding to term $i' \neq i; i' \in D_k$. Add $|D_k|$ new nodes, corresponding to GDP_p^i , to L . For every one of the new nodes, let the corresponding Lagrange multipliers $(\lambda_{kj}^{p,i})$ be λ_{kj}^p . Go to step 1.

225 There are two main differences in the proposed disjunctive branch and bound with respect to the one presented in Section 2.3. The first in Step 3a and the second is the inclusion of the new Step 5.

The first difference is that when Step 3a is selected, the Lagrangean multipliers are updated. Note that this implies that for the particular node in which the (R-HR) was solved, (LHR_λ) is in fact the Lagrangean dual of the HR of GDP_p . It is important (while not required) to perform Step 3a at the root node, so the

230 Lagrangean multipliers are initialized with the Lagrangean dual of the HR of the original GDP. In subsequent nodes, it may or may not be useful to solve the larger HR reformulation in order to obtain better lower bounds and updated Lagrangean multipliers.

The second difference is Step 5, which is optional. In step 5, (LHR_λ) is solved to obtain integer solutions. The discrete solution is fixed and a small LP is solved (which corresponds to the original GDP with fixed

235 decisions). If the LP is feasible, it provides a valid upper bound and feasible solution. Note that if Step 3a was selected (LHR_λ) is in fact the Lagrangean dual. Otherwise, it is a Lagrangean relaxation that uses the Lagrangean multipliers inherited from the parent node. Also note that if step 3b was selected, it is possible that $\hat{z}^p \geq z^p$ (e.g. the Lagrangean relaxation of the HR provides better bound than the continuous relaxation

of the BM). For this reason, Step 5 sets $z^p = \max\{z^p, \hat{z}^p\}$.

240

Same as the disjunctive branch and bound presented in Section 2.3, this algorithm converges in a finite number of iterations. In the worst case, evaluating every single resulting node of the search tree.

5. Illustrative example

In this section, we present a simple example and the application of the proposed disjunctive branch and bound to solve it. Consider the following analytical example:

$$\begin{aligned}
& \min 7x_1 - 2x_2 \\
& s.t. \\
& \left[\begin{array}{c} Y_{11} \\ 0.9487x_1 + 0.3162x_2 \leq 11.3842 \\ -0.5145x_1 - 0.8575x_2 \leq -10.2899 \\ x_2 \leq 9 \end{array} \right] \vee \left[\begin{array}{c} Y_{12} \\ 0.9615x_1 - 0.2747x_2 \leq 5.7691 \\ -0.6247x_1 + 0.7809x_2 \leq 0.4685 \\ -0.7071x_1 - 0.7071x_2 \leq -4.2426 \end{array} \right] \\
& \vee \left[\begin{array}{c} Y_{13} \\ 0.8944x_1 + 0.4472x_2 \leq 6.2610 \\ -0.9864x_1 + 0.1644x_2 \leq -0.3288 \\ 0.5547x_1 - 0.8321x_2 \leq -2.7735 \end{array} \right] \\
& \left[\begin{array}{c} Y_{21} \\ 0.9806x_1 - 0.1961x_2 \leq 2.1573 \\ -0.6x_1 + 0.8x_2 \leq 4.8 \\ -0.5547x_1 - 0.8321x_2 \leq -4.9923 \end{array} \right] \vee \left[\begin{array}{c} Y_{22} \\ 0.9806x_1 + 0.1961x_2 \leq 7.2563 \\ -0.9487x_1 + 0.3162x_2 \leq -3.4785 \\ 0.3162x_1 - 0.9487x_2 \leq 0.3162 \end{array} \right] \\
& \vee \left[\begin{array}{c} Y_{23} \\ x_1 \leq 10 \\ -0.3162x_1 + 0.9487x_2 \leq 6.3246 \\ -0.5547x_1 - 0.8321x_2 \leq -11.3714 \end{array} \right] \\
& \left[\begin{array}{c} Y_{31} \\ x_1 \leq 6 \\ -0.3714x_1 - 0.9285x_2 \leq -3.1568 \\ -x_1 \leq -1 \\ x_2 \leq 6 \end{array} \right] \vee \left[\begin{array}{c} Y_{32} \\ 0.7071x_1 + 0.7071x_2 \leq 10.6066 \\ -0.3162x_1 - 0.9487x_2 \leq -7.9057 \\ -0.4472x_1 + 0.8944x_2 \leq 6.7082 \end{array} \right] \\
& \vee_{i \in D_k} Y_{ki}; \quad k \in \{1, 2, 3\} \\
& 0 \leq x_1, x_2 \leq 10 \\
& Y_{ki} \in \{True, False\}; \quad k \in \{1, 2, 3\}, i \in D_k
\end{aligned} \tag{4}$$

The feasible region of problem (4) is shown in Figure 1. Figure 1a shows the feasible region of each disjunction, projected onto the space of the continuous variables. Figure 1b shows the optimal solution to problem (4) in the projection onto the continuous variables.

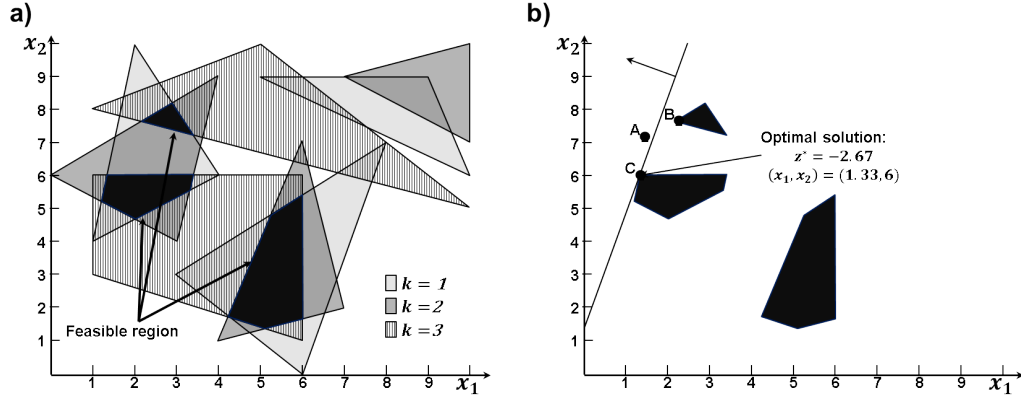


Figure 1: Illustration of the feasible region of: a) the feasible region projected onto x_1 and x_2 , and b) the optimal solution.

The application of the proposed disjunctive branch and bound to (4) is as follows:

0. *Initialize.* Set $L = N_0$, $z^{up} = \infty$, $(x^*, y^*) = \emptyset$. Initialize $\lambda_{kj}^0 = 0$.

