Computers chem. Engng, Vol. 17, No. 9, pp. 929–942, 1993
Printed in Great Britain.
0098-1354/93 $6.00 + 0.00

GRAPH-THEORETIC APPROACH TO PROCESS SYNTHESIS: POLYNOMIAL ALGORITHM FOR MAXIMAL STRUCTURE GENERATION

F. FRIEDLER,[1]† K. TARJAN,[2] Y. W. HUANG[3] and L. T. FAN[3]

[1]Department of Systems Engineering, Research Institute of Chemical Engineering, Hungarian Academy of Sciences, Veszprém, Pf. 125, 8201, Hungary

[2]Department of Mathematics, University of Veszprém, Veszprém, Hungary

[3]Department of Chemical Engineering, Kansas State University, Manhattan, KS 66506, U.S.A.

(Received 7 January 1992; final revision received 8 March 1993; received for publication 17 March 1993)

Abstract—The maximal structure, which can be expressed as a process graph (P-graph), is the union of all combinatorially feasible process structures of a synthesis problem. This is analogous to the conventionally used term "superstructure" which has not yet been defined mathematically, and thus, cannot be analyzed mathematically. Since a mathematical programming method of process synthesis requires a mathematical model, as its input, based on the super or maximal structure, generating this structure is a problem of fundamental importance. Algorithm MSG, presented here, appears to be the first published algorithm for generation of the maximal structure. This algorithm is efficient because its complexity is polynomial. It is, therefore, advantageous for solving large industrial process synthesis problems. The mathematical basis of maximal structure, the proof of all statements and validation of algorithm MSG are also presented.

1. INTRODUCTION

A mathematical programming method of process synthesis should involve two major steps, the generation of the mathematical model and the solution of this model. The first step is the real "synthesis" part while the second step is the "analysis" part of process synthesis. The majority of the available methods of process synthesis based on mathematical programming deals only with the second step, analysis. This step demands a mathematical model as its input. Since such a model is usually derived from the so-called superstructure, it plays an essential role in process synthesis (Umeda *et al.*, 1972; Ichikawa and Fan, 1973; Grossmann, 1985, 1990; Floudas, 1988). In spite of its importance, in-depth studies of the fundamental mathematical properties of the superstructure have not been undertaken; moreover, the superstructure hitherto has never been rigorously defined mathematically. Mathematical studies of the superstructure are inconsequential for relatively simple process synthesis problems, such as those solved so far by mathematical programming methods. For a complex industrial process synthesis problem, however, it is highly desirable to know: (i) whether useless processing or operating units are included in the model of the problem, i.e. whether the model is more complex than required; and (ii) if there is any operating unit missing from the model, which should be an element of a solution, thereby preventing the optimality to be attained. It is necessary, therefore, that structures of the process be theoretically analyzed and studied from the combinatorial point of view in process synthesis.

For illustration, let us consider an industrial-scale process synthesis problem where the products and available raw materials are specified and where sev29,7
eral dozen operating units have been identified as the possible elements of the optimal process. The MINLP (mixed integer-nonlinear programming) model of this problem has the same number of binary variables as the operating units. The solution of this model is extremely tedious even if it can be executed by an available technique. Thus, it is worthwhile to examine the following questions prior to solving the problem:

(i) Is there any combination of the available operating units that can produce the product from the raw materials? i.e. is there any combinatorially feasible solution to the problem?
(ii) Is it necessary to include all the operating units in the model of synthesis?
(iii) Can the number of binary variables of the model be minimized by an efficient combinatorial technique to reduce the model's complexity? Note that the difficulty of solving a model usually increases exponentially with the number of binary variables.

†To whom all correspondence should be addressed.

The technique introduced in this paper provides an efficient algorithm to answer the above questions, thereby giving rise to enormous savings in the computational effort required by a mathematical programming method for solving the model of the problem. This technique, however, requires rigorous, theoretical analysis of process synthesis.

