Empirical Analysis of Some Procedures for Solving Process Network Synthesis Problem*

B. Imreh[1] and G. Magyar[2]

[1] Department of Informatics, József Attila University, Szeged, Hungary
[2] Turku Centre for Computer Science (TUCS) and Department of Computer Science, University of Turku, DataCity, Turku, Finland

Process network synthesis (PNS) has an enormous practical impact and a structural model can be given for it on the basis of a combinatorial approach. An optimal solution to this model can be generated by different branch-and-bound procedures. In the present work, we compare such procedures empirically.

Introduction

The mixed integer programming model of process network synthesis (PNS) contains a large number of binary variables associated with operating units which renders this model difficult to solve by any available method. On the other hand, a combinatorial analysis of the MIP models of PNS and that of feasible process structures have yielded mathematical tools which provide a good background for obtaining an optimal solution by a branch-and-bound search.

The present work demonstrates that the size of a PNS problem can be reduced in general, and compares empirically some branch-and-bound procedures for solving PNS problem under the assumption that each operating unit has a positive fixed charge in the object function. Firstly, three procedures, that of a conventional branch-and-bound procedure, the accelerated branch-and-bound algorithm (ABBA, see [11]), and a modified version of the ABBA (see [14]) are compared under the same bounding function, i.e., the three procedures being compared differ only in the branching strategy applied. After this, the modified ABBA, the modified ABBA equipped with a stronger bounding function (cf. [14]), and an adaptation of the latter procedure are compared. In this case, the different procedures differ only in the bounding functions.

The paper is organized as follows: Section 1 introduces the PNS problem and some relevant basic concepts. In Section 2, two classes of PNS problems involved in the empirical analysis are described, and the measure of the reduction is demonstrated. Section 3 presents the above mentioned different branch-and-bound methods for solving PNS problem. Section 4 contains the results of our computational studies. Finally, Section 5 presents some conclusions based on the empirical analysis.

1. Preliminaries

In a manufacturing system, materials of different properties are converted into desired products through various physical, chemical, and biological transformations. Devices in which these transformations are carried out are called *operating units* and a manufacturing system can be considered as a network of operating units, i.e., *process network*. Naturally, minimizing the cost of a process network is essential indeed. There are several papers on the application of global optimization methods for solving PNS

* This work was partially supported by the Hungarian Ministry of Education, Grant 635/96.

problems (see, e.g., [3] and [13]). Here, we use a combinatorial approach based on the feasible graphs of processes (cf. [5], [6], and [7]). In this approach, an operating unit denoted by u can be given by its input and output materials, and its behaviour can be described by such a directed graph in which edges lead from every input material into u and from u into every output material. Joining the graphs belonging to the given operating units, we obtain a network of operating units which is called a *process graph* or P-*graph* in short.

A *design problem* $\mathbf{M}$ is defined from structural point of view by the *raw materials*, the *desired products*, and the available operating units. The available operating units determine the structure of the problem as a process graph containing the corresponding interconnections among the operating units. Let us denote this P-graph by (M, O). A subgraph of (M, O), which is also a P-graph, can be assigned to each feasible process of the design problem; such a subgraph represents the structure or network of the process under consideration. If additional constraints like the material balance are disregarded, the subgraphs of (M, O), which can be assigned to the feasible processes, have some characteristic combinatorial properties (see [6]). The subgraphs of (M, O) satisfying these properties are called *solution-structures*. Let us denote the set of solution-structures of $\mathbf{M}$ by $S(\mathbf{M})$. It can be seen that $S(\mathbf{M})$ is closed under the finite union. Thus, the union of all the solution-structures is also a solution-structure provided that $S(\mathbf{M}) \neq \emptyset$; it is the greatest solution-structure with respect to the relation, subgraph ordering, which is called the *maximal structure* of $\mathbf{M}$.

Now, a class of optimization problems can be defined such that each operating unit has a positive fixed charge. We are to find a feasible process with the minimum cost; by the cost of a process, we mean the sum of fixed charges of the operating units belonging to the process under study. Each feasible process in this class is uniquely determined from the corresponding solution-structure and vice versa. Hence, the problem under consideration can be formalized as follows.

Let a design problem $\mathbf{M}$ be given; moreover, let z be a positive real-valued function defined on O. The optimization problem is then

$$\min\Big\{\sum_{u\in o} z(u) : (m, o) \in S(\mathbf{M})\Big\}. \qquad (1)$$

Problem (1) is called PNS *problem*. It has been proved [2] that this problem is NP-hard; therefore, the branch-and-bound technique may be an appropriate tool for its solution. Prior to presenting our branch-and-bound procedures, let us introduce several notions.