1. *Terminate.* $L = N_0$, go to Step 2.

2. *Node selection.* Choose node N_0 and delete it from L . Go to 3a.

3a. *Bound.* The solution of the (R-HR) of GDP_0 (which corresponds to N_0) yields the following values:

Continuous variables and objective function $(x_1^0, x_2^0, z^0) = (1.5, 7.1, -3.6)$ (Point A in Figure 1b).

Binary variables $y_{13}^0 = y_{21}^0 = 1$, $y_{11}^0 = y_{12}^0 = y_{22}^0 = y_{23}^0 = 0$, and $y_{31}^0 = 0.43$, $y_{32}^0 = 0.57$

Lagrangian multipliers $\lambda_{kj}^0 = (-6.286, 1.048, -0.714, 0.952, 0, 0)$ (i.e. Lagrangian multipliers corresponding to the constraints $x_j = \sum_{i \in D_k} \nu^{ki}$; $k \in K$; $j \in J_k$ in (R-HR)).

4. *Prune.* Since $z^0 < z^{up}$ and $y^0 \notin \mathbb{Z}^q$, go to step 5.

5. *Primal heuristic (optional).* The solution of (LHR_λ) of GDP_0 with λ_{kj}^0 yields: $\hat{z}^0 = -3.6$, $\hat{y}_{13}^0 = \hat{y}_{21}^0 = \hat{y}_{32}^0 = 1$ (Point B in Figure 1b).

The solution (SP) with fixed \hat{y}^0 is feasible and it yields the following values:

Continuous variables and objective function $(\bar{x}_1^0, \bar{x}_2^0, \bar{z}^0) = (2.15, 7.62, -0.154)$.

Note that this solution indicates that even at the root node, it is possible to obtain a feasible solution. In this case, we update the best known solution: $z^{up} = -0.154$, $(x^*, y^*) = (\bar{x}^0, \hat{y}^0)$

6. *Branch.* Select $k = 3$ for branching. Since $|D_3| = 2$, two new nodes are created: $LDGP_0^1, LDGP_0^2$.

$LDGP_0^1$ is as follows:

$$\begin{aligned}
& \min 7x_1 - 2x_2 \\
& s.t. \quad x_1 \leq 6 \\
& \quad -0.3714x_1 - 0.9285x_2 \leq -3.1568 \\
& \quad -x_1 \leq -1 \\
& \quad x_2 \leq 6 \\
& \quad \left[\begin{array}{c} Y_{11} \\ 0.9487x_1 + 0.3162x_2 \leq 11.3842 \\ -0.5145x_1 - 0.8575x_2 \leq -10.2899 \\ x_2 \leq 9 \end{array} \right] \vee \left[\begin{array}{c} Y_{12} \\ 0.9615x_1 - 0.2747x_2 \leq 5.7691 \\ -0.6247x_1 + 0.7809x_2 \leq 0.4685 \\ -0.7071x_1 - 0.7071x_2 \leq -4.2426 \end{array} \right] \\
& \quad \vee \left[\begin{array}{c} Y_{13} \\ 0.8944x_1 + 0.4472x_2 \leq 6.2610 \\ -0.9864x_1 + 0.1644x_2 \leq -0.3288 \\ 0.5547x_1 - 0.8321x_2 \leq -2.7735 \end{array} \right] \\
& \quad \left[\begin{array}{c} Y_{21} \\ 0.9806x_1 - 0.1961x_2 \leq 2.1573 \\ -0.6x_1 + 0.8x_2 \leq 4.8 \\ -0.5547x_1 - 0.8321x_2 \leq -4.9923 \end{array} \right] \vee \left[\begin{array}{c} Y_{22} \\ 0.9806x_1 + 0.1961x_2 \leq 7.2563 \\ -0.9487x_1 + 0.3162x_2 \leq -3.4785 \\ 0.3162x_1 - 0.9487x_2 \leq 0.3162 \end{array} \right] \\
& \quad \vee \left[\begin{array}{c} Y_{23} \\ x_1 \leq 10 \\ -0.3162x_1 + 0.9487x_2 \leq 6.3246 \\ -0.5547x_1 - 0.8321x_2 \leq -11.3714 \end{array} \right] \\
& \quad \vee_{i \in D_k} Y_{ki}; \quad k \in \{1, 2, 3\} \\
& \quad 0 \leq x_1, x_2 \leq 10 \\
& \quad Y_{ki} \in \{True, False\}; \quad k \in \{1, 2, 3\}, i \in D_k
\end{aligned} \tag{5}$$

Note that in (5), the constraints corresponding to $Y_{31} = True$ are included as global constraints, while constraints corresponding to $Y_{32} = True$ are removed from the formulation. $LDGP_0^2$ can be constructed in the same manner, but including as global constraints the ones corresponding to $Y_{32} = True$ while removing the constraints corresponding to $Y_{31} = True$. For simplicity, $p = 1$ is assigned to $LDGP_0^1$, and $p = 2$ to $LDGP_0^2$. Two new nodes are added L : $L = \{N_1, N_2\}$ (corresponding to $p = 1, 2$). The Lagrangean multipliers of the new nodes are initialized using λ_{kj}^0 from the parent node: $\lambda_{kj}^1 = \lambda_{kj}^2 = (-6.286, 1.048, -0.714, 0.952, 0, 0)$.

In the second iteration, the following results are obtained for the node N_1 using step 3a:

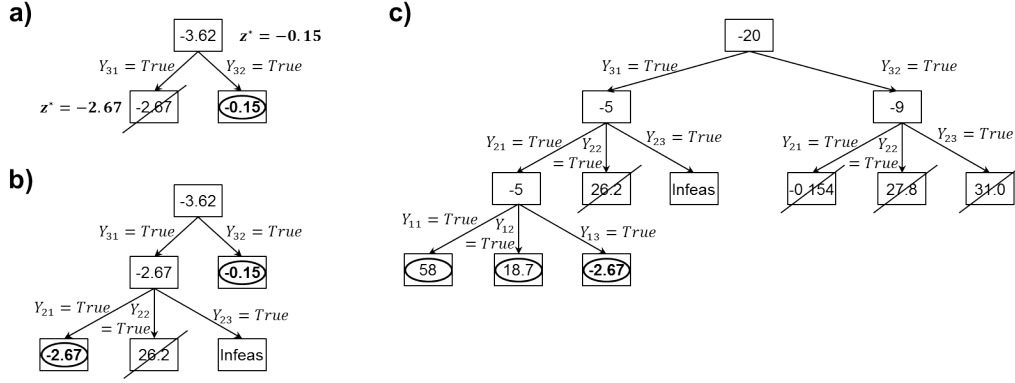


Figure 2: Disjunctive branch and bound tree for: a) the proposed algorithm, b) HR, and c) BM.