The maximal structure, to be defined later, denotes the union of all combinatorially feasible structures of a process synthesis problem. The term, maximal structure, appears to be more appropriate than superstructure in process synthesis; the former is more expressive semantically than the latter. It is worth noting that the superstructure has a different connotation in computer science (Cantone *et al.*, 1989). The maximal structure is the largest structure that must be taken into account in process synthesis; it is also the simplest one that contains all combinatorially feasible process structures. An algorithmic construction of the maximal structure has been hitherto impossible. The present work is intended to remedy this situation; the development of the algorithm for the maximal structure generation can be regarded as part of the effort towards automatic process synthesis. The mixed integer programming model of a process synthesis problem can be composed algorithmically from the models of individual operating units by the maximal structure generation algorithm presented here. Moreover, if this process synthesis problem does not have a feasible process structure, the present algorithm will indicate that no mathematical programming examination will be necessary.

The majority of the available mathematical programming methods for process synthesis requires a feasible initial structure. This structure can be generated algorithmically when the maximal structure of the problem is known. Furthermore, in theory, all structures satisfying the objective of the synthesis problem can be generated combinatorially.

The focus of this paper is on the total flowsheet synthesis (see, e.g. Siirola and Rudd, 1971; Lu and Motard, 1985; Douglas, 1988); nevertheless, the results can be extended to other classes of synthesis problems. The algorithm presented is general and has been proven to yield the maximal structure for a synthesis problem without fail, if it exists. Since the algorithm is polynomial in complexity, it is an efficient combinatorial algorithm and thus, is applicable to the solution of large industrial-scale synthesis problems.

2. BASIC TERMINOLOGY AND SUPPORTING THEOREMS

Prior to applying routinely the algorithm for generating a graph representing the maximal structure for a synthesis problem, it is essential to evaluate the complexity of the algorithm and to prove its validity. This should be performed based on rigorous mathematical definitions of the problem and on the basic theorems related to the definitions. An axiom system for this purpose has been presented elsewhere (Friedler *et al.*, 1992).

2.1. Synthesis problem

Let M be a given finite set of all material species, or materials in short, which are to be involved in the synthesis of a process system; it may be a set of names or vectors of characteristics of these materials. The exact description, i.e. the contents, of set M may vary depending on the level of precision or details desired. A simple example for set M can be found in the synthesis of a system for separating the four-component mixture, ABCD, into the pure products, A, B, C and D, only by sharp separators. For this example, $M = \{A, B, C, D, AB, BC, CD, ABC, BCD, ABCD\}$. Instead of the name or characteristics, a simple labeling of the materials is also acceptable. Note that such a labeling exists for any process synthesis problem that can be solved by available mixed-integer mathematical programming methods.

A synthesis problem involving materials represented by set M can be defined by triplet (P, R, O) where P is the set of products ($P \neq \emptyset$); R, the set of raw materials ($R \neq \emptyset$); and O, the set of operating units ($O \neq \emptyset$). The relationships among M, P, R and O can be mathematically expressed as follows:

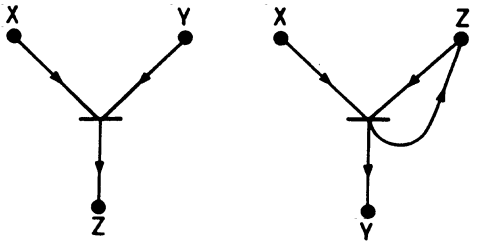
$$P \subseteq M,\ R \subseteq M,\ M \cap O = \emptyset \tag{1}$$

and

$$O \subseteq \mathscr{P}(M) \times \mathscr{P}(M), \tag{2}$$

where $\mathscr{P}$ is a power set, the set of all subsets of a set, and $\times$ is the Cartesian product, signifying the formation of the set of pairs of the elements of two sets. In other words, O is the set of pairs of two subsets of M. If (α, β) is an element of O, then, α designates the set of input materials into operating unit (α, β) and β is the set of output materials from operating unit (α, β). Each element of α represents one material as one input stream to the operating unit, which is independent of streams represented by the other elements of α. Similarly, each element of β represents one material as one independent output stream. eeSets α and β may have common elements; this case represents a simple recycling around the operating unit.

