Acta Cybernetica **26** (2024) 475–499.

Comparing Structural Constraints for Accelerated Branch and Bound Solver of Process Network Synthesis Problems*

Emília Heinc[ab] and Balázs Bánhelyi[ac]

Abstract

The P-Graph methodology can be used to find the optimal solution for large processing system. This methodology solves the combinatorial part of the problem more efficiently than the traditional branch and bound method due to the utilized relationships inherent in the structure. However, reducing the number of possibilities developed in the constraint functions also plays a major role in this algorithm. In this publication, we present a new constraint function that also takes into account the minimum cost structure and compares it with earlier versions.

Keywords: P-Graph, Accelerated Branch and Bound, structural constraint

1 Introduction

The task of process network synthesis is to determine the optimal structure of a process system, the optimal configurations, and operating sizes of the functional units that make up the system and perform various operations [12]. Process synthesis plays a critical role in reducing material, energy consumption, and negative environmental impacts, thereby increasing profitability. Several examples in the literature demonstrate that efficient process synthesis can reduce energy consumption by up to 50% and costs by 35% [13]. Ideally, the structure of a process and the operational configurations that make up the process could be designed and synthesized simultaneously because their performance interacts. In practice, however, it is extremely difficult due to the simultaneous continuous and discrete nature of the task. The discrete nature is caused by the structure of the process, which leads to

*The research was supported by the Government of Hungary and funded by the European Union Recovery and Resilience Plan within the framework of the Artificial Intelligence National Laboratory Program (RRF-2.3.1-21-2022-00004).

[a]Institute of Informatics, University of Szeged, Hungary
[b]E-mail: heincze@inf.u-szeged.hu, ORCID: 0009-0000-2172-505X
[c]E-mail: banhelyi@inf.u-szeged.hu, ORCID: 0000-0003-4408-9054

DOI: 10.14232/actacyb.297735

the combinatorial complexity of the problem that makes it complex to find an optimal solution to the problem. The process network synthesis problems formulate a MILP problem with many binary variables. Finding the optimal subnetwork is an NP-hard problem. Combinatorial analysis can be applied to this type of problem. The method is used to reduce the number of possible solutions by exploiting the unique properties of the so-called PNS (Process Network Synthesis) problems is the ABB (Accelerated Branch and Bound) method [10]. It is based on the branch and bound method, i.e. the method uses a lower bound submethod to exclude solutions that cannot provide a better solution than the currently known best solution. It is critical for the computation time of solving the problem with the B&B method to find a tighter lower bounding submethod. The currently available implementations and the previous studies do not exploit all the information, considering only the continuous part of the problem by calculating the LP relaxation of the MILP problem. In this article, we introduce a better lower bounding sub-method taking into consideration not just the continuous but also the structural nature of the PNS problem.

2 The Process Network Synthesis

2.1 P-Graph and basic notations methodology

The P-Graph (Process Graph) methodology was developed in the early 1990s for the complex chemical production system to model and optimize. Its name derives from a directed graph obtained by P-Graph, which provides the ability to use combinatorially feasible solution structures to determine the optimum for large tasks [8]. The P-Graph methodology based on graph theory and combinatorial techniques provides a solution to facilitate finding the optimal Process Network Synthesis (PNS) subproblem. The P-Graph can be described with (M, O) structures, where the M is the finite set of materials, and the finite set of operating units, $O \in \wp(M) \times \wp(M)$. The two sets are disjoint, i.e. $M \cap O = \emptyset$.

Definition 1. *The* P-Graph(M', O') *is the subgraph of the* P-Graph(M, O)*, i.e.* P-Graph$(M', O') \subseteq$ P-Graph(M, O) *if* $M' \subseteq M$ *and* $O' \subseteq O$.

The Process Network Synthesis problems, or PNS problems, in short, are defined as (P, R, O) triplets, where P stands for the set of products, R stands for the set of resources or raw materials and O is the set of operating units, where $P \cap R = \emptyset$, $P \subseteq M$, $R \subseteq M$, and $M \cap O = \emptyset$. If $(\alpha, \beta) \in O$, then α is the *input-set*, and β is the *output-set* of (α, β). The sets of input and output materials of set o of operating units are denoted by $mat^{in}(o)$ and $mat^{out}(o)$ separately, which are defined as:

$$mat^{in}(o) = \bigcup_{(\alpha,\beta)\in o} \alpha \text{ and } mat^{out}(o) = \bigcup_{(\alpha,\beta)\in o} \beta.$$

Let the either consumed or produced materials by the operating unit o be:

$$mat(o) = mat^{in}(o) \cup mat^{out}(o).$$

M is the set of materials in the PNS problem that are used (consumed or produced) by at least one operating unit from the set O, i.e. $M = \bigcup_{o \in O} mat(o)$.

In the methodology, a directed bipartite graph was used to represent the structure of a process system. We distinguish two kinds of nodes, the material (set of M) and operating units (set of O) in the graph. The directed edges represent the connection between the operating units and materials. The edges from the materials to the operating units mark the relation of the operating units that consume the materials. The edges from the operating units to the materials represent the relation of producing the given materials. In the PNS problems, costs can be assigned to the operating units and raw materials. In the following sections, the fix, installation cost is denoted by $fix\_cost(O') : \wp(O) \rightarrow \mathbb{R}_{\geq 0}$ and the operating cost is marked by $op\_cost(O') : \wp(O) \rightarrow \mathbb{R}_{\geq 0}$.

Definition 2. *The* P-Graph(m, o) *is a combinatorially feasible structure or solution structure, in short of the PNS problem (P, R, O) if it satisfies the following five listed axioms:*

(S1) $P \subset m$

(S2) $\forall X \in m, X \notin mat^{out}(o)$ *if and only if* $X \in R$

(S3) $o \subseteq O$

(S4) $\forall y_0 \in o, \exists\ path[y_0, y_n]$, *where* $y_n \in P$

(S5) $\forall X \in m, \exists(\alpha, \beta) \in o$ *such that* $X \in (\alpha \cup \beta)$

The aim of the problem is that the solution structure with the optimal summed cost is found, i.e. to produce all of the products from the raw materials at minimum cost. The algorithm that finds all the possible solution structures, will be the SSG algorithm, and the method that finds the solution structures with the optimal summed cost will be the ABB algorithm.

To solve this problem, we have to first find the maximum solution structure which contains all combinatorially feasible process structures. This method is called the MSG method (Maximal Structure Generation) [7].

Definition 3. *Let* $\Delta : M \rightarrow \wp(O)$, *where* $\Delta(X) = \{(\alpha, \beta) : (\alpha, \beta) \in O$ *and* $X \in \beta\}$ *i.e. determines the set of operating units producing all materials* $X \in M$.

Definition 4. *Let the decision mapping* $\delta(X)$ *be the subset of* $\Delta(X)$, *i.e.* $\delta(X) \in \Delta(X)$ $X \in M$.

Definition 5. *Expanding the decision mapping definition for the set of materials let the* $\delta[m] = (X, \delta(X) \mid X \in m)$.

Let the set of operating units of decision-mapping $\delta[m]$ be marked as $op(\delta[m])$, where

$$op(\delta[m]) = \bigcup_{X \in m} \delta(X).$$

Let the complement of decision-mapping $\delta[m]$ defined by

$$\overline{\delta}[m] = \{(X, Y) \mid X \in m \text{ and } Y = \Delta(X) \setminus \delta(X)\}.$$

Let $\overline{\delta}(X)$ be a set of operating units not included in $\delta(X)$:

$$\overline{\delta}(X) = \Delta(X) \setminus \delta(X).$$

Definition 6. *Decision mapping $\delta[m]$ is consistent if $|m| \leq 1$, or $(\delta(X) \cap \delta(Y)) \cup (\overline{\delta}(X) \cap \overline{\delta}(Y)) = \Delta(X) \cap \Delta(Y) \, \forall \, X, Y \in m$.*

Definition 7. *Let $\delta_1[m_1]$ and $\delta_2[m_2]$ be consistent decision-mappings. Then $\delta_1[m_1]$ is an extension of $\delta_2[m_2]$, i.e. $\delta_1[m_1] \geq \delta_2[m_2]$ if $m_2 \subseteq m_1$ and $\delta_1(X) = \delta_2(X)$ for $X \in m_2$.*

It can easily be proved that there is a bijective transition between the consistent decision mapping $\delta[m]$ and the P-Graph (m, o), where $m = \bigcup_{(\alpha,\beta)\in o}(\alpha \cup \beta)$ and $o = \bigcup_{X\in m} \delta(X)$ [6].

Definition 8. *Let the set of included operating units in $\delta[m]$ decision mapping be noted as O_I, where then*

$$O_I = op(\delta[m]).$$

Let the set of excluded operating units in the $\delta[m]$ decision mapping be O_E, where

$$O_E = op(\overline{\delta}[m]).$$

2.2 Solution Structure Generation algorithm

Further investigation is aided by the SSG (Solution Structure Generation) algorithm, which generates each combinatorially feasible structure exactly once. The algorithm is based on decision mappings. Decision mappings involve deciding which operating units produce the materials, i.e. which operating units are involved in a given solution structure [9]. Consequently, during decision mapping, we also decide which operating units will be excluded from the given structure. We must be consistent in our decisions because even if it has already been decided that an operating unit for one material should not be included in the structure, we cannot choose again when deciding on another material. All output materials for an operating unit, if included in the structure, must be specified, an inconsistent decision would result in certain substances being produced and certain substances not. The SSG implementation of the decision mapping-based algorithm calls itself recursively [6, 5].