(R-HR): $(z^1, x_1^1, x_2^1, y_{11}^1, y_{12}^1, y_{13}^1, y_{21}^1, y_{22}^1, y_{23}^1) = (-2.67, 1.33, 6, 0, 0, 1, 0.852, 0.025, 0.123)$

(LHR $_{\lambda}$): $(\hat{z}^1, \hat{y}_{11}^1, \hat{y}_{12}^1, \hat{y}_{13}^1, \hat{y}_{21}^1, \hat{y}_{22}^1, \hat{y}_{23}^1) = (-2.67, 1.33, 6, 0, 0, 1, 1, 0, 0)$

(SP): $(\bar{z}^1, \bar{x}_1^1, \bar{x}_2^1) = (-2.67, 1.33, 6)$ (Point C in Figure 1b).

Note that the solution obtained by (R-HR) is optimal in the continuous variables. However, it did not yield 0-1 values to $(y_{21}^1, y_{22}^1, y_{23}^1)$. After solving (LHR $_{\lambda}$), the best known solution is updated: $z^{up} = -2.67$,

$(x^*, y^*) = (\bar{x}^0, \hat{y}^0)$. The node can be pruned since $z^1 = z^{up}$.

Node N_2 yields an integer solution with $z^2 = -0.154$ and it can be pruned. After evaluating N_1, N_2 , all the nodes are pruned ($L = \emptyset$) and the algorithm terminates.

Figure 2 shows the disjunctive branch and bound tree for different versions of the algorithm for problem (4). Figure 2a shows the tree for the proposed algorithm, as described earlier (using step 3a in each of the nodes). Figure 2b presents the tree of the HR disjunctive branch and bound (i.e. the algorithm presented in Section 2.3 using Step 3a at every node). Finally, Figure 2c presents the tree of the BM disjunctive branch and bound (i.e. using Step 3b at every node).

It is easy to see from Figure 2 that the proposed algorithm requires fewer number of nodes than the other two algorithms (3 in the proposed algorithm, 6 in the HR, and 12 in the BM). Nevertheless, it requires the evaluation of 3 LPs in N_0 and in N_1 , while the BM and HR disjunctive branch and bound require the evaluation of one LP at every node. In this simple example, the solution of every LP takes a fraction of a second. However, in larger problems the difference in solution time among (R-HR), (LHR $_{\lambda}$), (R-BM), and (SP) can be very significant. For example, in instance 11 of Section 6.1 the solution times (using CPLEX 12.6.1[20]) of (R-HR), (LHR $_{\lambda}$), (R-BM), and (SP) at the root node are 21.5s, 0.7s, 0.2s, 0.1s, respectively. Note that the solution time of (LHR $_{\lambda}$) is about 30 times faster than the solution of (R-HR) ((LHR $_{\lambda}$) is the Lagrangean dual of (R-HR), so they provide the same lower bound). Also, (LHR $_{\lambda}$) was evaluated as a single LP and a single core, so the difference in solution time comes from CPLEX exploiting the structure of (LHR $_{\lambda}$). If (LHR $_{\lambda}$) is solved by solving the smaller LPs in parallel (Property 3.2), the solution time can be

further reduced.

6. Results

In this section, we present the performance of the proposed disjunctive branch and bound against the simple BM and HR disjunctive branch and bound. We also compare against a fourth disjunctive branch and bound with a “random heuristic”. This algorithm is exactly the same as the proposed algorithm, but the primal heuristic is random (i.e. instead of solving (LHR_λ) and fixing \hat{y} at its solution, \hat{y} is fixed randomly). This comparison is important to show that (LHR_λ) provides a good heuristic for finding feasible solutions, and it is not only the additional work at each node what drives the improvement in the disjunctive branch and bound. The M-parameters of the Big-M reformulation were obtained by using the variable bounds (i.e. for a constraint of the type $ax \leq b$ the parameter $M = \sum_{j:a_j \geq 0} x^{up} - \sum_{j:a_j < 0} x^{lo} - b$).

The results were obtained using an Intel(R) Core(TM) i7 CPU 2.93 GHz and 4 GB of RAM. The algorithm was implemented in GAMS 24.4.5[21] using CPLEX 12.6.1[20]. The proposed algorithm uses step 3b at every node, except for the root node (i.e. it solves (R-HR) at the root node, initializes the Lagrangean multipliers, then solves (R-BM) at every other node without modifying the multipliers). (LHR_λ) is solved as a single LP by CPLEX, but computational experience shows that CPLEX can exploit the decomposable structure of (LHR_λ) (as described in Section 5). The algorithms use a breadth first branching strategy, and at every node selects the disjunction with fewest disjunctive terms that yielded non integer variables for branching.

The solution times presented in this section refer to the solve time (i.e. it includes the time to generate the model at every node, but ignores the time to create each node and to decide on branching and pruning). While in most cases the wall time is very close to the solve time, in some instances it is not. The reason for comparing the solve time (vs. wall time) is to consider only the time spent in solving the problem, and ignore inefficiencies in the code (that could be reduced in a lower level programming language or improvements in the branch and bound code).

Note that this implementation of the disjunctive branch and bound algorithm is a prototype, and its purpose is to show the improvement of the basic algorithm when using (LHR_λ) as a primal heuristic. The algorithm is implemented in GAMS, and it has to generate an MILP model at every node of the search tree. Also, it does not include presolve, heuristics or cuts, so it is much slower than CPLEX. Future work includes improvements in the implementation of the algorithm.

The algorithms were tested with 100 instances of 3 problems (i.e. a total of 300 instances). The solution time performance curve over all instances is presented in Figure 3. Figure 3 shows the percentage of instance solved vs. time. The figure shows four algorithms: “HR” refers to the HR disjunctive branch and bound, “BM” to the BM one, “ALG” to the proposed modified algorithm, and “RAN” to the algorithm with a random

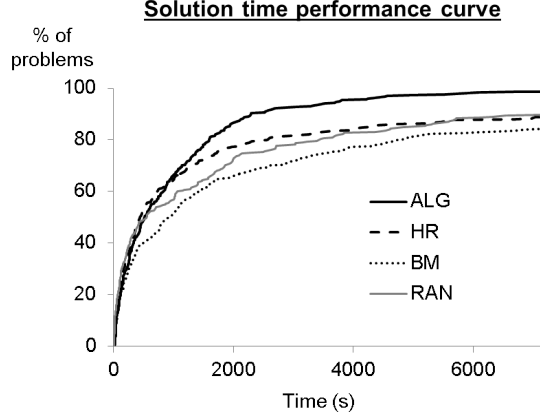


Figure 3: Solution time performance curve for the 300 tested instances.

heuristic. It can be observed from the figure that the proposed algorithm is considerably better than the rest. The HR branch and bound performs second, but the random heuristic algorithm performs similarly in larger instances. The worst performer is the BM disjunctive branch and bound. Out of the 300 instances, the BM, HR, ALG, and RAN solve 253, 266, 296, and 269 instances, respectively.