For example, if M has three elements, X, Y and Z, i.e. $M = \{X, Y, Z\}$, then, $\mathscr{P}(M) = \{\emptyset, \{X\}, \{Y\}, \{Z\}, \{X, Y\}, \{X, Z\}, \{Y, Z\}, \{X, Y, Z\}\}$. Naturally, if $O = \{(\{X, Y\}, \{Z\}), (\{X, Z\}, \{Y, Z\})\}$, then, O



(a) Operating unit ({X, Y}, {Z}). (b) Operating unit ({X, Z}, {Y, Z}).

Fig. 1. Operating units.

satisfies constraint (2). The first element of set O, operating unit ($\{X, Y\}, \{Z\}$), has two input materials X and Y and one output material Z (see Fig. 1a). The second element of set O, operating unit ($\{X, Z\}, \{Y, Z\}$), has two input materials X and Z and two output materials Y and Z. Since the input and output share common element Z, this operating unit forms a simple recycling (see Fig. 1b).

It is also assumed that for a synthesis problem a product cannot be a raw material, i.e.

$$P \cap R = \varnothing. \tag{3}$$

This constraint can be satisfied by any synthesis problem even if a material can be produced and purchased (e.g. extracting solvent). In this case, constraint (3) is satisfied by identifying or visualizing a feeder between the material which is an input to the entire system and the recycled material so that these two materials can be differentiated; this signifies that the former is indeed a raw material.

2.2. *Process graph and process structure*

Conventional graphs are suitable for analyzing a process structure (Mah, 1990). Friedler *et al.* (1992), however, have demonstrated that such graphs are incapable of uniquely representing process structures in synthesis. Thus, a special directed bipartite graph, a process graph or P-graph in short, has been introduced to circumvent this difficulty. It is bipartite since its vertices are partitioned into two sets, and no two vertices of the same set are adjacent in the graph.

The mathematical definition of a P-graph and that of a process structure as a P-graph will be elaborated in what follows.

Given two finite sets m and o with:

$$o \subseteq \mathscr{P}(m) \times \mathscr{P}(m), \tag{4}$$

a P-graph is defined to be pair (m, o). The vertices of the graph are the elements of:

$$V = m \cup o, \tag{5}$$

where the vertices belonging to set m are the vertices of the M-type, and those belonging to set o are the vertices of the O-type. The arcs of the graph are the elements of:

$$A = A_1 \cup A_2, \tag{6}$$

$$A_1 = \{(x, y) | y = (\alpha, \beta) \in o \text{ and } x \in \alpha\} \tag{7}$$

$$A_2 = \{(y, x) | y = (\alpha, \beta) \in o \text{ and } x \in \beta\}. \tag{8}$$

An arc is given in A_1 or A_2 as the pair of vertices of the initial and terminal points. According to expression (7), (x, y) is an arc of the graph if y is an element of o in the form of the pairs of α and β, (α, β), so that x is an element of α. Vertex x is the initial point and vertex y is the terminal point of this arc. In expressions (7) and (8), x designates a vertex of the M-type; y, a vertex of the O-type; and α and β, sets of the vertices of the M-type. Each arc in A_1 is from a vertex of the M-type to a vertex of the O-type, and that in A_2 is from a vertex of the O-type to a vertex of the M-type. Note that the arcs are only implicitly defined for a P-graph. This is in sharp contrast to the conventional definition of a graph where both the vertices and arcs are explicitly given.

For illustration, let $m = \{A, B, C, D, E, F\}$ and $o = \{(\{B, C\}, \{A, F\}), (\{D\}, \{C\}), (\{E, F\}, \{C\})\}$. Since the elements of o are the pairs of some subsets of m, o satisfies constraint (4), i.e. (m, o) is a P-graph. The vertices of this graph are in set V, $V = \{A, B, C, D, E, F, (\{B, C\}, \{A, F\}), (\{D\}, \{C\}), (\{E, F\}, \{C\})\}$, where A, B, C, D, E, F are the vertices of the M-type, and the remaining three vertices are those of the O-type. One of the arcs of this graph is from vertex B to vertex $(\{B, C\}, \{A, F\})$ according to expression (7), i.e. arc $(B, (\{B, C\}, \{A, F\}))$. This graph is given in Fig. 2.