As we see in the Algorithm 1, the procedure returns with all possible decision mappings over the PNS(P, R, O) problem. In the further section, the ABB algorithm will be introduced which is based on the SSG algorithm as working over all the possible decision mappings in the input PNS(P, R, O) problem but it returns with the optimal cost decision mapping.

Algorithm 1 Main and recursive part of SSG algorithm

```
Input $M, PNS(P,R,O)$
  procedure SSG($M, PNS(P,R,O)$)
  if $P = \emptyset$ then                                   If there is nothing to produce
    Stop
  end if
  SSGD($p, \emptyset, \emptyset$)
  return
  procedure SSGD(p, m, $\delta[m]$)
  if $p = \emptyset$ then
    write $\delta[m]$                                        $\delta[m]$ defines a solution-structure
    return
  end if
  let $x \in p$
  $C := \wp(\Delta(x)) \setminus \emptyset$
  for $\forall c \in C$ do
    if $\forall y \in m$, $(c \cap \overline{\delta}(y) = \emptyset$ and $(\Delta(x) \setminus c) \cap \delta(y) = \emptyset)$ then
      $\delta[m \cup \{x\}] := \delta[m] \cup (x, c)$
      $SSGD\left(\left(p \cup mat^{in}(c)\right) \setminus (R \cup m \cup \{x\}), m \cup \{x\}, \delta[m \cup \{x\}]\right)$
    end if
  end for
  return
```

2.3 Mathematical model for P-Graph

The continuous variables of the model are denoted by x and the binary variables by y.

These variables are assigned to operating units. The continuous variable x_i indicates the operational size of the operating unit $O_i (\in O)$, and the binary variable y_i indicates whether the unit is in the structure or not: if the value of the binary variable $y_i \in \{0, 1\}$ is 0, then the operating unit O_i is not in the structure, and if it is 1, then it is involved. If the operation unit O_i is part of the structure, i.e. $y_i = 1$, then the operation size of the operation unit, which is a continuous variable x_i, can take any value from 0, and the operation unit between its upper capacity limit, U_i. Formally: $x_i \leq y_i U_i$, where U_i is the upper bound of the capacity of the operating unit O_i. If nothing of the sort is defined in the task boundary, an arbitrarily large number M can be used instead of U_i.

The objective function is to minimize cost. The cost is composed of the investment cost, the operating cost of the operating units, and the price of the raw material. These components cover the full cost of the network, i.e. the process to be synthesized considers the full cost. In the model, the costs of the operating units are simply entered with the relationship $a + bx$, where x is the size or capacity of each operating unit, a is the fixed cost, and b is the proportional cost which contains the price of the raw material.

In addition to the capacity constraints listed above, additional constraints are imposed on material balances, products, and raw materials. For products, we usually set lower limits to determine how much we need to produce at least of a given product, while for raw materials we may set upper limits if these types of raw material quantities are not available in unlimited amounts. Material balance conditions should be defined for intermediate products. These conditions state that at least as much of each intermediate product must be produced as is necessary for the operation of the operating units that use it, otherwise the manufacturing process would come to a standstill.

2.4 Accelerated Branch and Bound method

Since this is a mixed-integer programming problem, a general branch-and-bound-type method can be used to solve this model. Although the optimal solution to the problem can also be determined by using these methods, their efficiency can be further improved since the special properties of the synthesis tasks are not taken into account in the search for a solution. Accordingly, the P-Graph method for determining the optimal solution is a special algorithm of the Constraint and Separation type, ABB is used.

This algorithm uses the previously described decision mappings of the SSG algorithm for binary variables in the branch-and-bound tree. Earlier on, the branch-and-bound method used continuous relaxation of the mathematical model in addition to the structural constraints of the constraint SSG. In this relaxed model, the binary variables (y_i) were not considered, and the model was limited to determining the optimal values of the continuous variables (x_i). This optimization task provided a lower bound on the operating costs. In the following section, when the lower bounds are defined precisely for our branch-and-bound method, the relaxation type of the bound is nominated as $Lower\_Bound_{relaxed}$ (see Algorithm 2).

Algorithm 2 Relaxed lower bound algorithm

```
Input $PNS(P, R, O)$ problem, $O_I$, $O_E$
  procedure $Lower\_Bound_{relaxed}(PNS(P, R, O), O_I, O_E)$
  return $LP_{Solver}(PNS(P, R, O), O_I, O_E)$
```

The ABB algorithm is given as Algorithm 3 below. The method's inputs are the $PNS(P, R, O)$ problem in which the algorithm is running over, and the *Lower_Bound* function that will be used to prune the branch-and-bound tree. The M and *neutralExtension* variables are used implicit by ABB algorithm. The M denotes the set of materials that will be considered, and the *neutralExtension* is a Boolean variable that decides whether the neutral extension acceleration will be used or not.

The ABBD sub-method is called recursively in the ABB algorithm. It can be defined as a node from the branch-and-bound tree when it is called. The decision-

Algorithm 3 Main and recursive part of ABB algorithm

```
Input $PNS(P, R, O), Lower\_Bound$
Global variables $R, \Delta(x), (x \in M), U, currentbest$
  procedure ABB($PNS(P, R, O), Lower\_Bound$)
  $U := \infty; currentbest := \infty$
  $O := MSG(PNS(P, R, O))$
  ABBD(P, $\emptyset$, $\delta[\emptyset]$)
  return
  procedure ABBD(p, m, $\delta[m]$)
  if neutralExtension then
    let $\hat{\delta}[\hat{m}]$ be the maximal neutral extension of $\delta[m]$
    $p := (mat^{in}(op(\hat{\delta}[\hat{m}])) \cup P) \setminus (\hat{m} \cup R)$
    $O_I := op(\hat{\delta}[\hat{m}])$
    $O_E = op(\bar{\hat{\delta}}[\hat{m}])$
  end if
  $bound = Lower\_Bound(PNS(P, R, 0), O_I, O_E)$
  if $p = \emptyset$ then                                    Halting condition.
    if $U \geq bound$ then
      $U = bound$;
      update currentbest;
    end if
    return
  end if
  if $bound \geq U$ then                                     Cutting the branch.
    return
  end if
  $x \in p$;
  $C := \wp(\Delta(x)) \setminus \{\emptyset\}$;
  for $\forall c \in C$ do
    if $\forall y \in m, c \cap \bar{\delta}(y) = \emptyset \& (\Delta(x) \setminus c) \cap \delta(y) = \emptyset$ then
      $m' := m \cup \{x\}$;
      if $S(\delta[m']) = \emptyset$ then
        Continue;
      end if
      $\delta[m'] := \delta[m] \cup \{(x, c)\}$;
      $p := (mat^{in}(op(\delta[m'])) \cup P) \setminus (m' \cup R)$;
      $O_I := op(\delta[m'])$;
      $O_E := op(\bar{\delta}[m'])$;
      $ABBD(p, m', \delta[m'])$
    end if
  end for
  return
```

mapping is the fundamental tool for finding the optimal solution structure, as it defines obviously the node, or partial problem, S of the branch-and-bound tree. First, the partial problem has to be defined precisely for the algorithm. The definition of the S will be:

Definition 9. *Let $S(\delta[m_i])$ be the partial problem of ABB in solving PNS problem (P, R, O):*

$$S(\delta[m_i]) := \{\delta[m_k] : \delta[m_k] \geq \delta[m_i] \text{ and } graph(\delta[m_k]) \in PNS(P, R, O)\},$$

where $graph(\delta[m_k]) \in PNS(P, R, O)$ means that $\delta[m_k]$ is the combinatorially feasible structure of $PNS(P, R, O)$.

The ABBD(P, $\emptyset$, $\delta[\emptyset]$) or $S(\emptyset)$ is called for the root node in the tree. The first parameter is the set of materials that are obligatory to produce. In the root node, it is the products from the PNS problem. The second parameter is the set of materials that have been produced already. The third parameter is the decision mapping which is valid in the current branch. As there is no material produced in the root node, the last two parameters are zero in the root node.

As an acceleration of the algorithm, the neutral extension of the current decision mapping, $\delta[m]$ can be used. The method extends the decision mapping with the materials that have to be produced. If we have one possible way to produce it, then the decision of which operating unit will produce these materials is straightforward [10]. It could be either that (1) all of the operating units are included and excluded, i.e. $\Delta(m) \cap (O_I \cup O_E) = \Delta(m)$, or (2) it is not decided which operating units produce it, but there is one possible operating unit that could produce it, the other operating units have already been excluded, i.e.$(O_I \cap \Delta(m) = \emptyset)$ & $(|\Delta(m) \setminus O_E| = 1)$.

The halting condition has been defined in the ABBD algorithm. If there aren't any obligatory products then the algorithm returns with the currently best cost. In this case from the validity of the lower bound sub-method, the return value equals the optimal cost of the current examined branch.