While Figure 3 presents the performance over all instances, the performance varies in the different examples. In the remaining of this section, we present more details and results for each example. We first present random instances for unstructured linear GDP problems. We then present results of two particular GDP problems: the strip packing and the contracts problem. The strip packing problem does not include logic constraints, while the contracts problem does. 100 instances are tested for each of these problems.

6.1. Unstructured GDP problems

For this example, 100 random instances are generated for the unstructured GDP problems without logic relations or global constraints (note that any GDP with global constraints can always be reformulated as a GDP without global constraints by using improper basic steps[13]). The coefficients of a^{ki} range between -1.00 and 1.00 . The coefficient range of c is $[-10.0, 10.0]$; of x^{lo} is $[-100, -10]$; and of x^{up} is $[10, 100]$. In order to avoid very high density in the matrices A^{ki} , and to give the problem structure (e.g. to relate more some variables to some disjunctions), the following formula was used to calculate A^{ki} ; $k \in K, i \in D_k$ (where $e \in E_{ki}$ are the constraints in ki and $j \in J_k$ are the variables in $k \in K$):

$$A_{ej}^{ki} = A0_{ej}^{ki} \alpha_{ej}^{ki} \beta_{ej}^{ki}$$

where,

$A0_{ej}^{ki}$ is a random parameter between -1.00 and 1.00 .

α_{ej}^{ki} is the probability of a variable appearing in an equation. (in all tested instances $\alpha_{ej}^{ki} = 0.5$).

$\beta_{ej}^{ki} = \text{round}(\text{rand-between}(0, 1 - |(j - k * |J|/|K|)/|J|))$.

Table 1: Average statistics for the different random problems

Problem	# instances	E_{max}	0-1 vars.	Variables		Constraints	
				BM	HR	BM	HR
k5-i40-v40	10	50	106	147	3,511	3,064	9,948
k5-i40-v50	10	50	113	164	4,650	3,344	12,515
k6-i25-v40	10	50	82	123	2,723	2,361	7,750
k6-i25-v50	10	50	86	137	3,523	2,523	9,533
k7-i15-v40	10	40	61	102	2,032	1,442	5,524
k7-i15-v50	10	40	63	114	2,567	1,490	6,674
k8-i5-v40	10	30	28	69	942	470	2,466
k8-i5-v50	10	30	29	80	1,202	500	3,061
k10-i3-v50	10	20	25	76	1,053	304	2,648
k10-i3-v80	10	20	25	106	1,655	304	4,022

To illustrate the idea behind the generation of A^{ki} , consider a problem with 40 variables, 10 disjunctions, 20 disjunctive terms per disjunction, and a maximum of 30 constraints per disjunctive term. In disjunction $k = 1$, the probability of variable $j = 1$ appearing in any constraint of any disjunctive term is $\alpha_{e,1}^{1,i} \beta_{e,1}^{1,i}$. Since $\beta_{e,1}^{1,i} = \text{round}(\text{rand-between}(0, 0.925))$, the probability is $\alpha_{e,1}^{1,i} \beta_{e,1}^{1,i} = (0.5)(0.46) = 23\%$. Note that, since there are up to 30 constraints in each of the 20 disjunctive terms of disjunction $k = 1$, there is a very high probability that this variable appears in a constraint of $k = 1$. For $k = 1$ and $j = 20$, $\beta_{e,20}^{1,i} = \text{round}(\text{rand-between}(0, 0.6))$ so the probability of variable $j = 20$ appearing in any constraint of any disjunctive term of $k = 1$ is 8%. The probability of variable $j > 25$ appearing in any constraint of any disjunctive term of $k = 1$ is 0%. The probability of variable $j = 20$ appearing in any constraint of any disjunctive term of $k = 5$ is 25%.

Table 1 presents the statistics for the instances that were generated randomly. All infeasible instances were removed from the test set, and all infeasible disjunctive terms were removed from the feasible instances. The number of constraints in a disjunctive term ($|E_{ki}|$) is random between $E_{max}/5$ and E_{max} . The problem identifier indicates the number of disjunctions $|K|$, disjunctive terms D_k , and number of continuous variables in the GDP. The table shows the average number of binary variables, as well as the average problem size for the BM and HR. It can be observed from Table 1 that in some of these instances the difference in problem size between the BM and HR is considerable, particularly in the number of variables.

Figure 4 shows the performance of the different algorithms for the 100 random instances. Figure 4a shows that the solution time performance of the proposed and the random algorithms is much better than the

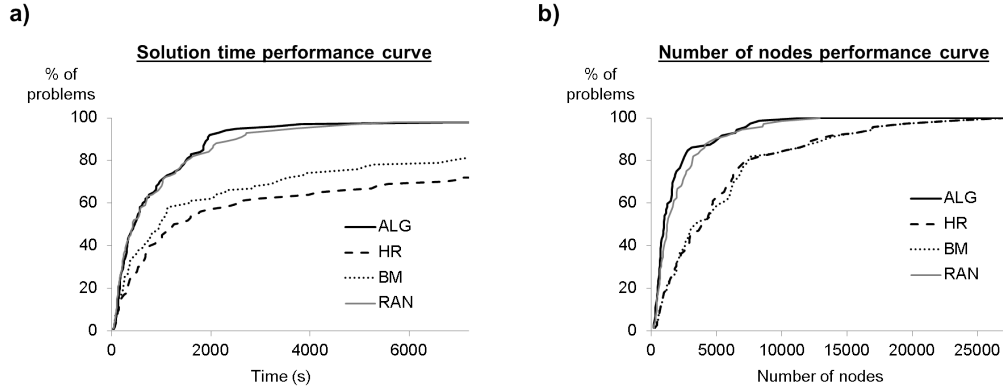


Figure 4: Performance curves for the different algorithm for: a) solution time, and b) number of nodes for the 72 instances that all algorithms solve.

traditional BM and HR disjunctive branch and bound for this example. The figure shows that the proposed and random algorithms solve 98 of the instances, while the BM and HR versions can only solve 81 and 72, respectively. Note that in this example, the BM disjunctive branch and bound performs better than the HR. This is because the number of nodes required in the BM and HR disjunctive branch and bound is very similar as can be observed in Figure 4b. Figure 4b shows the required number of nodes to solve the 72 instances that all algorithms solved. The figure shows that the proposed algorithm requires the fewest number of nodes. The BM and HR, however, require very similar number of nodes. Considering that the HR generates a much larger MILP (as shown in Table 1), it is expected that the solution time of the BM disjunctive branch and bound is faster than that of the HR.