The union and intersection of the two P-graphs, (m_1, o_1) and (m_2, o_2), are defined, respectively, as:

$$(m_1, o_1) \cup (m_2, o_2) = (m_1 \cup m_2, o_1 \cup o_2) \tag{9}$$

and

$$(m_1, o_1) \cap (m_2, o_2) = (m_1 \cap m_2, o_1 \cap o_2). \tag{10}$$

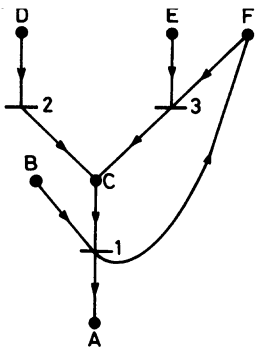


Fig. 2. Simple example of a P-graph.

Table 1. Operating units of Example 1

No.	Type	Inputs	Outputs
		R_1, R_2, R_3	
		R_2, R_3, R_4, R_6	
		R_2, R_3, F	
		R_2, R_3, F	
		R_5	
		D	
		E	
		B, R_6	
		C, R_6	
		K, R_7	
		L, F	
		J, R_4	
		O	
		T	
		N, R_8	
		N, R_9	
		H, R_6	
		I, F, R_6	
		F, G, R_6	
		S	

Furthermore, (m_1, o_1) is a subgraph of (m_2, o_2), i.e. $(m_1, o_1) \subseteq (m_2, o_2)$, if $m_1 \subseteq m_2$ and $o_1 \subseteq o_2$.

If there exist $x_1, x_2, x_3, \ldots, x_{n-1}, x_n$, such that $(x_1, x_2), (x_2, x_3), \ldots, (x_{n-1}, x_n)$ denote arcs of P-graph (m, o), then $[x_1, x_n]$ is defined to be a path from vertex x_1 to vertex x_n. Paths $[x_1, x_i]$ and $[x_i, x_m]$ determine a path from x_1 to x_m; this path is denoted by $[x_1, x_i][x_i, x_m]$. For P-graph (m, o), the in-degree of vertex x, $d^-(x)$, is defined as the number of arcs (y, x); note that $x, y \in (m \cup o)$. For example, there exists a path from vertex D to vertex A on the P-graph given in Fig. 2; the in-degree of vertex C is two, i.e. $d^-(C) = 2$.

The structure of a process can be defined as a P-graph. For synthesis problem (P, R, O), let m be a subset of material set M, and o be a subset of operating unit set O. Moreover, let us suppose that sets m and o satisfy expression (4). Then, the structure of the system with set m of materials and set o of operating units is formally defined as P-graph (m, o). This definition leads to the following properties:

(i) Materials and operating units are represented by vertices of the M- and O-types, respectively.

(ii) Two vertices, one of the M-type and the other of the O-type, are linked by an arc if and only if the corresponding linkage is realized in the process represented by the graph.

(iii) The direction of an arc corresponds to that of flow of a material.

Note that, in practice, no P-graph can arbitrarily be the structure of a process. Any P-graph must satisfy a set of combinatorial properties to appropriately represent a feasible process structure. These properties give rise to a set of axioms to be discussed in the succeeding subsection.

Example 1. Production of PMM. This example is a simplified version of an industrial process synthesis problem, the production of perchloromethyl-mercaptan (PMM), to illustrate the terms introduced. PMM, i.e. material A, is produced from raw materials $R = \{R_1, R_2, R_3, \ldots, R_9\}$. The result of experimental investigations and past experience have enabled us to identify the following set of plausible operating units. $O_0 = \{(\{R_1, R_2, R_3\}, \{B\}), (\{R_2, R_3, R_4, R_6\}, \{C\}), (\{R_2, R_3, F\}, \{D\}), (\{R_2, R_3, F\}, \{E\}), (\{R_5\}, \{F\}), (\{D\}, \{G, H\}), (\{E\}, \{G, I\}), (\{B, R_6\}, \{J, K, L\}), (\{C, R_6\}, \{K, L, M\}), (\{K, R_7\}, \{N\}), (\{L, F\}, \{O\}), (\{J, R_4\}, \{P\}), (\{O\}, \{S, N\}), (\{T\}, \{S, N\}), (\{N, R_8\}, \{U, F\}), (\{N, R_9\}, \{V, F\}), (\{H, R_6\}, \{W, X\}), (\{I, F, R_6\}, \{W, X\}), (\{F, G, R_6\}, \{M, T, Y\}), (\{S\}, \{A\})\}$. The elements of set O_0 are also given in Table 1. P-graph (M_0, O_0), where $M_0 = \{R_1, R_2, R_3, \ldots, R_9, A, B, C, \ldots, X, Y\}$, is established in Fig. 3. The structure given in this figure is not the maximal structure of the problem; in fact, the maximal