Let $\mathbf{M}$ be a design problem with $S(\mathbf{M}) \neq \emptyset$. Then, $\mathbf{M}$ has a maximal structure denoted by (M', O'). Now, we can construct a new design problem $\mathbf{M}'$, called *reduced design problem*, by leaving out the unnecessary raw materials and reducing the available operating units to O'. Obviously, $\mathbf{M}'$ is uniquely determined and its maximal structure is also (M', O'); moreover, $S(\mathbf{M}) = S(\mathbf{M}')$. An efficient algorithm (cf. [6] and [10]) is available for deciding if $S(\mathbf{M}) = \emptyset$; if it is not, this algorithm generates the corresponding maximal structure. Consequently, in what follows, we confine ourselves strictly to reduced design problems.

Now, a new concept, that of decision-mapping (see [12]), is introduced. Let $\mathbf{M}$ be a reduced design problem and denote by R the set of the raw materials. Then, the maximal structure, (M, O), of $\mathbf{M}$ determines a function Δ as follows. For any material $X \in M \setminus R$, let $\Delta(X)$ be the set of the operating units in O producing X. It is worth noting that one of the defining properties of the solution-structures implies $\Delta(X) \neq \emptyset$. Let m be a subset of M; furthermore, let $\delta(X)$ be a subset of $\Delta(X)$, for all $X \in m$. This mapping denoted by $\delta[m]$ is called a *decision-mapping belonging* to $\mathbf{M}$. A decisison-mapping can be visualized as a sequence of decisions, each of which is concerned with a single material involved in the process being synthesized; it identifies the set of operating units which are to be used for producing directly the material under consideration. Decisions contained in such a sequence are independent, and each decision is material-specific. Suppose that we have made two decisions independently: One is to produce material X and the other is not to produce material Y in a single process by adopting an operating unit generating both materials X and Y. Obviously, these two independent decisions are contradictory. To circumvent such a contradiction, decision-mappings

must be consistent. Hence, a special class of decision-mappings need be introduced for this purpose. Decision-mapping $\delta[m]$ is said to be *consistent* when $\delta(X) \cap \Delta(Y) \subseteq \delta(Y)$ is valid, for all $X, Y \in m$, and the set of all consistent decision-mappings of $\mathbf{M}$ is denoted by $\Omega_{\mathbf{M}}$. In particular, if $\delta[m] \in \Omega_{\mathbf{M}}$ and $m = M \setminus R$, then we sometimes use the shorter notation δ instead of $\delta[M \setminus R]$. Let us denote by $op(\delta[m])$ the function giving the set of operating units contained in any set $\delta(X)$ for some $X \in m$. Furthermore, for any finite set of operating units o, let us denote by $mat^{in}(o)$ the set of the input materials of the operating units in o, by $mat^{out}(o)$ the set of the output materials of the operating units in o, finally, let $mat(o) = mat^{in}(o) \cup mat^{out}(o)$.

Relation extension on $\Omega_{\mathbf{M}}$ is reflexive, antisymmetric and transitive; hence, it is a partial ordering on $\Omega_{\mathbf{M}}$. Let us denote the set of all maximal elements of this partially ordered set by $\Omega'_{\mathbf{M}}$. Then, the following statement is valid:

Lemma 1. *If $\delta[m] \in \Omega_{\mathbf{M}} \setminus \Omega'_{\mathbf{M}}$, then $\delta[m]$ has at least one proper extension.*

A strong relationship exists between some elements of $\Omega'_{\mathbf{M}}$ and the solution-structures of $\mathbf{M}$. Namely, there is an injective mapping ρ of $S(\mathbf{M})$ into $\Omega_{\mathbf{M}'}$. Let $S'(\mathbf{M})$ denote the image of $S(\mathbf{M})$ under ρ. The elemes ntof $S'(\mathbf{M})$ are called *feasible solutions*. (Note that ρ is not surjective in general.)

Using this relationship, it is possible to solve the problem stated below instead of problem (1)

$$\min\Big\{ \sum_{u \in op(\delta)} z(u) : \delta \in S'(\mathbf{M}) \Big\}. \qquad (2)$$

Function z can be defined on $\Omega_{\mathbf{M}}$ as follows: for any $\delta[m] \in \Omega_{\mathbf{M}}$, let $z(\delta[m]) = \sum_{u \in op(\delta[m])} z(u)$. To solve problem (2), the following observation is important: since $S'(\mathbf{M}) \subseteq \Omega'_{\mathbf{M}}$, any partition of $\Omega'_{\mathbf{M}}$ induces a partition of $S'(\mathbf{M})$. The branch-and-bound procedures presented in Section 3 are based on this observation.