Figure 5 shows the number of nodes required to find the optimal solution and the first feasible solution for the 72 instances that all algorithms solve. While the figure does not show the time to prove optimality, finding a feasible and the optimal solution is an important consideration in practice. It can be observed from the figure that the proposed algorithm is better at finding the optimal solution (Figure 5a) and the first feasible solution (Figure 5b). The random heuristic algorithm is very good in this example, requiring a similar number of nodes to find the first and optimal solution as the proposed method. However, the BM and HR versions require many more nodes to find a feasible solution and the optimal solution. The figure shows that there is small difference in the HR and BM disjunctive branch and bound with respect to the number of nodes required to find first and optimal solutions. The small difference seems to indicate that the HR requires fewer nodes to find the optimal solution, while the BM requires fewer to find the first feasible solution.

Note that the proposed algorithm performs better than the other versions of the disjunctive branch and bound for unstructured GDP instances. In the remaining of this section, we present results for two particular GDP problems: strip packing and contracts.

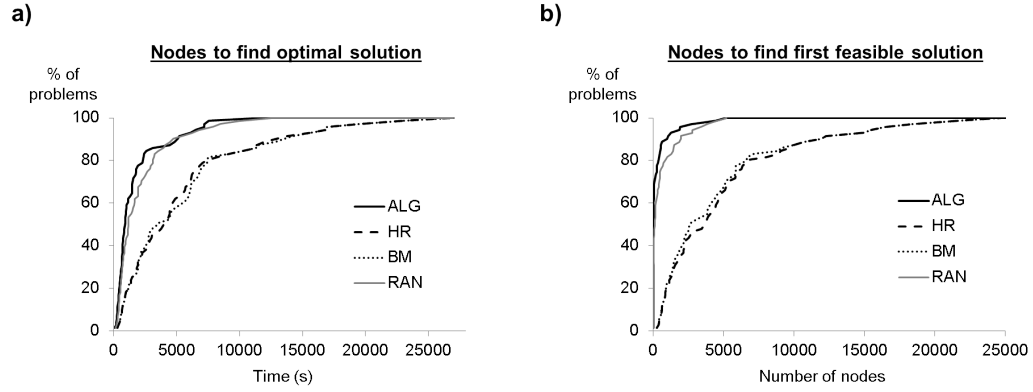


Figure 5: Number of nodes required to find: a) the optimal solution, and b) the first feasible solution for the 72 instances that all algorithms solve.

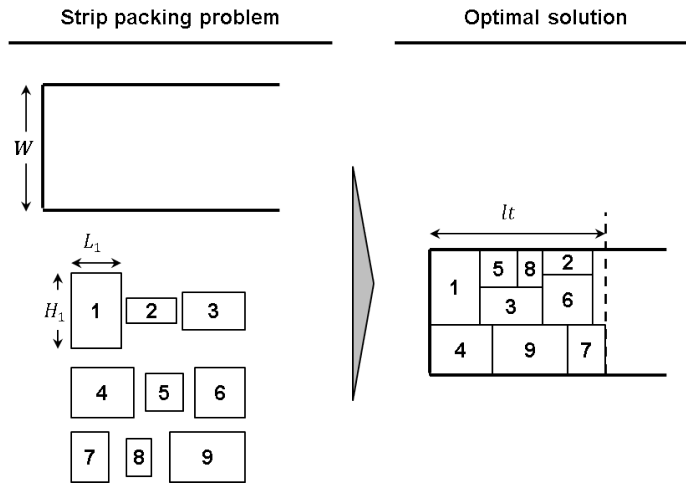


Figure 6: Illustration of the strip packing problem.

390 6.2. Strip packing problem

Given set of N rectangles, the strip packing problem consists on placing them on a strip while minimizing the its length. The rectangles cannot overlap or be rotated. The height and length of each rectangle is known ($H_i, L_i; i \in N$), and the strip has width W [22]. Figure 6 illustrates the strip packing problem.

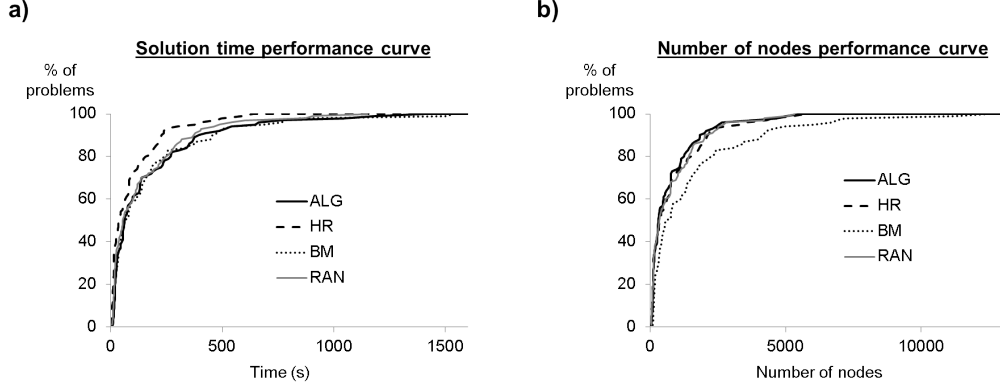


Figure 7: Performance curves for the different algorithm for: a) solution time, and b) number of nodes.

The GDP formulation of this problem is as follows[22, 23]:

$$\begin{aligned}
 & \min lt \\
 & s.t. \quad lt \geq x_i + L_i \quad i \in N \\
 & \quad \left[\begin{array}{c} Z_{ij}^1 \\ x_i + L_i \leq x_j \end{array} \right] \vee \left[\begin{array}{c} Z_{ji}^1 \\ x_j + L_j \leq x_i \end{array} \right] \\
 & \quad \vee \left[\begin{array}{c} Z_{ij}^2 \\ y_i - H_i \geq y_j \end{array} \right] \vee \left[\begin{array}{c} Z_{ji}^2 \\ y_j - H_j \geq y_i \end{array} \right] \quad i, j \in N, i < j \\
 & \quad Z_{ij}^1 \vee Z_{ji}^1 \vee Z_{ij}^2 \vee Z_{ji}^2 \quad i, j \in N, i < j \\
 & \quad 0 \leq x_i \leq UB - L_i \quad i \in N \\
 & \quad H_i \leq y_i \leq W \quad i \in N \\
 & \quad Z_{ij}^1, Z_{ij}^2 \in \{True, False\} \quad i, j \in N, i \neq j
 \end{aligned} \tag{6}$$

In (6), the objective is to minimize the length lt . The coordinates of the upper-left corner of rectangle i are represented with the variables (x_i, y_i) . The global constraints $(lt \geq x_i + L_i)$ enforce length of the strip corresponds to the largest $x_i + L_i$ (i.e. the coordinate of the top-left corner plus the length of the rectangle). There is a disjunction for every pair of rectangles $i, j \in N, i < j$. Each of the terms of the disjunction represents the relative position of rectangle i with respect to rectangle j . The first term, corresponding to $Z_{ij}^1 = True$, represents rectangle i to the left of rectangle j . $Z_{ji}^1 = True$ represents rectangle i to the right of rectangle j . $Z_{ij}^2 = True$ represents rectangle i on top of rectangle j . Finally, $Z_{ji}^2 = True$ represents rectangle i below rectangle j . The parameter UB is an upper bound for the strip (e.g. $UB = \sum_i L_i$).