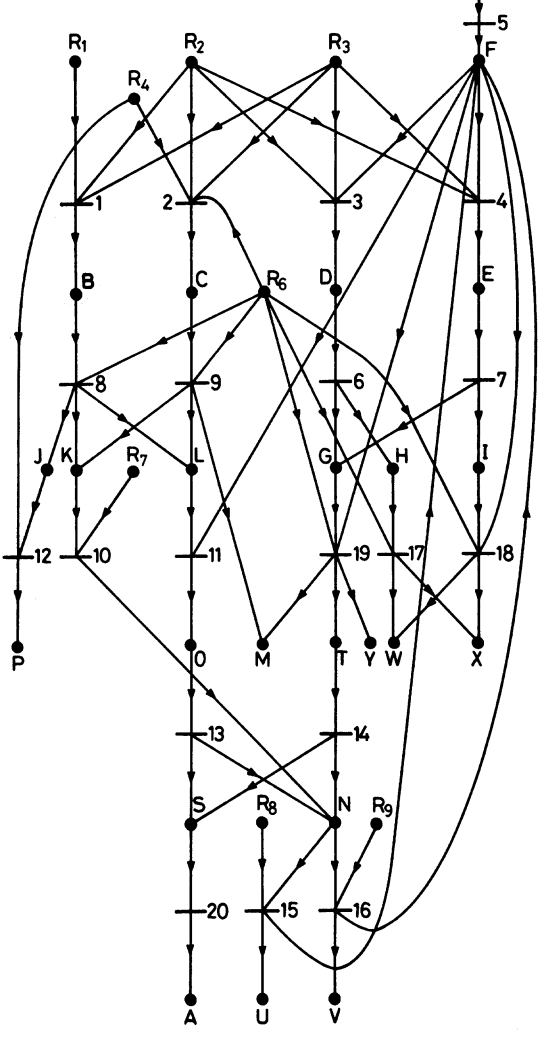


Fig. 3. P-graph (M_0, O_0) of Example 1.

structure is a substructure of this structure, as will be elaborated later.

2.3. Solution-structure

The materials and operating units in a feasible process structure must always conform to certain combinatorial properties. For example, a structure containing no linkage between a raw material and a final product is unlikely to represent any practical process. Hence, it is of vital importance to identify the general combinatorial properties to which a process structure must conform. More important, only those structures satisfying these properties can be feasible structures of a process: no other structures need be considered.

A set of axioms has been constructed to express the combinatorial properties to which a feasible process structure should conform. Since the axioms are the basic statements that cannot be proved and derived from other statements in any theory, they should be simple and obvious. Thus, an axiom system may establish a stable basis for deriving the correspondences in complex theoretical problems. Since process synthesis does not belong to the simple problems, the axiomatic approach is a possible way to explore its theoretical basis.

P-graph (m, o) is defined to be combinatorially feasible or to be a *solution-structure* of synthesis problem (P, R, O) if it satisfies the following axioms:

(S1) Every final product is represented in the graph, i.e. $P \subseteq m$.

(S2) A vertex of the M-type has no input if and only if it represents a raw material, i.e. $\forall x \in m$, $d^-(x) = 0$ if and only if $x \in R$.

(S3) Every vertex of the O-type represents an operating unit defined in the synthesis problem, i.e. $o \subseteq O$.

(S4) Every vertex of the O-type has at least one path leading to a vertex of the M-type representing a final product, i.e. $\forall y_0 \in o$, $\exists$ path $[y_0, y_1]$, where $y_1 \in P$.

(S5) If a vertex of the M-type belongs to the graph, it must be an input to or output from at least one vertex of the O-type in the graph, i.e. $\forall x \in m$, $\exists(\alpha, \beta) \in o$ such that $x \in (\alpha \cup \beta)$.