In the ABBD sub-method, as we calculate the minimal cost of the sub-tree in the branch-and-bound tree, the lower bound has to be calculated for the summed cost. The *Lower_Bound* must be a valid lower bound sub-method. The validity of the newly introduced lower bound will be proved in Theorem 1.

If the lower bound is greater than the actual best solution then the branch is going to be cut because the optimal solution can't be better than the current best solution.

After the examination of the halting conditions, the branching part is executed: as it is introduced in the SSG algorithm, we select a material x from the set of mandatory products, p that hasn't been decided which operating units produce yet. The son of the current partial problem will be the recursive sub-problem. The recursively called method's parameter is the decision mapping that consists of the material x and it is consistent with the current decision mapping in the main problem. Formally defined as:

$$\begin{aligned} son(S(\delta[m]), x) := \quad & \{S(\delta'[m']) : S(\delta'[m']) \neq \emptyset \ \& \ \delta'[m'] = \delta[m] \cup \{\, (x, c) \,\} \\ & \quad for\ c \in (\wp(\Delta(x))) \setminus \emptyset \ \& \ \delta'[m'] \ is\ consistent\}\,. \end{aligned}$$

3 Lower bound submethods in ABB algorithm

In each of our relaxed models in our ABB algorithm with a new lower bound, the minimum cost of the structurally feasible part of the residual was determined by the modified SSG algorithm for the free part of the P-Graph. The modified SSG algorithm is similar to the ABB method that has been called recursively with the sum of the fixed costs as the lower bound. The combination of these two optimization models gives a better lower bound for the sub-problems of B&B. Of course, operational and structural lower bounds need not necessarily come from the same feasible structure. The general $Lower\_Bound$ function inputs are the $PNS(P, R, O)$ problem the algorithm is running over, and the currently included and excluded operating units in the branch by the decision mapping $\delta[m]$.

3.1 Relaxed lower bound

Let us first consider the well-known and commonly used relaxed model, which is typically used for B&B algorithms and basic P-Graph solutions.

Definition 10. *$LP_{Solver}(PNS(P, R, O), O_I, O_E)$ is the optimal value of PNS problem with $y_i = 0$, where $o_i \in O_E$ and $y_i = 1$, where $o_i \in O_I$.*

The most obvious lower bound will be the optimum of the relaxation of the current MILP model derived from the decision mapping. The relaxed optimum of the current MILP problem is denoted as $LP_{Solver}(PNS(P, R, O), O_I, O_E)$, where O_I is the included operating unit, which means that also in the relaxed problem the $y_i = 1 \,|\, \forall o_i \in O_I$ is set. As the $x_i \leq y_i U_i$ constraints have been set for the operation number of operating unit i.

The same is true for O_E which denotes the excluded operating units also in the relaxed problem, i.e. $y_i = 0 \,|\, \forall o_i \in O_E$. In this case, the operating unit i has not been installed so from the constraints also listed above the operating unit cannot do any operation. (Also it has been excluded.)

The $x_i \leq y_i U_i$ relation is excluded from the constraints for all the free operating units. It means that the y_i for the free operating units will be set to 0 in the optimal solution as all the y variables have non-negative coefficients in the objective function.

To obtain the optimum of the relaxed LP problem that is get from the transformation of the current decision mapping, $\delta[m]$ to an LP problem is essential to choose a reliable LP solver. These LP solvers could be CPLEX [4], XPress [3], or Gurobi [11]. The Gurobi was used in our implementation.

3.2 Defining the new lower bound of ABB algorithm

Our aim is to introduce a new lower bound which gives a tighter bound to the optimum value than the current relaxed lower bound, and the LP solver calling number is not greater than the total LP callings in the case with the relaxed bound.

The new lower bound has been improved to take into consideration not only the currently summed included operating units' cost, but the remained operating units' fix costs as well.

The main idea was that the same ABB algorithm can be used for the calculation of the free or remained operating units with modified parameters. Both parameters of ABB, the PNS problem, and the lower bound have been modified to make the algorithm calculate the optimal structural value. A modified PNS problem is used to calculate the lower bound of the MILP model's integer part. Let $PNS_{IP}^{\delta[m]}(P', R', O')$ be the new PNS problem derived from the original $PNS(P, R, O)$ problem, where

- $R' = mat^{out}(O_I) \cup R$
- $P' = \big(P \cup mat^{in}(O_I)\big) \setminus R'$
- $O' = \{(\alpha', \beta') : \alpha' = \alpha, \beta' = \beta \setminus mat^{out}(O_I), (\alpha, \beta) \in O \setminus O_E\}$.

In the previously introduced ABB algorithm, the $PNS_{IP}^{\delta[m]}$ problem was formulated by the $get\_IP\_problem(PNS(P, R, O), O_I, O_E)$ method, where $O_I = op(\delta[m])$ and $O_E = op(\overline{\delta}[m])$ and $\delta[m]$ is the current decision mapping in the original problem. The second parameter is changed to the $Summed\_Weight$ lower bound (listed in Algorithm 4), which returns the summed cost of the included units. This lower bound cost gives a lower bound to the actual optimal structural cost of the free operating units. It will be proved in the Lemma 4.

Algorithm 4 The current summed structural cost

```
Input $PNS(P', R', O')$ problem, $O_I$, $O_E$
  procedure $Summed\_Weight(PNS(P, R, O), O_I, O_E)$
  return $\sum_{o \in O_I} fix\_cost(o)$
```

The optimal free operating units' structural cost will be added to the previously defined remained graph's optimal operating cost.

The detailed method is listed in Algorithm 5. In the rest of the section, the validity of our new lower bound will be proved.

Definition 11. *Let the A be all the possible extensions of decision mapping $\delta[m]$ over arbitrary $PNS(P, R, O)$ problem.*

$$A(\delta[m]) = \{\delta^*[m^*] \mid \delta^*[m^*] \geq \delta[m] \text{ and } \exists \delta^+[m^+], graph(\delta^+[m^+]) \in PNS(P, R, O) \text{ where } \delta^*[m^*] \leq \delta^+[m^+]\}.$$

Definition 12. *Let the δ' be all the possible decision of materials, and A' be all δ' decision mapping over the problem $PNS_{IP}^{\delta[m]}(P', R', O')$, i.e. $A' = A(\delta'[\emptyset])$.*

Definition 13. *Let the bijective transition F from $O \setminus O_E$ to $O' \setminus O_E$ be $F((\alpha, \beta)) = (\alpha, \beta \setminus mat^{out}(O_I))$, where O' is the operating unit set of $PNS_{IP}^{\delta[m]}$ problem, and O is the operating unit set of the PNS problem.*

Algorithm 5 New lower bound algorithm

```
Input $PNS(P, R, O)$ problem, $O_I$, $O_E$
  procedure $Lower\_Bound_{new}(PNS(P, R, O), O_I, O_E)$
  $PNS_{IP}^{\delta[m]}(P', R', O')$ :=get_IP_problem($PNS(P, R, O), O_I, O_E$)
  $O'$ :=MSG($PNS_{IP}^{\delta[m]}(P', R', O')$)
  IPCurrentBest:=0
  if $P' \neq \emptyset \,||\, O' \neq \emptyset$ then
    IPCurrentBest:=
                  ABB($PNS_{IP}^{\delta[m]}(P', R', O')$, $Summed\_Weight$).currentbest
  end if
  return $LP_{Solver}(PNS(P, R, O), O_I, O_E) + IPCurrentBest$
```

Definition 14. *Let the $\mathcal{F}$ transition be the following: $\mathcal{F} : \wp(O \setminus O_E) \to \wp(O' \setminus O_E)$ $\mathcal{F}(Op) = \bigcup_{o \in Op} F(o)$.*

Definition 15. *Let $PNS_{IP}^{\delta[m]}(P', R', O')$ derived from $PNS(P, R, O)$ problem, where the current decision mapping is $\delta[m]$, i.e.*

$$PNS_{IP}^{\delta[m]}(P', R', O') = get\_IP\_problem(PNS(P, R, O), op(\delta[m]), op(\overline{\delta}[m])).$$

Definition 16. *Let the f be a transition between the $PNS(P, R, O)$ problem's set of decision mappings and the $PNS_{IP}^{\delta[m]}(P', R', O')$ problem's set of decision mappings. f is derived from the F in the following way:*

$$f_{\delta[m]} : A(\delta[m]) \to A', f_{\delta[m]}(\delta^*[m^*]) := \{(X, \mathcal{F}(\delta^*(X))) \,|\, \exists (X, Op) \in \delta^+[m^+] \text{ and } X \in m^*, \text{ where } graph(\delta^+[m^+]) \in PNS_{IP}^{\delta[m]}(P', R', O')\} \text{ and } Op \subseteq O'.$$

where $\delta^*[m^*] \geq \delta[m]$ and $PNS_{IP}^{\delta[m]}(P', R', O')$ is calculated over $\delta[m]$ and $\delta[m] \in A(\delta[\emptyset])$ and also the (S4) axiom is satisfied in $\delta[m]$.

Example 1. In the example shown in Figure 1, produce D are $\{o_1\}$, $\{o_2\}$, $\{o_1, o_2\}$.