2. Some classes of PNS problems

In this section, such classes of PNS problems are described which are involved in the empirical analysis outlined below. To generate realistic S PNproblems randomly, we have investigated some practical applications published in the works [4], [8], [9], and [10]. Analyzing these applications, the generation of one class of problems under a fixed number of materials denoted by n is established according to the following rules:

(a) The number of raw materials is $0.35n$.

(b) The number of desired products is fixed to 1.

(c) The number of operating units is $m = 0.6n$.

(d) Three types of operating units are distinguished:

(i) *Operating units producing the desired product.* Their number is $j = 0.1m$. Each of them has one output material which is the desired product, 50% have two input materials while the remaining 50% have three input materials. None of the input materials of this type of operating units is a raw material; moreover, the desired product cannot appear as an input materia l.

(ii) *Operating units consuming raw material.* Their number is $k = 0.2m$. This group of operating units is divided into two subgroups; the *homogeneous* and *inhomogeneous operating units*. For each of the homogeneous operating units, all input materials are raw materials and each of these operating units has only one output material which is not the desired product. The *inhomogeneous operating units* have one no raw material input differing from the desired product and each of the remaining input materials is a raw material. Their output materials differ from the desired product. The number of homogeneous operating units is $0.5k$, their distribution with respect to the number of input materials being presented in Table I.

Similarly, the number of inhomogeneous operating units is $0.5k$, their distribution with respect

Table I

number of input materials	1	2	3	4
operating units %	10	40	40	10

Table II

inputs/outputs	2/2	2/3	3/2
operating units %	50	20	30

Table III

inputs/outputs	1/1	1/2	1/3	1/4	2/1	2/2	2/3
operating units %	10	50	15	5	5	10	5

to the ratio of the input and output materials being given in Table II.

(iii) *Operating units consuming no raw material.* The input and output materials of these operating units differ from both the desired product and raw materials. Their number is $l = 0.7m$ and their distribution with respect to the ratio of the input and output materials is presented by Table III.

(e) The fixed charges of the operating units are independently drawn from the uniform distribution of the integers over interval [1, 100].

To increase the complexity of the above class of PNS problems, we have investigated another class of PNS problems which can be derived from the above class with the following modifications:

(1) The number of desired products is fixed to 2.

(2) The number of operating units is $m = 0.65n$,

(3) The number of operating units producing one of the desired products is $j = 0.2m$.

(4) The number of operating units consuming raw materials is $k = 0.3m$.

(5) The number of operating units consuming no raw material is $l = 0.5m$.

The input and output materials for the operating units of this class were generated according to Tables I, II, and III, respectively. To distinguish the two classes, the former one is called *class A* and the latter one is called *class B*.

We generated randomly 10000 PNS problems in accordance with the above rules and distributions for both classes and for all n, $n = 20, 30, \ldots, 150$. The computational results are summarized in Tables IV and V for the classes A and B, respectively. For each problem, the reduced design problem was determined by the reduction procedure from [10]. For each group of PNS problems belonging to a fixed n,

- the average of the numbers of the materials included in the reduced problems,
- the average of the numbers of the operating units included in the reduced problems,
- the average of the ratios, number of operating units/number of materials, with respect to the reduced problems

Table IV

n	20	30	40	50	60	70	80
unsolvable	4169	3027	2866	1693	1718	1427	1087
solvable%	58.31	69.73	71.34	83.07	82.82	85.73	89.13
materials	12.84	18.21	23.14	27.39	32.33	36.97	40.93
operating units	9.07	12.19	15.72	18.22	20.63	24.19	26.08
$\|O'\|/\|M'\|$ (%)	70.63	66.96	67.97	66.55	63.81	65.42	63.70

n	90	100	110	120	130	140	150
unsolvable	981	812	607	570	444	451	320
solvable%	90.19	91.88	93.93	94.30	95.56	95.49	96.80
materials	45.89	49.66	54.55	58.89	63.40	67.54	71.85
operating units	29.96	31.51	34.43	37.45	39.84	42.75	45.23
$\|O'\|/\|M'\|$ (%)	65.28	63.46	63.10	63.59	62.84	63.30	62.95