In other words, for synthesis problem (P, R, O), Axiom (S1) demands that all the final products must be generated by the process represented by a solution-structure; Axiom (S2) explicates the meaning of raw materials, which, by definition, should not be generated by the process being synthesized. If a material can be purchased as a raw material, and it can also be produced by an operating unit in the system, then, a feeder should be represented between the material that is an input to the process and the same material that is produced, and subsequently recycled to the output of this operating unit. The cost of this feeder can be specified to be zero, if necessary. According to Axiom (S3), only those operating units that are defined in the problem can appear in a solution-structure; Axiom (S4) prohibits the existence of an operating unit which is not contributing to product generation; and according to Axiom (S5), only those materials that belong to at least one operating unit of the structure can exist in the structure.

The synthesis problem defined and the axiom system presented in the preceding paragraph are general in the sense that no constraint is imposed on the numbers of products, raw materials and operating units, the numbers of inputs and outputs of each operating unit, and the number of connections among the operating units and materials. For example, no constraint is imposed on the existence of recycling.

One of the basic properties of the set of solution-structures, $S(P, R, O)$, is expressed by the following theorem (Friedler *et al.*, 1992).

Theorem 2.3.1. The set of solution-structures is closed under union, i.e. the union of two solution-structures remains a solution-structure. In other words, if:

$$\sigma_1 \in S(P, R, O) \text{ and } \sigma_2 \in S(P, R, O),$$

then

$$(\sigma_1 \cup \sigma_2) \in S(P, R, O).$$

2.4. Maximal structure

A mathematical definition of the maximal structure (superstructure) of a synthesis problem can be given based on the terms defined in the preceding subsections. For synthesis problem (P, R, O), suppose that $S(P, R, O) \neq \emptyset$; then, the union of all solution-structures, $\mu(P, R, O)$, is defined to be the *maximal structure*, i.e.

$$\mu(P, R, O) = \bigcup_{\sigma \in S(P,R,O)} \sigma. \tag{11}$$

In this graph, each arc or vertex belongs to at least one solution-structure, and each solution-structure is a subgraph. Thus, the mathematical model of a synthesis problem should be based on the maximal structure; this model's complexity is minimal. Naturally, the maximal structure cannot be directly determined algorithmically from its definition. Mathematical analyses of process structures and their features provide the necessary framework to develop an efficient algorithm for generating the maximal structure.

Since the set of solution-structures, $S(P, R, O)$, is closed under the union operation according to Theorem 2.3.1 and the cardinality of this set is finite, the maximal structure is one of the solution-structures. This is expressed by the following theorem (Friedler *et al.*, 1992).

Theorem 2.4.1. $\mu(P, R, O) \in S(P, R, O)$.

Example 2. Let $M_1 = \{A, B, C, D, E, F, G, H, I, J, K, L, M, N, Q, T, U, V\}$ be the set of materials; $P_1 = \{B\}$, the set of products; $R_1 = \{F, H, M, T\}$, the set of raw materials; and:

$$O_1 = \{(\{C, D, F\}, \{A\}), (\{D\}, \{B, G\}), (\{E\}, \{B, U\}), (\{F, G\}, \{C, D\}), (\{G, H\}, \{D\}), (\{H, I\}, \{E\}), (\{J, K\}, \{E\}), (\{M\}, \{G\}), (\{N, Q\}, \{H\}), (\{T, U\}, \{I\}), (\{V\}, \{J\})\},$$

the set of operating units of synthesis problem (P_1, R_1, O_1). P-graph (M_1, O_1), illustrated in Fig. 4, is the structure of a system determined by sets M_1 and O_1. Naturally, this is neither a solution-structure nor the maximal structure of this synthesis problem since it fails to satisfy Axioms (S2), (S4) and (S5). Axiom (S2) is violated because material H is produced by an operating unit, but H is a raw material. Axiom (S4) is not satisfied because no path exists to product B from operating unit 1; see Fig. 4. Axiom (S5) is also violated because material L is not an input to or an output from any operating unit. Obviously, even if a structure violates only one axiom, it cannot be a solution-structure. Figure 5, however, presents a solution-structure; and Fig. 6, the maximal structure of the problem.