The different algorithms were tested on 100 random instances of the strip packing problem. The range of values of the random parameters is as follows: $N = 4, 5$; $W = 5-7$; $L_i = 1-10$; $H_i = 2-5$.

Figure 7a shows that the HR disjunctive branch and bound performs better than the other three for this problem. The BM, random heuristic, and the proposed branch and bound perform similarly in terms of

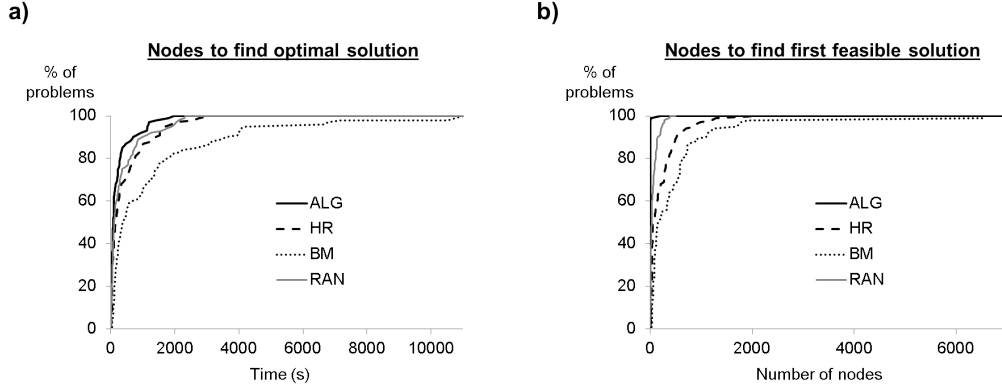


Figure 8: Number of nodes required to find: a) the optimal solution, and b) the first feasible solution.

solution time. It is interesting to note that the proposed algorithm performs the third in terms of solution time. However, Figure 7b shows that the proposed algorithm requires the fewest number of nodes. Because some of the nodes require the evaluation of more than one LP, the fewer number of nodes is not reflected in the solution time for this problem. As expected, the random heuristic and the HR disjunctive branch and bound require fewer nodes than the BM disjunctive branch and bound. The random heuristic algorithm requires similar number of nodes as the HR. However, it requires the evaluation of two LPs in many of the nodes, so the performance in terms of the solution time is worse (as shown in Figure 7a). The heuristic that uses the Lagrangean relaxation requires fewer nodes than the random heuristic, which shows that the former is a better heuristic than randomly fixing variables. However, the improvement is small in this example, and hence the time spent in solving the Lagrangean relaxation at every node increases the total solution time. Note that for this problem the performance curves of all algorithms are not very different.

Figure 8 shows the performance curve of the number of nodes required to find the optimal solution and to find a feasible solution. Figure 8a shows that the number of nodes required to find an optimal solution is smaller for the proposed algorithm than for the others. Figure 8b shows that the first feasible solution can be almost always be found at the root node in the proposed and random algorithms. In particular, the proposed algorithm finds the first feasible solution at the root node in 94 instances; and in 4 instances it finds it at the first node. The improvement in finding optimal and feasible solutions can be better appreciated in Table 2. Table 2 shows an average of the metrics presented in Figures 7 and 8. Note that both algorithms with primal heuristic find the optimal and feasible solutions faster than the BM and HR disjunctive branch and bound. Out of the two algorithms with primal heuristic, the one that uses the Lagrangean relaxation finds the optimal and first feasible solution in fewer nodes.

Table 2: Average of performance metrics for the different algorithms

Average of metric	BM	HR	LAG	RAN
Solution time (s)	172.6	93.0	167.4	143.9
Total number of nodes	1,546	859	763	843
Number of nodes to find optimal	1,258	447	229	364
Number of nodes to find feasible	493	214	2	66.5

6.3. Contracts problem

Given feedstock requirements of a raw material at every time period $D_t; t \in T$, the problems consists on finding the best purchasing contracts to minimize costs. The inventory of the feedstock s_t allows to carry material from a time period to the next ones, but there is a cost associated with inventory α_t^S . In general, there are three types of contracts. The “standard contract” which allows buying any amount of material at a given price γ_t . The “bulk contract” which provides a discount β_t^B of the material, but there is a minimum purchase requirement $F_t^{B,lo}$. The last type of contract requires purchasing materials for the following $q \geq 1$ time periods, all of which include the same discount β_{tq}^L over the same price γ_t and the same minimum purchase requirement $F_{tq}^{L,lo}$.

The GDP of the contracts problem is as follows:

$$\begin{aligned}
& \min \sum_{t \in T} (\alpha_t^S s_t + c_t) \\
s.t. \quad & f_t \geq D_t & t \in T \\
& s_t = s_{t-1} + x_t - f_t & t \in T \\
& \left[\begin{array}{c} Y_t^S \\ c_t = \gamma_t x_t \end{array} \right] \vee \left[\begin{array}{c} Y_t^B \\ c_t = (1 - \beta_t^B) \gamma_t x_t \\ x_t \geq F_t^{B,lo} \end{array} \right] \vee \\
& \vee \left[\begin{array}{c} Y_t^0 \\ 0 \leq c_t \end{array} \right] \vee_{q=1, \dots, |T|-t} \left[\begin{array}{c} Y_{tq}^L \\ c_{t'} = (1 - \beta_{tq}^L) \gamma_t x_{t'} \quad t' = t, \dots, t+q \\ x_{t'} \geq F_{tq}^{L,lo} \quad t' = t, \dots, t+q \end{array} \right] & t \in T \\
& Y_t^S \vee Y_t^B \vee Y_t^0 \quad \vee_{q=1, \dots, |T|-t} Y_{tq}^L & t \in T \\
& Y_t^0 \Leftrightarrow \bigvee_{\substack{t' < t \\ q \geq t-t'}} Y_{t'q}^L & t \in T \\
& Y_1^0 = False \\
& 0 \leq x_t \leq x^{up} & t \in T \\
& 0 \leq c_t \leq c^{up} & t \in T \\
& 0 \leq s_t \leq s^{up} & t \in T \\
& Y_t^S, Y_t^B, Y_t^0 \in \{True, False\} & t \in T \\
& Y_{tq}^L \in \{True, False\} & 1 \leq q \leq |T| - t; \quad t \in T
\end{aligned} \tag{7}$$