If the D is produced by o_1, then $\delta[m] = \{(D, \{o_1\})\}$. If the o_1 operating unit is included, then o_2 gets in the excluded unit set. $\delta^*[m^*]$ can be $\{(D, \{o_1\})\}$ or $\{(D, o_1), (E, o_3)\}$. $f_{\delta[m]}(\{(D, \{o_1\})\}) := \emptyset$, $f_{\delta[m]}(\{(D, o_1), (E, o_3)\}) := \{(E, \{\mathcal{F}(o_3)\})\}$.

If the D is produced by o_2, then $\delta[m] = \{(D, \{o_2\})\}$. The O_E is $\{o_1\}$. The $\delta^*[m^*]$ can be $\{(D, \{o_2\})\}$ or $\{(D, \{o_2\}), (E, \{o_2\})\}$ or $\{(D, \{o_2\}), (E, \{o_2, o_3\})\}$. In the last two cases, f returns with $\emptyset$ since in the original $\delta[m]$, all products have already been produced by o_2, and R' includes the O_I outputs. In the first case, since all materials have also been produced from m^*, return with $\emptyset$.

If the D is produced by o_1 and o_2, then $\delta[m] := \{(D, \{o_1, o_2\})\}$. The $\delta^*[m^*]$ can be $\{(D, \{o_1, o_2\})\}$, $\{(D, \{o_1, o_2\}), (E, \{o_2\})\}$, $\{(D, \{o_1, o_2\}), (E, \{o_2, o_3\})\}$. In all cases, f returns with $\emptyset$ for the same reasons as in the previous case.

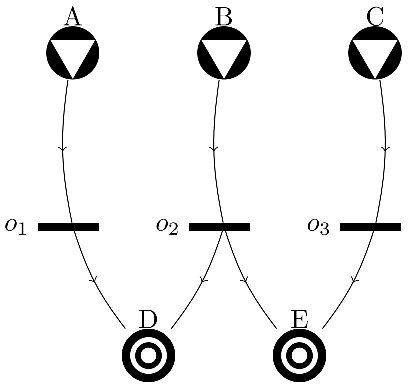


Figure 1: The illustration of an example for representing all possible cases of $f_{\delta[m]}(\delta^*[m^*])$.

Lemma 1. *Let's suppose that $\delta^*[m^*]$ and $\delta^{**}[m^{**}]$ are $\in A(\delta[m])$ and $\delta^{**}[m^{**}] \geq \delta^*[m^*](\geq \delta[m])$. Then $f_{\delta[m]}(\delta^{**}[m^{**}]) \geq f_{\delta[m]}(\delta^*[m^*])$.*

Proof. If $\delta^{**}[m^{**}] \geq \delta^*[m^*]$, then $m^* \subseteq m^{**}$ and $\forall X \in m^* : \delta^*(X) = \delta^{**}(X)$. Let's suppose that $X \in m^*$ and $(X, \mathcal{F}(\delta^*(X))) \in f_{\delta[m]}(\delta^*[m^*])$. Because of the definition of $f_{\delta[m]}$, exists at least one $\delta^+[m^+]$, where $\delta^+[m^+] \in graph(PNS_{IP}^{\delta[m]}(P', R', O'))$ and $X \in m^+$. As X also $\in m^{**}$, then it is also will be produced in $f_{\delta[m]}(\delta^{**}[m^{**}])$. Because $\delta^*(X) = \delta^{**}(X)$, as $(X, \mathcal{F}(\delta^{**}(X))) \in f_{\delta[m]}(\delta^{**}[m^{**}])$, then $(X, \mathcal{F}(\delta^*(X)))$ $\in f_{\delta[m]}(\delta^{**}[m^{**}])$.

□

Lemma 2. *Let suppose that $\delta^*[m^*] \in A(\delta[m])$ and $graph(\delta^*[m^*]) \in PNS(P, R, O)$. Then $graph(f_{\delta[m]}(\delta^*[m^*])) \in PNS_{IP}^{\delta[m]}(P', R', O')$.*

Proof. It is wanted to be proven that $graph(f_{\delta[m]}(\delta^*[m^*]))$ is a solution structure of the $PNS_{IP}^{\delta[m]}(P', R', O')$ problem if $graph(\delta^*[m^*])$ is a solution structure in the $PNS(P, R, O)$ problem. A P-Graphs is considered a solution structure if and only if it satisfies the five axioms listed previously. In the proof, it is shown that $graph(f_{\delta[m]}(\delta^*[m^*]))$ satisfies all the five axioms in the $PNS_{IP}^{\delta[m]}(P', R', O')$ problem.

The first axiom is satisfied if $P' \subset mat^{out}(f_{\delta[m]}(\delta^*[m^*]))$. The P' is equal to $(P \cup mat^{in}(O_I)) \setminus (mat^{out}(O_I) \cup R)$. The P' set is equivalent to $(P \setminus mat^{out}(O_I)) \cup (mat^{in}(O_I) \setminus (mat^{out}(O_I) \cup R))$, as $P \cap R = \emptyset$. It can be divided into two disjoint sets, $P \setminus mat^{out}(O_I)$ and $(I \cap mat^{in}(O_I)) \setminus (mat^{out}(O_I))$, where I is the set of intermediate materials from the original problem $(I = M \setminus (R \cup P))$. Since all connections to $mat^{out}(O_I)$ were eliminated in the $PNS_{IP}^{\delta[m]}$ problem, none of the previously produced materials (i.e. $mat^{out}(O_I)$) are chosen in $f_{\delta[m]}(\delta^*[m^*])$. All materials that are in both I and $mat^{in}(O_I)$ had to be produced. In addition, all elements of P have already been produced. Therefore, all elements from $(P \setminus$

$mat^{out}(O_I)) \cup (mat^{in}(O_I) \setminus (mat^{out}(O_I) \cup R))$ are produced in the resulting P-Graph.

The second axiom states that $\forall X \in m^f, X \notin mat^{out}(o^f)$ if and only if $X \in R'$. From the proof of the first axiom, it is known that the elements of $mat^{out}(O_I)$ are not in $mat^{out}(f_{\delta[m]}(\delta^*[m^*]))$. Based on the second axiom, it can be concluded that R is also not in the set of produced materials.

The third axiom is satisfied if $op(f_{\delta[m]}(\delta^*[m^*])) \subseteq O'$. It is trivially satisfied since the operating units in the P-Graph are determined as $(F(o) \,|\, \forall o \in f_{\delta[m]}(\delta^*[m^*])$.

The fourth axiom states that $\forall y_0 \in op(f_{\delta[m]}(\delta^*[m^*])), \exists\ path[y_0, y_n],\ where\ y_n \in P'$, i.e. for all operating units there is a path to at least one product. All connections to any operating unit to the previously produced materials ($mat^{out}(O_I)$) are eliminated in $f_{\delta[m]}(\delta^*[m^*])$. It induced that, if an operating unit only produces these materials, then it will be left out in $f_{\delta[m]}$, as none of the solution structures in the $PNS_{IP}^{\delta[m]}$ problem contains it. If the operating unit is not eliminated in the resulting decision mapping, then whether it has a path to a product from P, because $\delta^*[m^*]$ is a solution structure in the original problem, or if it does not have a path to any original product, then it has at least one path to material from $mat^{in}(O_I)$. From this material from $mat^{in}(O_I)$ as the fourth axiom also applied for $\delta[m]$, originally, in the $\delta^*[m^*]$ has a path to a product from P.

The fifth axiom states that for all materials, there is at least one operating unit that either produces or consumes them. This axiom is trivially satisfied because the materials in the P-Graph correspond to the consumed and produced materials of the decision mapping. □

Lemma 3. *Images of all extension of $\delta[m]$ in the PNS problem are possible decision mapping in $PNS_{IP}^{\delta[m]}$ problem, i.e. $f_{\delta[m]}(\delta^*[m^*]) \in A'\ \forall \delta^*[m^*] \geq \delta[m]$.*

Proof. The lemma claims that $\exists \delta^+[m^+]$ where $\delta^+[m^+] \in PNS_{IP}^{\delta[m]}(P', R', O')$ and $f_{\delta[m]}(\delta^*[m^*]) \leq \delta^+[m^+]$. As $\delta^*[m^*] \in A(\delta[m])$, this implies that $\exists \delta^+_{PNS}[m^+_{PNS}]$ where $graph(\delta^+_{PNS}[m^+_{PNS}]) \in PNS(P, R, O)$ and $\delta^*[m^*] \leq \delta^+_{PNS}[m^+_{PNS}]$. Because of Lemma 2, $graph(f_{\delta[m]}(\delta^+_{PNS}[m^+_{PNS}]) \in PNS_{IP}^{\delta[m]}(P', R', O')$.