Table V

n	20	30	40	50	60	70	80
unsolvable	7252	5984	5443	3649	3730	2793	2645
solvable%	27.48	40.16	45.57	63.51	62.70	72.07	73.55
materials	15.50	22.25	27.45	32.73	37.65	43.33	47.94
operating units	10.82	15.87	18.29	21.93	24.21	28.29	30.61
$\lvert O'\rvert/\lvert M'\rvert$ (%)	69.78	71.29	66.62	67.00	67.00	65.30	63.86

n	90	100	110	120	130	140	150
unsolvable	2362	1970	1447	1356	1070	1016	797
solvable%	76.38	80.30	85.53	86.44	89.30	89.84	92.03
materials	51.61	56.43	62.53	65.59	71.49	75.35	80.36
operating units	32.26	35.23	39.42	40.41	44.45	46.16	49.41
$\lvert O'\rvert/\lvert M'\rvert$ (%)	62.51	62.44	63.03	61.62	62.18	61.27	61.48

were calculated. When a problem of a group under consideration had no solution-structure, then it was not included in the average, and the number of such problems of the corresponding group were calculated as well; the second rows of Tables IV and V show these values, while the third rows give the numbers of the solvable problems in percent. The fourth rows present the average of the numbers of materials, the fifth rows give the average of the numbers of operating units, finally, the sixth rows contain the average of the ratios, number of operating units/number of materials, regarding the reduced models, respectively.

3. Branch-and-bound procedures

In this chapter, different branch-and-bound algorithms are constructed to solve (2). The branch-and-bound method has been widely published in the literature (see e.g. [1], [15]). The algorithms presented here are particular ones based on special features of PNS problems. To give their descriptions, let $\mathbf{M}$ be a reduced design problem with the maximal structure (M, O) and let z be the object function. Our goal is to solve problem (2). Since $S'(\mathbf{M})$ is a nonempty finite set, there should exist at least one optimal solution for (2).

One of the main components of a branch-and-bound algorithm is the branching rule which divides some subsets of $S'(\mathbf{M})$ into their partitions. Here, $S'(\mathbf{M})$ is given in an implicit way and this makes it difficult to give an appropriate branching rule. This difficulty can be eliminated by the including set, $\Omega'_{\mathbf{M}}$, which can be more easily treated than $S'(\mathbf{M})$. In general, the *branching rule* or *branching operation* can be defined as a mapping which assigns to some nonempty subsets of $\Omega'_{\mathbf{M}}$ their, not necessarily nontrivial partitions. Here, a more general function is applied. To define it, let $\delta[m] \in \Omega_{\mathbf{M}} \setminus \Omega'_{\mathbf{M}}$. Then, let us define set $\omega(\delta[m])$ as follows:

$$\omega(\delta[m]) = \{\delta : \delta \in \Omega'_{\mathbf{M}} \text{ and } \delta[m] \leq \delta\},$$

i.e., $\omega(\delta[m])$ is the set of all consistent maximal extensions of $\delta[m]$. From Lemma 1, $\omega(\delta[m])$ is nonempty. It is worth noting that $\omega(\delta[\emptyset]) = \Omega'_{\mathbf{M}}$. The elements of set $\omega(\delta[m]) \cap S'(\mathbf{M})$ are called the *feasible solution extensions* of $\delta[m]$. The sets assigned to different decision-mappings from $\Omega_{\mathbf{M}} \setminus \Omega'_{\mathbf{M}}$ by ω are disjoint, provided they differ in at least one common material. This observation is presented by the following statement.

Lemma 2. *If* $\delta[m], \delta'[m'] \in \Omega_{\mathbf{M}} \setminus \Omega'_{\mathbf{M}}$ *and* $\delta(X) \neq \delta'(X)$ *for some* $X \in m \cap m'$, *then* $\omega(\delta[m]) \cap \omega(\delta'[m']) = \emptyset$.

In the light of Lemma 2, we can define a partition of $\omega(\delta[m])$ for any $\delta[m] \in \Omega_{\mathbf{M}} \setminus \Omega'_{\mathbf{M}}$ as follows: Since $\delta[m] \notin \Omega'_{\mathbf{M}}$, $m \neq M \setminus R$. Moreover, $m \subseteq M \setminus R$ implies that there exists an X in set $M \setminus (R \cup m)$. Since (M, O) is the maximal structure, $\Delta(X) \neq \emptyset$. For each subset Q_i of $\Delta(X)$, $i = 1, 2, \ldots, 2^{|\Delta(X)|}$, the decision-mapping $\delta_i[m \cup \{X\}]$ can be defined by

$$\delta_i[m \cup \{X\}] = \delta[m] \cup \{(X, Q_i)\}.$$

Lemma 1 indicates that at least one consistent decision-mapping exists among $\delta_i[m \cup \{X\}]$, $i = 1, 2, \ldots, 2^{|\Delta(X)|}$. Let us denote the consistent members of the decision-mappings by $\delta_{i_t}[m \cup \{X\}]$, $t = 1, 2, \ldots, k$. Without loss of generality, i_t can be replaced by t for $t = 1, 2, \ldots, k$. Then, the following theorem can be stated on the partitioning of $\omega(\delta[m])$:

Theorem 1. *The sets, $\omega(\delta_t[m \cup \{X\}])$, $t = 1, 2, \ldots, k$, constitute a (not necessarily non-trivial) partition of $\omega(\delta[m])$.*

Theorem 1, together with an additional rule on material selection, establishes a *branching rule* or *branching operation* denoted by φ, i.e.,

$$\varphi(\delta[m]) = \{\omega(\delta_t[m \cup \{X\}]) : t = 1, 2, \ldots, k\}.$$

The other main component of a branch-and-bound procedure is the *bounding rule* or *bounding operation*. In our case, it is defined as a real-valued function denoted by g on set $\Omega_{\mathbf{M}}$ which gives a lower bound of values $z(\delta')$, $\delta' \in S'(\mathbf{M}) \cap \omega(\delta[m])$, for all $\delta[m] \in \Omega_{\mathbf{M}} \setminus \Omega'_{\mathbf{M}}$, while it provides value $z(\delta)$ if $\delta[m] \in S'(\mathbf{M})$, and ∞ if $\delta[m] \in \Omega'_{\mathbf{M}} \setminus S'(\mathbf{M})$.

A branch-and-bound algorithm can be considered as a special intelligent enumerating procedure based on an enumeration tree. Its intelligence means that by using the bounding operation, the algorithm does not necessarily build up the whole enumeration tree, but only a part of this tree. Specifically, it does not branch the classes of the partition determined by the leaves of the actual tree for which it is known that they do not contain feasible solutions or they do not contain better feasible solutions than a known one. A class satisfying one of the last two conditions is called *fathomed* or *dead leaf*; otherwise, it is referred to as a *live leaf*. Now, we are ready to build up a branch-and-bound algorithm to solve (2).

Branch-and-Bound Algorithm

Initialization

- Let $L := \{\omega(\delta[\emptyset])\}$, $z^* := \infty$, $s := \emptyset$, and set $r := 0$. Calculate $g(\delta[\emptyset])$.

Iteration (r-th iteration)

- *Step 1.* Terminate if $L = \emptyset$; s contains the optimal solution, z^* provides the optimal value. Otherwise, proceed to Step 2 if $r > 0$ and to Step 3 if $r = 0$.
- *Step 2. (Selection.)* Choose an element $\omega(\delta[m])$ from L for which the ratio $g(\delta[m])/|m|$ is minimal. (If there are more candidates with the same value, choose one of them in a random way.) Proceed to Step 3.
- *Step 3. (Branching.)* Form the partition $\varphi(\delta[m])$ of $\omega(\delta[m])$; denote this partition by $\{\omega(\delta_i[m_i]) : i = 1, 2, \ldots, k\}$. Proceed to Step 4.
- *Step 4. (Calculating bounds.)* For each i, $i = 1, 2, \ldots, k$, calculate lower bound $g(\delta_i[m_i])$; furthermore, if $m = M \setminus R$ and $\delta_i \in S'(\mathbf{M})$ and $g(\delta_i) < z^*$, then let $z^* = g(\delta_i)$, $s := \{\delta_i\}$. Proceed to Step 5.
- *Step 5. (Fathoming.)* Redefine set L in the following way:

  $$L := \{\omega(\delta'[m']) : \omega(\delta'[m']) \in (L \setminus \{\omega(\delta[m])\}) \cup \varphi(\delta[m]), g(\delta'[m']) < z^*\}.$$

  Set $r := r + 1$ and proceed to the next iteration.

As far as the introduced procedure is concerned, we note that the selection of the actual leaf in Step 2 is based on the ratio $g(\delta[m])/|m|$ (see [14]). With respect to the leaf selection rule, we carried out more computational experiments to test its efficiency. The classical selection rule based on the minimal bound, some selection rules based on different combinations of the classical selection and the selection based on the ratio $g(\delta[m])/|m|$ were investigated, and we found in every case that the classical strategy was the best one in terms of computational time. On the other hand, the size of the built enumeration tree, and thus the number of live leaves

increased so rapidly under the classical strategy, that it was not practically possible to solve PNS problems of higher complexity. Therefore, we employed the strategy which minimized the ratio $g(\delta[m])/|m|$ because this approach, like a kind of depth-first search, kept the number of live leaves to a moderate value.

Furthermore, it is worth noting that the algorithm above is not complete in the following sense:

(1) In Step 3, the material selection rule (to choose a material from set $M \setminus (R \cup m)$) is not fixed, different rules yield different branching functions.