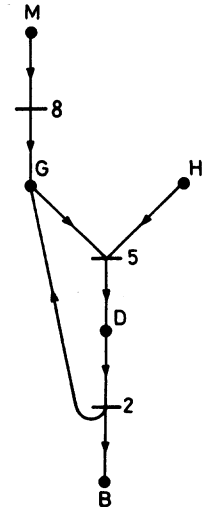


Fig. 5. A solution-structure of synthesis problem (P_1, R_1, O_1).

2.5. *Structural mappings*

Manipulation of P-graphs by combinatorial algorithms can be expressed more tersely by the so-called structural mappings than by set theoretical formalism. Given a P-graph (m, o) $[o \subseteq \mathscr{P}(m) \times \mathscr{P}(m)]$, let us define the following structural mappings.

Definition 2.5.1. For $o' \subseteq o$, let:

$$\Psi^-(o') = \bigcup_{(\alpha,\beta) \in o'} \alpha,$$

$$\Psi^+(o') = \bigcup_{(\alpha,\beta) \in o'} \beta$$

and

$$\Psi(o') = \Psi^-(o') \cup \Psi^+(o').$$

Among these mappings, Ψ^- yields the set of materials of a process structure, each of which is an inlet to at least one operating unit found in set o', Ψ^+ yields the set of materials, each of which is an outlet from at least one operating unit found in set o', and Ψ yields the set of the materials, each of which is either an inlet to or an outlet from at least one operating unit found in set o'. Definition 2.5.1 implies

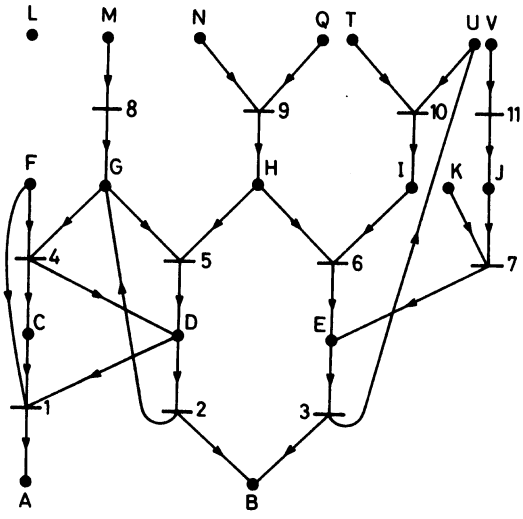


Fig. 4. P-graph (M_1, O_1).

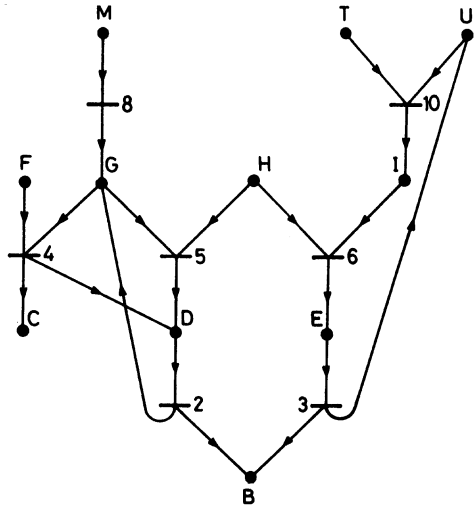


Fig. 6. Maximal structure of synthesis problem (P_1, R_1, O_1).

that for the two sets, $m' \subseteq m$ and $o' \subseteq o$, pair (m', o') is a P-graph if and only if $\Psi(o') \subseteq m'$.

Definition 2.5.2. For $m' \subseteq m$, let:

$$\varphi^-(m') = \{(\alpha,\beta) | (\alpha,\beta) \in o \text{ and } \beta \cap m' \neq \emptyset\},$$

$$\varphi^+(m') = \{(\alpha,\beta) | (\alpha,\beta) \in o \text{ and } \alpha \cap m' \neq \emptyset\},$$

and

$$\varphi(m') = \varphi^-(m') \cup \varphi^+(m').$$

Literally, φ^- yields the set of operating units of a process structure, each of which produces some materials found in set m' as its outlets, φ^+ yields the set of operating units, each of which consumes some materials found in set m' as its inlets, and φ yields the set of operating units, each of which either produces or consumes some materials found in set m'.