From Lemma 1 and $\delta^*[m^*] \leq \delta^+_{PNS}[m^+_{PNS}]$, $f_{\delta[m]}(\delta^*[m^*]) \leq f_{\delta[m]}(\delta^+_{PNS}[m^+_{PNS}])$. As $graph(f_{\delta[m]}(\delta^+_{PNS}[m^+_{PNS}])) \in PNS_{IP}^{\delta[m]}(P', R, O')$, the $\delta^+[m^+]$ is looking for will be $f_{\delta[m]}(\delta^+_{PNS}[m^+_{PNS}])$. □

Lemma 4. *The summed weight of included units of $f_{\delta[m]}(\delta^*[m^*])$ always is a value less than the summed weight of $\delta^*[m^*]$, i.e. $Summed\_Weight(O_I(f_{\delta[m]}(\delta^*[m^*]))) \leq Summed\_Weight(O_I(\delta^*[m^*]))$.*

Proof. Derived from the definition of $f_{\delta[m]}(\delta^*[m^*])$, if $(X, Op') \in f_{\delta[m]}(\delta^*[m^*])$, then $\exists Op \subseteq O$ where $\mathcal{F}(Op) = Op'$ and $(X, Op) \in \delta^*[m^*]$. It is previously known from the definition of F that the costs of the operating units have not been changed by the transition. It induces that the $Summed\_Weight(Op) = Summed\_Weight(Op')$. □

Lemma 5. *If there is an optimal solution to the $PNS(P, R, O)$ problem, and suppose it is $\delta^*[m^*]$, then $f_{\delta[m]}(\delta^*[m^*])$ is also the optimal solution among all possible $f_{\delta[m]}(\delta^{**}[m^{**}])$ where $\delta^{**}[m^{**}] \in A(\delta[m])$, if the aim of the optimization problem is to find the minimal fix costed solution structure, i.e. P-Graphs that satisfy only the axioms.*

Proof. The statement that $f_{\delta[m]}(\delta^*[m^*])$ is the optimal solution among all possible $f_{\delta[m]}(\delta^{**}[m^{**}])$ where $\delta^{**}[m^{**}] \in A(\delta[m])$ can be proved indirectly. Suppose that, $\exists\delta^{opt}[m^{opt}](\in A)$ where $f_{\delta[m]}(\delta^{opt}[m^{opt}])$ is the optimal solution among the $f_{\delta[m]}(\delta^{**}[m^{**}])$ in the sense of summed fix cost. In this case, the materials from M can be divided into three disjoint categories. The three parts are (1) m previously produced materials, (2) $bp = mat^{out}(O_I) \cap \overline{m}$ set of byproducts, (3) $np = (M \setminus mat^{out}(O_I))$ non-produced materials. It is obvious that $np \cup bp \cup m = m^*$, and the three sets are disjoint.

(1) If $mat \in m$. It is known that $\delta[m] \leq f_{\delta[m]}(\delta^*[m^*])$ and also $\delta[m] \leq f_{\delta[m]}(\delta^{opt}[m^{opt}])$. This part of the material production will be the same in the two possible solution structures.

(2) If $mat \in bp$. In this case, the mat can be produced neither in $f_{\delta[m]}(\delta^*[m^*])$, nor in $f_{\delta[m]}(\delta^{opt}[m^{opt}])$, as $mat \in mat^{out}(O_I)$ and it implies that none of any operating units from the possible arbitrary $f_{\delta[m]}(\delta^{**}[m^{**}])$ can produce mat. If $\delta^*[m^*]$ is the optimal solution in the PNS problem, then none of any mat from bp will be produced, as $mat \in mat^{out}(O_I)$, and only the installation costs count when the objective function was calculated.

(3) If $mat \in np$. In this case, the input and the output materials are remained the same compared to the F mapping of the operating units from $\Delta(mat)$. In other words, if $op \in \Delta(mat)$, then $mat^{in}(op) = mat^{in}(F(op))$ and $mat^{out}(op) = mat^{out}(F(op))$. This means that the operating units, that can produce the mat have remained the same in the sense that they can be defined by the same pair of material sets. It is known that only the materials from np can be produced in $f_{\delta[m]}(\delta^{opt}[m^{opt}])$ and $f_{\delta[m]}(\delta^*[m^*])$ because only these materials are produced in the arbitrary $f_{\delta[m]}$ mapping from the possible three categories.

The $\delta^*[m^*]$ is the optimal solution of the PNS problem, and $op(f_{\delta[m]}(\delta^*[m^*])) \subseteq \mathcal{F}(op(\delta^*[m^*]))$. If the objective function is also divided into three part according to the three previously defined disjoint finite sets of materials (m, bp, and np), then the $Summed\_Weight(op(\delta^*[m^*])) = Summed\_Weight(op(\delta[m])) + Summed\_Weight(op(\delta^*[np \cap m^*]))$, as none of any materials from bp has been produced. $Summed\_Weight(op(\delta^{opt}[m^{out}])) = Summed\_Weight(op(\delta[m])) + Summed\_Weight(op(\delta^{opt}[bp \cap m^{opt}])) + Summed\_Weight(op(\delta^{opt}[np \cap m^{opt}]))$. The $Summed\_Weight(op(\delta^*[np \cap m^*])) > Summed\_Weight(op(\delta^{opt}[np \cap m^{opt}]))$ is known from the indirect statement. Only the materials from np are produced in the $f_{\delta[m]}$ mappings, and the summed weight of $\delta^{**}[m^{**} \cap np]$ is equal to the summed weight of $f_{\delta[m]}(\delta^{**}[m^{**} \cap np])$ for arbitrary $\delta^{**}[m^{**}] \in A(\delta[m])$. It is a contradiction because $\exists\delta'[m'] \in A(\delta[m])$ where $Summed\_Weight(op(\delta'[m'])) < Summed\_Weight(op(\delta^*[m^*]))$ and $\delta'[m'] = \delta[m] \cup \delta^{opt}[np \cap m^{opt}]$ is also a possible solution structure in the PNS problem because $bp \subseteq mat^{out}(O_I)$. It follows that

$bp \subseteq m'$. □

Lemma 6. *$ABB(PNS_{IP}^{\delta[m]}(P', R', O'), Summed\_Weight)$ always returns with a valid lower bound for the $PNS_{IP}^{\delta[m]}(P', R', O')$ problem.*

Proof. Let's suppose that, running the ABB algorithm over $PNS(P, R, O)$ problem, $\delta[m]$ is the decision mapping in the current branch, i.e. the $PNS_{IP}^{\delta[m]}(P', R', O')$ problem is derived from $\delta[m]$. From Lemma 3 is known that, for all possible subnodes of the current branch-and-bound node $\delta^*[m^*]$ (i.e. $\delta^*[m^*] \geq \delta[m]$), the $f_{\delta[m]}(\delta^*[m^*])$ is a possible decision mapping in the $PNS_{IP}^{\delta[m]}$ problem.

It is also known from Lemma 5 that, the optimal solution structure among the possible $f_{\delta[m]}(\delta^*[m^*])$ decision mappings is at least the optimal solution structure of the $PNS_{IP}^{\delta[m]}$ problem.

From the definition of $f_{\delta[m]}$, it is known that, when a $(X, \mathcal{F}(op(X)))$ is added to decision mapping of the current branch, $\delta^*[m^*]$ where $X \in M$ and $op(X) \in \Delta(X)$, then in $f_{\delta[m]}(\delta^*[m^*] \cup (X, op(X)))$ is equal to either $f_{\delta[m]}(\delta^*[m^*])$ or $f_{\delta[m]}(\delta^*[m^*]) \cup (X, \mathcal{F}(op(X)))$.

From Lemma 4 is known that for each step, the summed weight of $f_{\delta[m]}(\delta^*[m^*])$ is less than the summed weight of $\delta^*[m^*]$.

Summing up the previous claims, it implies that, the decision mappings that the $f_{\delta[m]}$ function returns with are possible decision mapping over the $PNS_{IP}^{\delta[m]}$ problem. It is also known that among the possible solution structure, there is an optimal solution structure according to the minimal summed cost. The ABB algorithm always managed to find it over the $PNS_{IP}^{\delta[m]}$ problem only if the ABB algorithm over the PNS problem can find the optimal solution, i.e. the optimal solution exists among the possible $f_{\delta[m]}(\delta^*[m^*])$. And it is also known, that solution structure always gives a lower bound for the optimal solution structure in the original PNS problem, i.e. it provides a lower bound for all possible solution structures in the original problem. □

Theorem 1. *The new lower bound is correct, so*

$$\begin{aligned} &Lower\_Bound_{new}(PNS(P, R, O), O_I, O_E) := \\ &\qquad ABB(PNS_{IP}^{\delta[m]}(P', R', O'), Summed\_Weight) + \\ &\qquad LP_{Solver}(PNS(P, R, O), O_I, O_E), \end{aligned}$$

if P' is not empty, otherwise $LP_{Solver}(PNS(P, R, O), O_I, O_E)$.