(2) The bounding operation is not given, only some properties are prescribed.

Therefore, to present a complete procedure, we have to furnish a material selection rule and suitable bounding operation. In what follows, we apply some material selection rules and some bounding functions, then build up different complete branch-and-bound algorithms to solve (2).

We will use three material selection rules. To define them, it is assumed that $M = \{A_1, \ldots, A_n\}$, $P = \{A_1, \ldots, A_v\}$ and $R = \{A_w, \ldots, A_n\}$ for some integers v, w, n with $1 \leq v < w \leq n$ where P denotes the set of the desired products. Now, we are in a position to define our material selection rules for leaf $\omega(\delta[m]) \in L$.

Material selection on index. In this case, the material with the smallest index is chosen from set $M \setminus (m \cup R)$.

Input material selection rule. This means that the material with the smallest index is selected from the set $(mat^{in}(op(\delta[m])) \cup P) \setminus (R \cup m)$. This rule introduced in [11] implies the following property. If we cannot choose a material in accordance with this rule, i.e., the above set is empty, then the leaf $\omega(\delta[m])$ contains one and only one feasible solution which can be determined.

Input but no output material selection rule. Here, the material with the smallest index is chosen from the set $(mat^{in}(op(\delta[m])) \cup P) \setminus (mat^{out}(op(\delta[m])) \cup R)$. This rule introduced in [14] has the following important property. If we cannot choose a material using this rule, in $\omega(\delta[m]) \cap S'(\mathbf{M})$ there is a best feasible solution that can be determined.

Three bounding functions are involved in the investigations. Let us suppose that the leaf under consideration is $\omega(\delta[m])$.

Trivial bounding function. In this case, the sum of the values belonging to the operating units fixed by $\delta[m]$ gives a trivial lower bound. Formally, let

$$g(\delta[m]) = \begin{cases} \sum_{u \in op(\delta)} z(u) & \text{if } \delta[m] \notin \Omega'_{\mathbf{M}} \text{ or} \\ & \delta[m] \in S'(\mathbf{M}), \\ \infty & \text{otherwise.} \end{cases}$$

Refined bounding function. In this case, a stronger bounding procedure is applied. Its precise description is presented in [14]. Here, we only mention the basic idea of this procedure. Concerning the trivial bounding function, at the beginning of the procedure, when a small number of operating units are fixed, there is a large gap between the bound, $g(\delta[m])$, and the minimum value of $z(\bar{\delta})$, $\bar{\delta} \in \omega(\delta[m]) \cap S'(\mathbf{M})$. The deficiency of this bounding function arises from the following fact: when we determine the bound, $g(\delta[m])$, we do not take into account additional operating units which appear in the feasible solution extensions of $\delta[m]$. The refinement of the bounding function is based on this observation. To make up for this, a cost is assigned to each material X in M by an iterative procedure. This cost is a lower bound for the cost of producing X in any feasible solution in $\omega(\delta[m]) \cap S(\mathbf{M}')$. Using these costs, a sharper lower bound can be determined than the bound given by the trivial bounding function.

Modified refined bounding procedure. Since the refined bounding function needs a hard computation, moreover, it does not provide a sharp lower bound, especially when a few operating units are fixed, the bounding procedure is modified in the following way: The original refined bounding function is relaxed, furthermore, we count the cheap trivial bound when the number of fixed operating unit does not exceed 30%. When the number of fixed operating units exceeds 30%, then the bound calculation is altered and the relaxed refined bound is calculated.

Table VI

n	15	20	25	30	40	50	60	70	80	90
Conv.	38.44	354.4	1146	12799	–	–	–	–	–	–
ABBA	0.96	1.81	2.60	3.30	5.59	11.62	19.75	27.65	38.94	52.70
mABBA	1.10	1.52	2.46	2.91	5.01	9.16	18.86	23.54	32.32	41.80

Table VII

n	15	20	25	30	40	50	60	70	80	90
Conv.	270.7	1675	5082	20436	–	–	–	–	–	–
ABBA	8.63	12.12	22.90	25.29	42.71	73.09	100.6	130.1	165.4	188.0
mABBA	5.26	7.57	15.58	16.50	26.70	48.24	77.47	90.70	114.8	130.0

4. Empirical analysis

In this section, we compare several branch-and-bound procedures. In the comparison study, a 133Mhz Pentium PC was used, and the run-times were expressed in units of 0.01sec. In the first comparison, three procedures were compared under the trivial bounding function and various material selection rules. The first procedure called the *conventional procedure* applies the material selection on index, the second called the *Accelerated Branch-and Bound Algorithm* or ABBA for short (see [11]) uses the input material selection rule, while the third called the *modified* ABBA or mABBA for short (see [14]) applies the input but no output material selection rule. For all n, $n = 15, 20, 25, 30, 40, \ldots, 90$, we solved 100 randomly generated PNS problems of class A. Tables VI and VII show the average of the run-times and average of the number of iteration steps, respectively.