The global constraints in (7) enforce the demand satisfaction ($f_t \geq D_t$) and the material balance at the inventory ($s_t = s_{t-1} + x_t - f_t$). At each time period $t \in T$ there is a disjunction. The term that corresponds to the Boolean variable Y_t^S represents the “standard contract”, where any amount x_t can be purchased at price γ_t . Y_t^B represents the “bulk contract”, where $x_t \geq F_t^{B,lo}$ can be purchased with a discount in the price β_t^B . Y_t^0 is active when a long term contract from a previous time period is selected, and such a contract involves time period t ($Y_t^0 \Leftrightarrow \bigvee_{\substack{t' < t \\ q \geq t-t'}} Y_{t'q}^L$). For example, if $Y_{1,3}^L$ is selected (e.g. in the time period $t = 1$, a contract with a length of $q = 3$ time periods was selected), then $Y_2^0 = Y_3^0 = Y_4^0 = True$. The term associated with Y_t^0 does not constrain the variables, since the corresponding cost and purchase bound are determined at the time period in which the long term contract is active ($c_{t'} = (1 - \beta_{tq}^L) \gamma_t x_{t'}; x_{t'} \geq F_{tq}^{L,lo}; t' = t, \dots, t+q$). For the first time period, Y_1^0 cannot be selected. When performing the BM or HR reformulation, the logic constraint ($Y_t^0 \Leftrightarrow \bigvee_{\substack{t' < t \\ q \geq t-t'}} Y_{t'q}^L$) can be reformulated as ($y_t^0 = \sum_{\substack{t' < t \\ q \geq t-t'}} y_{t'q}^L$).

Note that (7) involves logic constraints. Therefore, in order to preserve Properties 3.2 and 3.3, it would be necessary to use (3). However, the continuous relaxation of the HR and BM reformulations of (7) (as

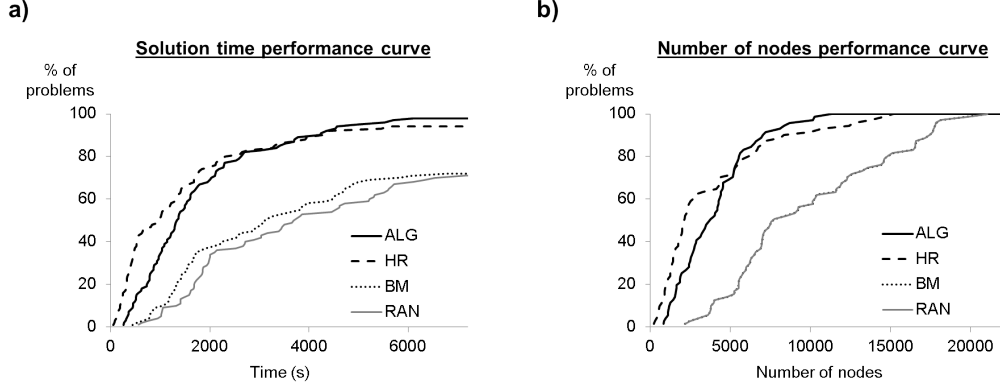


Figure 9: Performance curves for the different algorithm for: a) solution time, and b) number of nodes for the 71 instances that all algorithms solve.

well as (LHR_λ) with the logic constraints reformulated for the original binary variables) can be solved in a fraction of a second. Therefore, Property 3.2 (which established that (LHR_λ) can be solved by solving small LPs) does not have an important impact in the solution time of the problem. Furthermore, out of the over 2.2 million nodes solved for the instances of this example, there was not a case in which the continuous relaxation of (LHR_λ) with the logic constraints reformulated for the original variables gave a non-integer solution. While this is no proof that Property 3.3 holds, it indicates that (LHR_λ) with the logic constraints reformulated for the original variables can be used as primal heuristic for the tested instances.

The different algorithms were tested with 100 random instances of the contracts problem. The range of values of the random parameters is as follows: $|T| = 9-11$; $D_t = 50-100$; $\alpha_t^S = 5-20$; $\gamma_t = 10-30$; $\beta_t^B = 0.050-0.500$; $\beta_{tq}^L = 0.010-0.999$; $F_t^{B,lo} = 50-100$; $F_{tq}^{L,lo} = 50-100$. The algorithm uses (LHR_λ) with the logic constraints reformulated for the original variables as the primal heuristic.

Figure 9 presents the performance plots of the algorithms for solving the contracts problem instances. Figure 9a shows the performance of the solution times. It can be observed that the HR disjunctive branch and bound performs the best for the smaller problems. However, the proposed algorithms outperforms the HR after about 4,000 seconds. Of the 100 instances, the proposed algorithm solves 98, the HR 94, the BM 72, and the random heuristic 71. It is clear from Figure 9a that the BM and heuristic algorithms perform much worse than the other two for this problem. Figure 9b shows the number of nodes required to solve the 71 instances that all algorithms solve. Similar to the solution time, the HR and the proposed algorithm are the best performers. Their performance is close, the HR being better at first and the proposed algorithm taking over in the more difficult problems. In this example, the heuristic algorithms performs exactly the same as the BM branch and bound, in terms of number of nodes. This indicates that the random heuristic is poor, since it requires additional work and it does not reduce the number of nodes required for the BM disjunctive branch and bound.

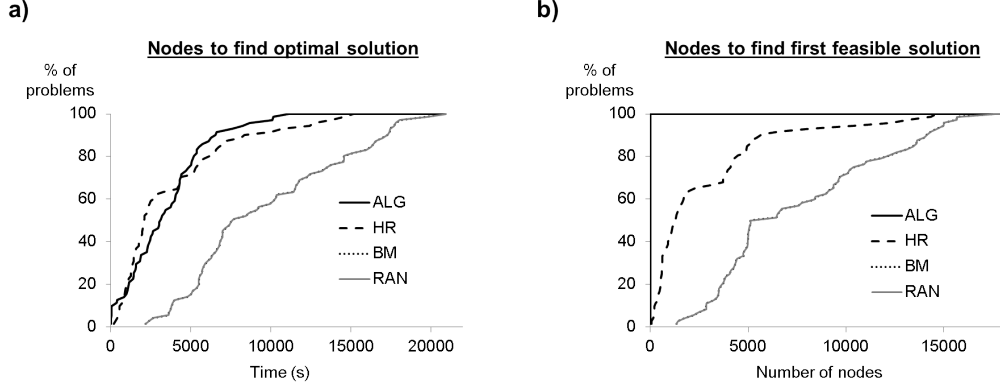


Figure 10: Number of nodes required to find: a) the optimal solution, and b) the first feasible solution for the 71 instances that all algorithms solve.