Proof. Let $X^* \in \mathbb{R}^n_{\geq 0}$ and $Y^* \in \{0, 1\}^n$, where (X^*, Y^*) is the optimal solution of the $PNS(P, R, O)$ problem and n stands for the number of operating units when the $\delta[m]$ is the current decision mapping in the branch, i.e. $O_I = op(\delta[m])$ and $O_E = op(\overline{\delta}[m])$. Here Y^* stands for the selected operating units, and X^* marks the operational size in the solution. From Lemma 3 we have that $ABB(PNS_{IP}^{\delta[m]}(P', R', O'), Summed\_Weight) \leq fix\_cost(O)^T Y^*$. It is known that $LP_{Solver}(PNS(P, R, O), O_I, O_E) \leq op\_cost(O)^T X^*$. A valid lower bound for the proportional part of the cost is given by $LP_{Solver}(PNS(P,R,O),O_I,O_E)$ because (1) the excluded units' proportional costs are not included due to the constraints $x \leq yM$ and $y = 0$, (2) the included units' proportional costs are included because of the above-listed constraint with $y = 1$, (3) the free units' variables will be set to 0 during the optimization as all the coefficients of the x-s and y-s are nonnegative numbers in the objective function. Because of the statements above, $ABB(PNS_{IP}^{\delta[m]}(P',R',O')) + LP_{Solver}(PNS(P,R,O),O_I,O_E)$ is a valid lower bound for the optimal cost in the $PNS(P,R,O)$ problem with O_I included and O_E excluded sets. □

Corollary 1. *As the fix costs are always non-negative values, our new lower bound always gives a tighter bound than the relaxed version, i.e.*

$$\begin{gathered}Lower\_Bound_{new}(PNS(P,R,O),O_I,O_E) \geq \\ Lower\_Bound_{relaxed}(PNS(P,R,O),O_I,O_E)\end{gathered}$$

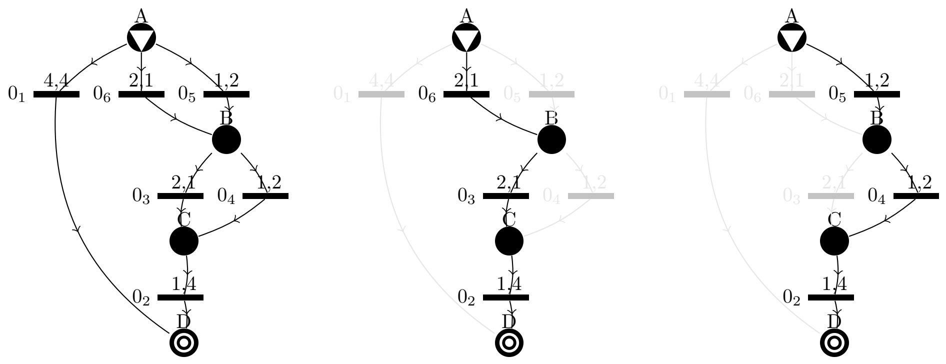
3.3 Illustration of new constraint and relaxed constraint

The following simple example illustrates the efficiency of our algorithm. In our example, we want to produce one product (D) from the raw material (A), using the operating units $O_1, O_2, \ldots, O_5$.

The final product (D) can be produced by either O_1 or O_2 or both operating units. If the machine O_1 chooses to produce the D final product, the O_1 unit consumes only the A raw material. The total production cost will be the sum of the fixed and proportional costs of the O_1 operating unit. The optimal solution is $y_1 = 1, x_1 = 1$, and the other variables are 0.

Consider another branch that chooses O_2. In this case, our previous production cost is $1 + 4$, because $y_2 = 1$ and $x_2 = 1$. In the original version, the operating cost of producing C is added to this cost. The optimal solution for $x_3 = 1$ and $x_6 = 1$ is 2 (see Figure 2b). For the lower bound, we obtain a value of 7, which is smaller than the previous value of 8. That is, this branch is explained by the previous constraints. However, structurally, the minimum cost of the operating units needed to produce C is 2, which is the minimum in the $y_4 = 1$ and $y_5 = 1$ case (see Figure 2c). Then the installation cost of $1 + 1$ is added to $5 + 2$. So, in total, the lower bound is 9, which is already worse than 8. With this new lower bound, the ABB algorithm is not explained this case.

The third branch includes both the O_1 and O_2 operating units. In this case, the considered inputs of the included units, A and C will be the new materials to be produced. The recursively called ABB method with modified fix costs for the C material, as it has been calculated in the second branch, returns 2. The optimal relaxed operational configuration for the current whole branch is $x_1 = 1$. The sum with the fixed cost of the included units will be 11 so it is fathomed.



(a) Original P-Graph (b) Optimal proportional cost (c) Optimal fix cost

Figure 2: A simple P-Graph in which the fixed and proportional costs are given above the operating units.

4 Results

To test our new lower bound calculating algorithm running time, we have to generate some test cases. There are many aspects to examine the performance of our new development. It is obvious that the CPU time could be taken into consideration but in this case teh solution of the LP relaxation problem should be added to the computational time. In a further development, it is not fixed whether we use an efficient LP solver method or use LP at all. The relationship between the produced and consumed materials does not have to be linear. It could be nonlinear as well. Because of these reasons, we preferably examine the total number of LP_solver executions and not the actual CPU time.

4.1 Result for P-Graphs without circle cases

Firstly, test cases have to be generated according to some predefined metrics to compare the number of LP_solvers calls in the case of ABB called with the $Lower\_Bound_{relaxed}$ and $Lower\_Bound_{new}$. The metrics in the first case will be the height and the width of the P-Graph. The whole structure of the graph will be a ($height \times width$) matrix of the operating units. Before the first level of operating units, there is a level of width the number of raw materials. Between the operating unit levels, there are also material levels which consist only of the width number of intermediate materials, and after the last level of operating units, there is also the width number of products.

In each n^{th} operating unit level ($n \in \{1, \ldots, weight\}$) operating unit consumes at least one material from the n^{th} material level. In the n^{th} material level ($n \in \{1, \ldots, weight\}$) all the materials consumed by at least one operating unit from the n^{th} operating unit level.

The same is true for the produced materials by the operating units in the n^{th} level: each n^{th} operating unit produces at least one material from the $(n+1)^{th}$ material level and each material from the $(n+1)^{th}$ level is produced at least by one operating unit from the n^{th} level. Alongside these connections listed above all the following layers' nodes (the n^{th} operating unit layer with the n^{th} and $(n+1)^{th}$ material layers) are connected with p probability.

In Figure 4's first plot this kind of P-Graph is illustrated. It is obvious that by running the ABB algorithm over these classes of P-Graphs the MSG algorithm won't exclude any operating unit from the original graph. It is also evident that the graph doesn't contain any operating unit circle because it doesn't hold edges that point to previous layers' elements.

The experiment of running the ABB algorithm and calculating the average number of LP solver calls according to the lower bound methods was completed, and the results are summarized and plotted in Figure 3.

The results are grouped by height. If we examine the branching method from the ABB algorithm, then it is straightforward that the algorithm has to examine all the possible operating unit combinations which can produce all the $x \in P$ material. The cutting can't be done on the first level of the tree, all the sub-branches have to

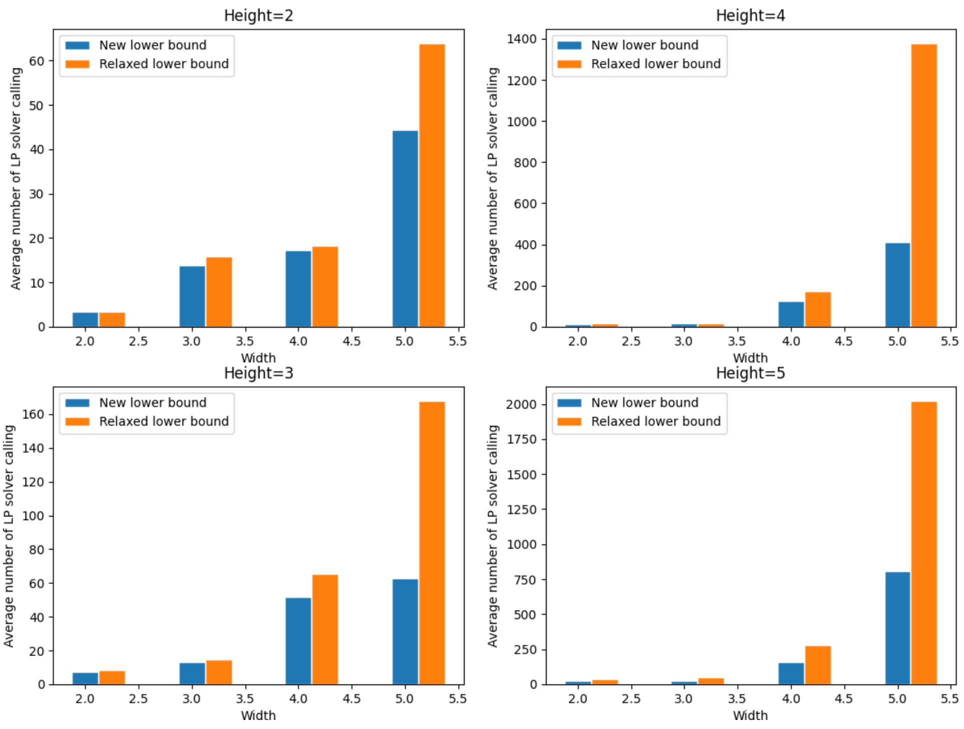
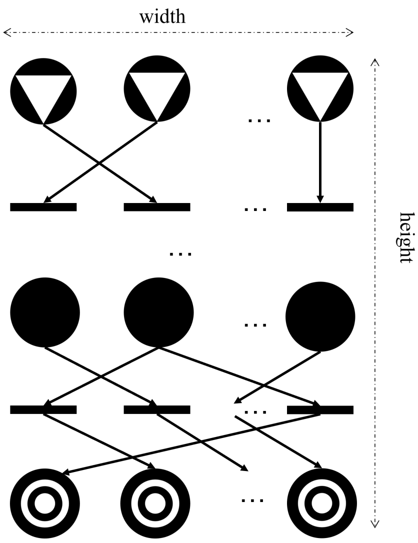
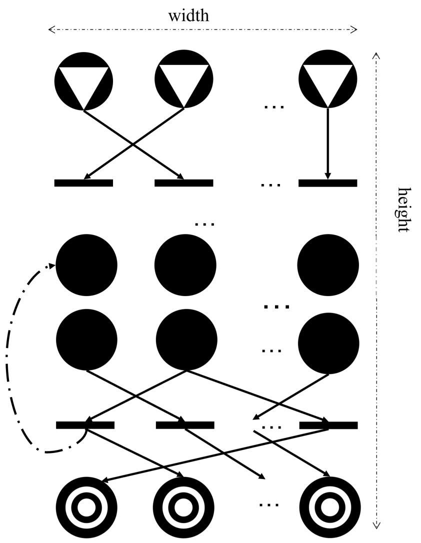