In the second comparison, three algorithms were tested. The first is the modified ABBA, the second, denoted by mABBA+g^*, is such a version of the mABBA which uses the refined bounding function, while the third, denoted by mABBA+$\tilde{g}$, is such an alteration of the mABBA which uses the modified refined bounding procedure. For all n, $n = 30, 40, \ldots, 100$, we solved 100 randomly generated PNS problems of the classes considered. For class A, Tables VIII and IX show the average of the run-times and average of the number of iteration steps, respectively. Tables X and XI present the corresponding values for class B.

5. Conclusions

First of all, we have to emphasize that all of the conclusions presented below concern PNS problems belonging to the classes considered.

The results in Tables IV and V show that the number of the unsolvable design problems decreases if the problem size increases, furthermore, this number increases if the operating units have a higher complexity. Another observation that the ratios, number of operating units/number of materials, show a small decrease depending on the size and the average is approximately 63%. This means that the reduction does not yield a large change in this ratio, and therefore, the measure of the reduction can be presented approximately by the decrease of the number of materials. The ratios, number of materials in the reduced model/original number of materials, in percent, show a decrease depending on the size, and the average measure of the reduction is approximately 47% and 38% for class A and class B, respectively, which is a significant reduction.

Regarding the first comparison, the results inTables VI and VII show that on the problem instances tested the conventional procedure is not suitable for problem solving, the run-time increases terribly fast depending on the problem size. As regards the other two procedures, both the run-time and number of iteration steps are significantly better for the mABBA procedure. (We obtained similar results for class B.) The latter observation motivated us to investigate such procedures which use the input but no output material selection rule and different bounding functions. It was established in the second comparison.

Table VIII

n	30	40	50	60	70	80	90	100
mABBA	3.24	8.00	8.19	11.70	22.20	29.23	41.48	59.53
mABBA + g^*	19.51	82.75	138.8	251.7	719.3	851.7	2030	2247
mABBA + $\tilde{g}$	12.87	44.86	39.64	59.46	86.41	136.8	126.4	296.3

Table IX

n	30	40	50	60	70	80	90	100
mABBA	21.83	39.70	39.74	50.22	86.80	109.8	136.5	199.8
mABBA + g^*	7.96	15.21	17.08	19.74	29.10	35.24	41.07	47.53
mABBA + $\tilde{g}$	13.71	25.04	33.12	42.40	79.92	91.55	131.0	173.2

Table X

n	30	40	50	60	70	80	90	100
mABBA	15.21	17.95	60.56	130.9	346.5	528.1	610.2	775.2
mABBA + g^*	81.60	153.8	572.4	1324	2913	6932	10195	16969
mABBA + $\tilde{g}$	58.11	96.39	292.8	564.7	918.8	2004	1763	2867

Table XI

n	30	40	50	60	70	80	90	100
mABBA	82.12	94.97	263.1	419.3	798.8	1431	1655	2083
mABBA + g^*	28.34	29.48	64.84	90.42	147.7	285.0	254.1	394.3
mABBA + $\tilde{g}$	41.89	57.89	165.9	323.7	683.3	1089	1562	1818

The results of the second comparison (Tables VIII, IX, X, XI) show that the stronger bounding functions decrease the number of iteration steps as it can be expected, but the run-time is significantly better for the mABBA procedure. This shows that the applied stronger bounding functions are too expensive, their applications are unjustified for the problem classes under consideration.

To summarize, our computational experiments indicate that on the problem instances tested, the mABBA procedure based on [11] and [14] led to distinct improvements in computational performance; furthermore, the extensions using sharper bounding functions did not produce any of the hoped-for improvements.

Acknowledgement

The authors would like to thank the anonymous referees for their helpful comments and suggestions.

References

[1] E. M. L. Beale, Branch and Bound Methods for Mathematical Programming Systems, *Annals of Discrete Mathematics*, **5** (1979), 201-219.

[2] Z. Blázsik and B. Imreh, A note on connection between PNS and set covering problems, *Acta Cybernetica*, **12** (1996), 309-312.

[3] C. A. Floudas and I. E. Grossmann, Algorithmic Approaches to Process Synthesis Logic and Global Optimization, *International Symposium on Foundations of Computer Aided Process Design*, Snowmass Village, CO, U.S.A., July 10-15, 1994.