Figure 10 presents the number of nodes required to find the optimal solution and the first solution. Figure 10a shows that the proposed algorithm is faster in finding the optimal solution. Furthermore, Figure 10b shows that the algorithm finds a feasible solution in every instance at the root node. Again, these two figures show that the heuristic algorithms performs exactly the same as the BM branch and bound in terms of number of nodes. For this example, the BM and random heuristic branch and bound perform much worse than the HR and the proposed algorithm.

7. Conclusions

In this work, we have presented a Lagrangean relaxation of the hull-reformulation of linear GDP problems. This Lagrangean relaxation can be applied to any linear GDP. We proved two important properties of this relaxation. The first one is that any vertex of the continuous relaxation of the Lagrangean relaxation yields 0-1 values for the binary variables of the hull reformulation. The second one is that it can be solved by solving several small LPs (potentially in parallel). This relaxation was incorporated into a disjunctive branch and bound as a primal heuristic, and the proposed algorithm was tested with 300 instances on 3 problems: unstructured GDPs, strip packing and contracts.

For every problem, the number of nodes required by the proposed algorithm is smaller than the number of nodes required by the HR, BM, and “random heuristic” disjunctive branch and bound. Furthermore, the number of nodes to find a feasible and optimal solution to problems is drastically improved when using the Lagrangean relaxation as primal heuristic. Over all the instances, the solution time performance of the proposed algorithm is better than alternative disjunctive branch and bound methods. Over all the instances, the BM, HR, ALG, and RAN solve 253, 266, 296, and 269, respectively.

In terms of solution time, the proposed algorithm and the heuristic one are the best for the unstructured

GDP instances. Not only is their solution time performance curve better, but they solve considerably more instances than the rest within the time limit (98% for the proposed and heuristic algorithms vs. 81% for the BM, and 72% HR). For the two examples of structured GDPs, the performance of the solution time is different for each of the problems. For the strip packing problem, the HR disjunctive branch and bound performs the best, while the proposed algorithm performs third (and very close to the last performer: the BM disjunctive branch and bound). For the contracts problem, the proposed algorithm performs the best, solving 98% of the instances (while the BM, HR, and random heuristic algorithm solve 72%, 94%, and 71%). The performance curve is similar to that of the HR disjunctive branch and bound, but it is able to solve more instances.

The proposed Lagrangean relaxation can be extended to nonlinear convex GDP problems. Future work will address these problems through the Lagrangean relaxation described in this paper, as well as improvements in the implementation.

Acknowledgments

The authors would like to acknowledge financial support from the Center for Advanced Process Decision-making (CAPD).

References

- [1] M. Guignard, Lagrangean relaxation, *Top* 11 (2) (2003) 151–200.
- [2] M. L. Fisher, The lagrangian relaxation method for solving integer programming problems, *Management science* 50 (12_supplement) (2004) 1861–1871.
- [3] A. Rong, R. Lahdelma, P. B. Luh, Lagrangian relaxation based algorithm for trigeneration planning with storages, *European Journal of Operational Research* 188 (1) (2008) 240–257.
- [4] S. Terrazas-Moreno, P. A. Trotter, I. E. Grossmann, Temporal and spatial lagrangean decompositions in multi-site, multi-period production planning problems with sequence-dependent changeovers, *Computers & Chemical Engineering* 35 (12) (2011) 2913–2928.
- [5] G. Cornuejols, M. L. Fisher, G. L. Nemhauser, Exceptional paper-location of bank accounts to optimize float: An analytic study of exact and approximate algorithms, *Management science* 23 (8) (1977) 789–810.
- [6] C. C. Carøe, R. Schultz, Dual decomposition in stochastic integer programming, *Operations Research Letters* 24 (1) (1999) 37–45.
- [7] A. M. Geoffrion, *Lagrangean relaxation for integer programming*, Springer, 1974.

- [8] R. Raman, I. E. Grossmann, Modelling and computational techniques for logic based integer programming, *Computers & Chemical Engineering* 18 (7) (1994) 563–578.
- 525 [9] L. A. Wolsey, G. L. Nemhauser, *Integer and combinatorial optimization*, John Wiley & Sons, 2014.
- [10] F. Trespalcios, I. E. Grossmann, Improved big-m reformulation for generalized disjunctive programs, *Computers & Chemical Engineering* 76 (2015) 98–103.
- [11] N. Beaumont, An algorithm for disjunctive programs, *European Journal of Operational Research* 48 (3) (1990) 362–371.
- 530 [12] S. Lee, I. E. Grossmann, New algorithms for nonlinear generalized disjunctive programming, *Computers & Chemical Engineering* 24 (9) (2000) 2125–2141.
- [13] I. E. Grossmann, F. Trespalcios, Systematic modeling of discrete-continuous optimization models through generalized disjunctive programming, *AIChE Journal* 59 (9) (2013) 3276–3295.
- [14] H. P. Williams, *Model building in mathematical programming*, John Wiley & Sons, 2013.
- 535 [15] E. Balas, Disjunctive programming: Properties of the convex hull of feasible points, *Discrete Applied Mathematics* 89 (1) (1998) 3–44.
- [16] I. E. Grossmann, S. Lee, Generalized convex disjunctive programming: Nonlinear convex hull relaxation, *Computational Optimization and Applications* 26 (1) (2003) 83–100.
- [17] J. P. Ruiz, I. E. Grossmann, A hierarchy of relaxations for nonlinear convex generalized disjunctive programming, *European Journal of Operational Research* 218 (1) (2012) 38–47.
- 540 [18] J. N. Hooker, Logic-based methods for optimization, in: *Principles and Practice of Constraint Programming*, Springer, 1994, pp. 336–349.
- [19] J. N. Hooker, *Integrated Methods for Optimization*, Springer US, 2012.
- [20] I. I. CPLEX, V12. 1: Users manual for cplex, International Business Machines Corporation 46 (53) (2009) 157.
- 545 [21] G. D. Corporation, *General Algebraic Modeling System (GAMS) Release 24.4.5*, Washington, DC, USA (2015).
URL <http://www.gams.com/>
- [22] N. W. Sawaya, I. E. Grossmann, A cutting plane method for solving linear generalized disjunctive programming problems, *Computers & Chemical Engineering* 29 (9) (2005) 1891–1913.
- 550

- [23] F. Trespacios, I. E. Grossmann, Symmetry breaking for generalized disjunctive programming formulation of the strip packing problem, *Annals of Operations Research* Submitted for publication.