Figure 3: Comparing the ABB algorithm with different lower bounds by the average number of executed LP-solver calls over 30 generated P-Graphs without circles.



(a) A general example for the P-Graph without circles with the parameter setting $p = 0.5$



(b) A general example for the P-Graph with circles with the parameter setting $p = 0.5$ and $p_{circ} = 0.1$

Figure 4: Examples for the randomly generated P-Graphs

be calculated for the cutting, which is in this case one of the first chosen product production with all the possible cases. The possibility of an operating unit in the n^{th} level producing the $x \in P$ material is constant p beside one operating unit which surely can produce it. It implies that $\mathbb{E}(|\Delta(x)|) = p * width + 1$. It means that beside the same heights, the number of the branch grows as $\Omega(2^{width})$ with any lower bound.

As the $Lower\_Bound_{new}$ gives a tighter lower bound than the relaxed type of lower bound with the same height and width, $Lower\_Bound_{new}$ always outperforms the $Lower\_Bound_{relaxed}$. It implies that in all cases the number of LP_{solver} function calling in the ABB algorithm is always smaller with the $Lower\_Bound_{new}$ lower bound, than with the $Lower\_Bound_{relaxed}$. The reason for it is that the calculation of the structural optimization of the free operating units in our new lower bound does not call the LP_{solver} method, and the solver is called exactly once in the lower bound method. It is also true for the relaxed version of the lower bound.

If the submethod gives a tighter lower bound for the actual optimal value then the method would be called fewer times. It also means that the LP_{solver} is called fewer times.

4.2 Results for the P-Graphs with circle cases

Considering the previous generated test cases, the P-Graph with circle cases hasn't been examined yet. The cases with circles are when the previous P-Graph includes some directed edges that point to the previous levels. For example, there is an operating unit in the n^{th} level which produces at least one material in the $m^{th}(m \leq n)$ level with the probability p_{circ}. It is easy to prove that a dependency chain can be formulated when an infinite number of materials should be produced. An example of the generation logic is shown in Figure 4's second plot. These chains evolved when there is no raw materials being part of the chain, just products, and intermediate materials. As a consequence, the chain will be excluded from the graph when the MSG algorithm is running over the $PNS(P, R, O)$ problem. The executions' results with the parameter settings are $p_{circ} = 0.1$ and $p = 0.5$. This means that a portion of ≈ 0.1 is erased from the P-Graph while the MSG algorithm has been running over. The randomly generated summed average LP solver calling results grouped by the height and width is demonstrated in Figure 5. Similar to the previous non-circle running comparison, the average number of calls grows exponentially by increasing the width of the graph. The growth is distorted by the fact that some operating units are erased randomly because of the circle cases.

4.3 Result for a specific P-Graph

To test the new lower bound performance on a specific graph, the graph depicted in Figure 6 is used. The concrete P-Graph is also the maximal structure, as the MSG algorithm running over it doesn't exclude any operating unit. The graph has 65 materials, and 35 operating units. The ABB algorithm has been run over the graph both with our new lower bound, and the relaxed lower bound. The comparison of the performances is listed in Table 1. To make an extended comparison, the fix and operating costs are multiplied with constant values. The constant values are different for the cost types. The first column contains these values. The second column contains the summed LP_Solver calling number during the running of the ABB algorithm. If all the constants are 0 then it equals to the case when there is no costs. In this case the lower bound sub method can't cut on any branch at all, because the branch-and-bound algorithms, also like the ABB, always take advantage of the costs when a cutting is performed. As it is listed in the table's first row, all the possible structures are examined. If the fixed cost is multiplied by 0 and the operational cost by 1, then the ABB algorithm could prune just with the optimal summed operating cost which is calculated by the LP_Solver in both lower bounds.

The two constants which are used to multiply the costs control that algorithm ABB($PNS_{IP}^{\delta[m]}(P', R', O')$, $Summed\_Weight$) or $LP_{Solver}(PNS(P, R, O), O_I, O_E)$ is more dominant than the fix cost or op cost multiplier was increased in the new lower bound. As the table shows, the basic case is when the costs are not multiplied then with the relaxed bound the number of LP calls is 48, and with the new lower

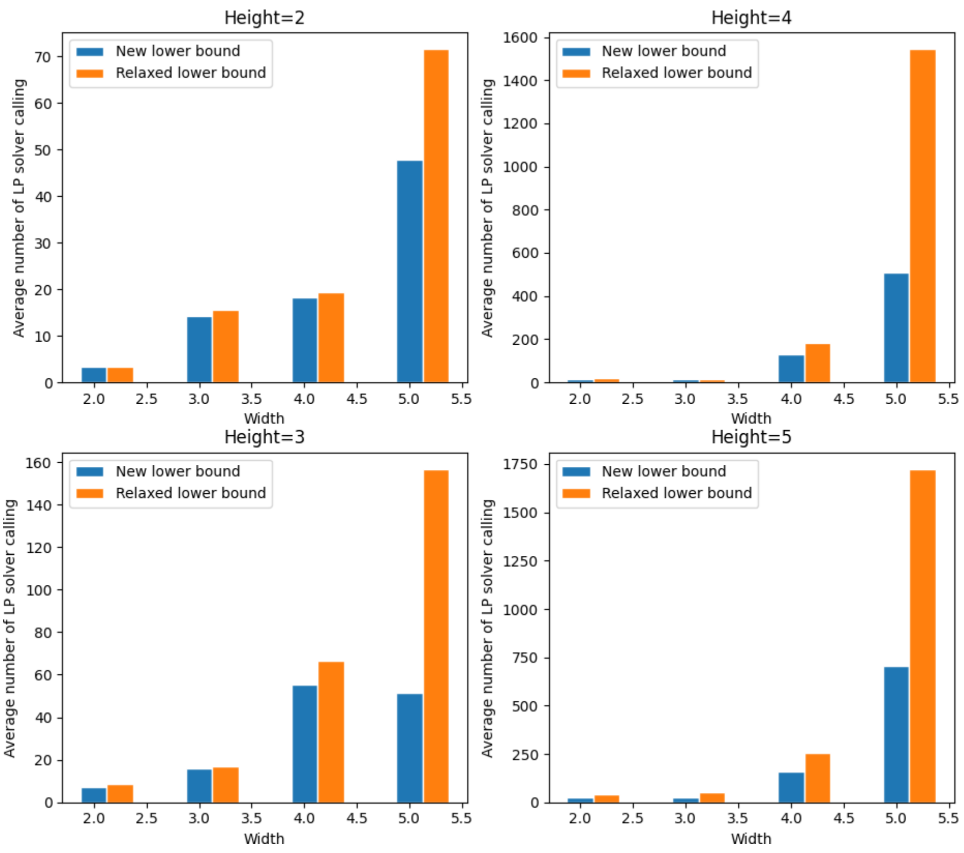


Figure 5: Comparing the ABB algorithm with different lower bounds by the average executed LP-solver numbers over 30 generated P-Graphs with circles, where $p_{circ} = 0.1$ and $p = 0.5$.

bound the number of calls is 31. That means the number of calls is reduced by $\approx 35\%$ with our new lower bound. With increasing the operational multiplier constant compared to the fix cost multiplier as the fix cost remains the same, but the operational cost is increased by the multiplication of $10^1, 10^2, \ldots, 10^5$ the number of LP calls remains the same. It has happened because the P-Graph costs aren't modified to the level when the differences between the structural cost and operational cost are so large that the algorithm could cut the branches according to the operational cost. If the operational cost would be significantly larger compared to the structural cost, the LP calls with the relaxed and the new lower bound converge to the same value, 1023.

If the fix cost multiplier constant is 1, and the operational cost multiplier is 0 (plotted in the last row in the table) then the free operating units' summed cost is 0. As a consequence of that, the branch can be cut down by the summed cost of

Table 1: Comparison of the running of ABB algorithm with different lower bound functions: in the first columns are the results of running with relaxed LP model bound, in the second columns are the results with our new lower bound.