[4] F. Friedler and L. T. Fan, Process Synthesis Incorporating In-Plant Waste Treatment: Algorithmic Approach, presented at the *Proceedings of the ACS Special Symposium on Hazardous-Waste Management*, Atlanta, GA, U.S.A., I 1992, 325-328.

[5] F. Friedler, L. T. Fan, B. Imreh, Process Network Synthesis: Problem Definition, *Networks*, **28** (1998), 119-124.

[6] F. Friedler, K. Tarján, Y.W. Huang, L.T. Fan, Graph-Theoretic Approach to Process Synthesis: Axioms and Theorems, *Chem. Eng. Sci.*, **47**(8) (1992), 1973-1988.

[7] F. Friedler, K. Tarján, Y. W. Huang, L. T. Fan, Combinatorial Structure of Process Network Synthesis, *Sixth SIAM Conference on Discrete Mathematics*, Vancouver, Canada, 1992.

[8] F. FRIEDLER, K. TARJÁN, Y. W. HUANG, L. T. FAN, Graph-Theoretic Approach to Process Synthesis: Application to Waste Minimization, presented at the *Proceedings of the 5th Conference on Hazardous Waste Research*, Kansas State University, Manhattan, KS, U.S.A., **2** 1990, 877-892.

[9] F. FRIEDLER, K. TARJÁN, Y.W. HUANG, L.T. FAN, A Systematic Approach to Waste Minimizing Synthesis of a Chemical Process: Production of Perchloromethyl Mercaptan, presented at the *Proceedings of the Conference of the AIChE National Meeting*, Pittsburg, PA, U.S.A., 1991, 93-98.

[10] F. FRIEDLER, K. TARJÁN, Y.W. HUANG, L.T. FAN, Graph-Theoretic Approach to Process Synthesis: Polynomial Algorithm for maximal structure generation, *Computer chem. Engng.*, **17** (1993), 922-942.

[11] F. FRIEDLER, J. B. VARGA, E. FEHÉR, L. T. FAN, Combinatorially Accelerated Branch-and -Bound Method for Solving the MIP Model of Process Network Synthesis, *Nonconvex Optimization and its Applications*, (eds.: C. A. Floudas and P. M. Pardalos), Kluwer Academic Publishers, Norwell, MA, U.S.A., 1996, 609-626.

[12] F. FRIEDLER, J.B. VARGA, L. T. FAN, Decision-mappings: a tool for consistent and complete decisions in process synthesis, *Chem. Eng. Sci.*, **50**(11) (1995), 1755-1768.

[13] I. E. GROSSMANN, V. T. VOUDOURIS, O. GHATTAS, Mixed-Integer Linear Programming Reformulations for Some Nonlinear Discrete Design Optimization Problems, In: *Recent Advances in Global Optimization* (eds.: C. A. Floudas and P. M. Pardalos) Princeton University Press, New Jersey, 1992.

[14] B. IMREH, F. FRIEDLER, L. T. FAN, An Algorithm for Improving the Bounding Procedure in Solving Process Network Synthesis by a Branch-and-Bound Method, *Developments in Global Optimization*, (eds.: I. M. Bonze, T. Csendes, R. Horst, P. M. Pardalos), KluwerAcademic Publisher, Dordrecht, Boston, London, 1996, 301-348.

[15] L. G. MITTEN, Branch-and-Bound Methods: General Formulation and Properties, *Operations Research*, **18** (1970), 24-34.

Received: September, 1997
Revised: June, 1998
Accepted: June, 1998

Contact address:
B. Imreh
Department of Informatics
József Attila University
Árpád tér 2
H-6701 Szeged, P.O.Box 652
Hungary
e-mail: imreh@inf.u-szeged.hu

G. Magyar
Turku Centre for Computer Science (TUCS)
and Department of Computer Science
University of Turku, DataCity
Lemminkäisenkatu 14 A, FIN-20520
Turku
Finland

BALÁZS IMREH received his M.S. and PhD in mathematics from the Faculty of Science, József Attila University in Szeged, Hungary in 1968 and 1983, respectively. He is currently an Associate Professor at the Department of Foundations of Computer Science of the same university. His research interests include Theory of Automata and Combinatorial Optimization. He has published more than 60 papers in international Scientific Journals and Conference Proceedings. He is a member of INFORMS and EATCS.

GÁBOR MAGYAR received his M.S. in computer science from the Faculty of Science, József Attila University in Szeged, Hungary in 1995. Currently, he is a scholar at Turku Centre for Computer Science (TUCS). His reserach interests cover Operations Research and Combinatorial Optimization.