Ratio	LP		Time		Comparing the LP ratio
(fix cost/ op. cost)	Relaxed bound	New lower bound	Relaxed bound	New lower bound	(New lower /Relaxed)
0/0	2099	2099	13478	24532	1
0/1	1023	1023	7271	19857	1
$1/10^{1\ldots5}$	48	31	361.6	464.8	0.65
1/1	48	31	352	460	0.65
$10^1/1$	48	31	378	475	0.65
$10^2/1$	44	27	420	453	0.61
$10^3/1$	91	60	837	948	0.66
$10^4/1$	95	70	763	1037	0.74
$10^5/1$	83	57	669	835	0.69
1/0	80	57	577	740	0.71

already included operating units' total cost added to the remained graph optimal structural cost. Then due to this, the relaxed bound functions as $Summed\_Weight$ (Algorithm 4) bound and the newly introduced lower bound uses also the remained part's optimal structural cost for the cut branch. This case is interesting when the structure was to be optimized, and the network wasn't used, or only rarely [1, 2].

In the previous rows in the table the cases are examined when the fix cost is multiplied by $10, 100, \ldots 10^5$. In these cases the results converge to the case, when the fix cost has remained the same, and the operating cost was erased.

From the examined cases in the table, the $10^2/1$ case gives the optimal number of LP solver calls with both kinds of lower bounds, and also the LP ratio is the smallest in this case. The 1/0 case doesn't give the optimal ratio. It is caused by the acceleration of the algorithm with the natural extension.

The next column in the table lists the running time of the algorithm in CPU seconds multiplied by 1000. Examining this column shows, that the algorithm with a relaxed lower bound gives almost always a better result than the running with the new lower bound. It is the case since it is not worth to call the LP solver fewer times as the calculation of free operating units' optimal cost is more time-consuming. It isn't the case if the calculation of the derived model is more time-consuming. An example is when the P-Graph isn't linear.

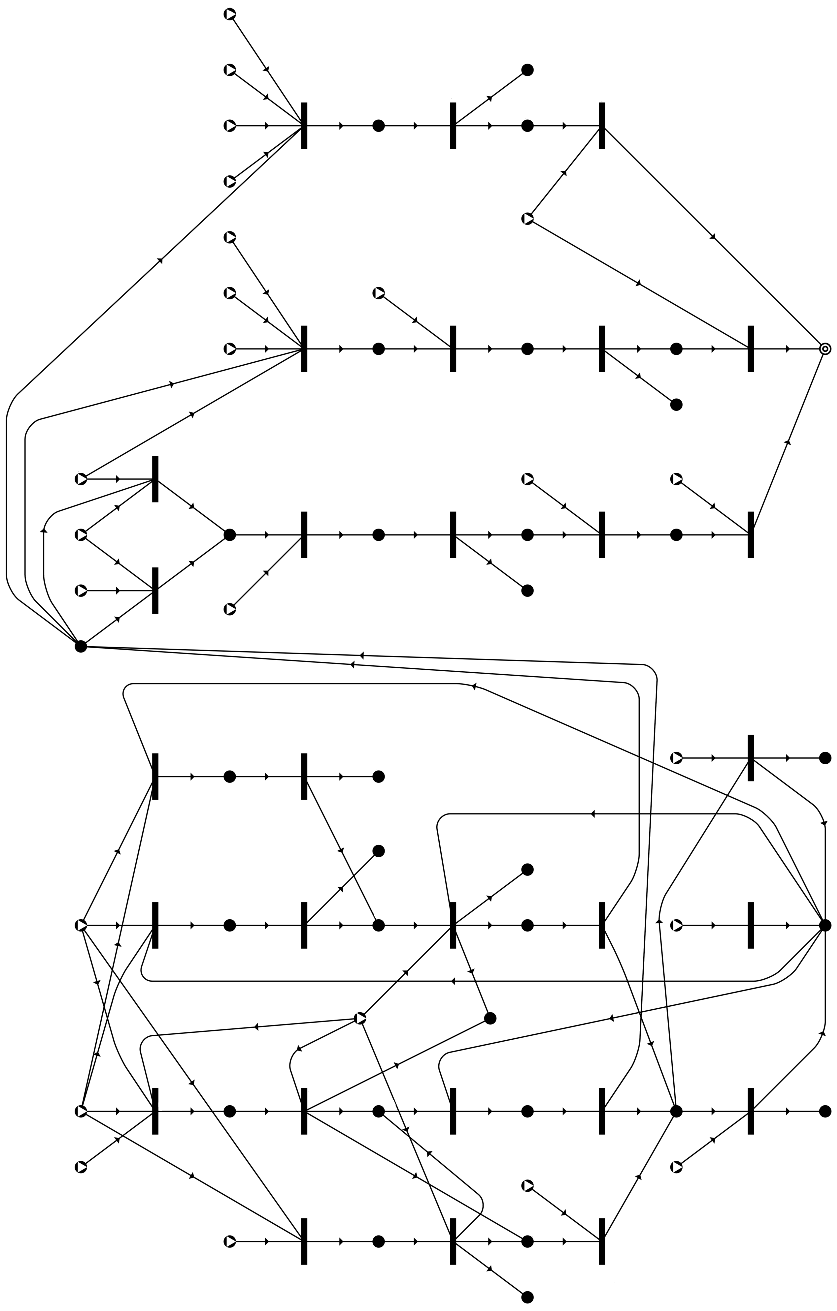


Figure 6: A specific graph with 65 materials and 35 operating units. The graph has been generated with the MSG algorithm.

5 Conclusion

We have created a new constraint calculation procedure for the ABB algorithm that gives a better one than the previously applied constraints. The computational cost of the new constraint is higher than the previous one, but we show that the overall number of linear programming problem solver calls will be reduced. It is advantageous to run the derived problem solver fewer times if in the operating unit model, the connection between the consumed and produced materials is nonlinear, or the variables are stochastic.

References

[1] Aviso, K., Lee, J.-Y., Dulatre, J., Madria, V., Okusa, J., and Tan, R. A p-graph model for multi-period optimization of sustainable energy systems. *Journal of Cleaner Production*, 161:1338–1351, 2017. DOI: 10.1016/j.jclepro.2017.06.044.

[2] Aviso, K., Yu, K., Lee, J.-Y., and Tan, R. P-graph optimization of energy crisis response in Leontief systems with partial substitution. *Cleaner Engineering and Technology*, 9:100510, 2022. DOI: 10.1016/j.clet.2022.100510.

[3] Berthold, T., Farmer, J., Heinz, S., and Perregaard, M. Parallelization of the FICO Xpress-Optimizer. *Optimization Methods and Software*, 33(3):518–529, 2018. DOI: 10.1080/10556788.2017.1333612.

[4] CPLEX, I. I. V12. 1: User's manual for CPLEX. *International Business Machines Corporation*, 46(53):157, 2009. URL: https://public.dhe.ibm.com/software/websphere/ilog/docs/optimization/cplex/ps_usrmancplex.pdf.

[5] Friedler, F., Orosz, ., and Losada, J. *P-graphs for Process Systems Engineering*. Springer Cham, London, 2022. URL: https://link.springer.com/book/10.1007/978-3-030-92216-0.

[6] Friedler, F., Tarjan, K., Huang, Y., and Fan, L. Combinatorial algorithms for process synthesis. *Computers and Chemical Engineering*, 16:S313–S320, 1992. DOI: 10.1016/S0098-1354(09)80037-9.

[7] Friedler, F., Tarjan, K., Huang, Y., and Fan, L. Graph-theoretic approach to process synthesis: Polynomial algorithm for maximal structure generation. *Computers and Chemical Engineering*, 17(9):929–942, 1993. DOI: 10.1016/0098-1354(93)80074-W.

[8] Friedler, F., Tarján, K., Huang, Y., and Fan, L. Graph-theoretic approach to process synthesis: axioms and theorems. *Chemical Engineering Science*, 47(8):1973–1988, 1992. DOI: 10.1016/0009-2509(92)80315-4.

[9] Friedler, F., Varga, J., and Fan, L. Decision-mapping: A tool for consistent and complete decisions in process synthesis. *Chemical Engineering Science*, 50(11):1755–1768, 1995. DOI: 10.1016/0009-2509(95)00034-3.

[10] Friedler, F., Varga, J., Fehér, E., and Fan, L. Combinatorially accelerated Branch-and-Bound method for solving the MIP model of Process Network Synthesis. In Floudas, C. and Pardalos, P., editors, *State of the Art in Global Optimization: Computational Methods and Applications*, pages 609–626, Boston, MA, 1996. Springer US. DOI: 10.1007/978-1-4613-3437-8_35.

[11] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022. URL: https://www.gurobi.com/documentation/current/refman/index.html.

[12] Nishida, N., Stephanopoulos, G., and Westerberg, A. A review of process synthesis. *AIChE Journal*, 27(3):321–351, 1981. DOI: 10.1002/aic.690270302.

[13] Siirola, J. Industrial applications of chemical process synthesis. *Advances in Chemical Engineering*, 23:1–62, 1996. DOI: 10.1016/S0065-2377(08)60201-X.