Example 2 revisited. To illustrate the structural mappings, let o' be the subset of O_1, i.e.

$$o' = \{(\{H, I\}, \{E\}), (\{J, K\}, \{E\})\} \subseteq O_1.$$

Then,

$$\Psi^-(o') = \{H, I, J, K\}, \Psi^+(o') = \{E\}$$

and $\Psi(o') = \{E, H, I, J, K\}$.

Furthermore, let $m' = \{C, D\}$; then,

$$\varphi^-(m') = \{(\{F, G\}, \{C, D\}), (\{G, H\}, \{D\})\},$$

$$\varphi^+(m') = \{(\{C, D, F\}, \{A\}), (\{D\}, \{B, G\})\}$$

and

$$\varphi(m') = \{(\{F, G\}, \{C, D\}), (\{G, H\}, \{D\}), (\{C, D, F\}, \{A\}), (\{D\}, \{B, G\})\}$$

3. ALGORITHM FOR GENERATING THE MAXIMAL STRUCTURE

According to the convention of computer science, the algorithm for maximal structure generation (MSG) is given in Pidgin Algol introduced by Aho *et al.* (1974); also see Papadimitriou (1982). A brief summary of Pidgin Algol is given in Appendix A.

The algorithm MSG in Pidgin Algol is presented in Fig. 7. This algorithm has two major parts, the reduction part and the composition part. In the reduction part (statements st1, st2, st3 and loop lp4 of Fig. 7), the materials and operating units that must

```
input: sets M, P, R, O;
comment: P⊆M, R⊆M, O⊆(℘(M)×℘(M)), O∩M = ∅, P∩R = ∅;
output: maximal structure (m, o) of synthesis problem (P, R, O);
begin
comment: reduction part of the algorithm;
st1:  O := O \ φ⁻(R);
st2:  M := Ψ(O);
st3:  r := Ψ⁻(O) \ (Ψ⁺(O)∪R);
lp4:  while r is not empty do
      begin
      let x be an element of r;
      M := M \ {x};
      o := φ⁺({x});
      O := O \ o;
      r := (r∪(Ψ⁺(o) \ Ψ⁺(O))) \ {x}
      end;
co5:  if P∩M≠P then stop;  comment: there is no maximal structure;
comment: composition part of the algorithm;
st6:  p := P;  m := ∅;  o := ∅;
lp7:  while p is not empty do
      begin
      let x be an element of p;
      m := m∪{x};
      o_x := φ⁻({x});
      o := o∪o_x;
      p := (p∪Ψ⁻(o_x)) \ (R∪m)
      end;
st8:  m := Ψ(o)
end
```

Fig. 7. Algorithm MSG.

not belong to the maximal structure are directly and partially excluded from consideration. In the composition part (statements st6, st8 and loop lp7 of Fig. 7) those and only those materials and operating units are selected from the structure resulting from the reduction part that must belong to the maximal structure.

In generating the maximal structure for synthesis problem (P, R, O), the operating units producing the raw materials are excluded from consideration by statements st1 and st2. Set r, defined in statement st3, contains the materials that are only consumed but never produced by the operating units; however, these materials are not defined by the synthesis problem as raw materials. Obviously, this set of materials also need be excluded from the maximal structure. This is achieved in loop lp4, after which the set of materials M and that of operating units O satisfy Axiom (S2). Condition co5 checks if Axiom (S1) is satisfied, i.e. if every product is represented in the process structure generated thus far. If the condition is not satisfied, no maximal structure exists for the synthesis problem. Otherwise, the maximal structure is constructed stepwisely by collecting the operating units satisfying Axiom (S4) in loop lp7. Note that Axiom (S3) is automatically satisfied since only the operating units defined by the synthesis problem are considered by the algorithm. The set of materials contained in the maximal structure are obtained through the mapping operation of statement st8. Such a mapping assures that Axiom (S5) is satisfied by the maximal structure; the maximal structure of Example 1 generated by algorithm MSG is given in Fig. 8.

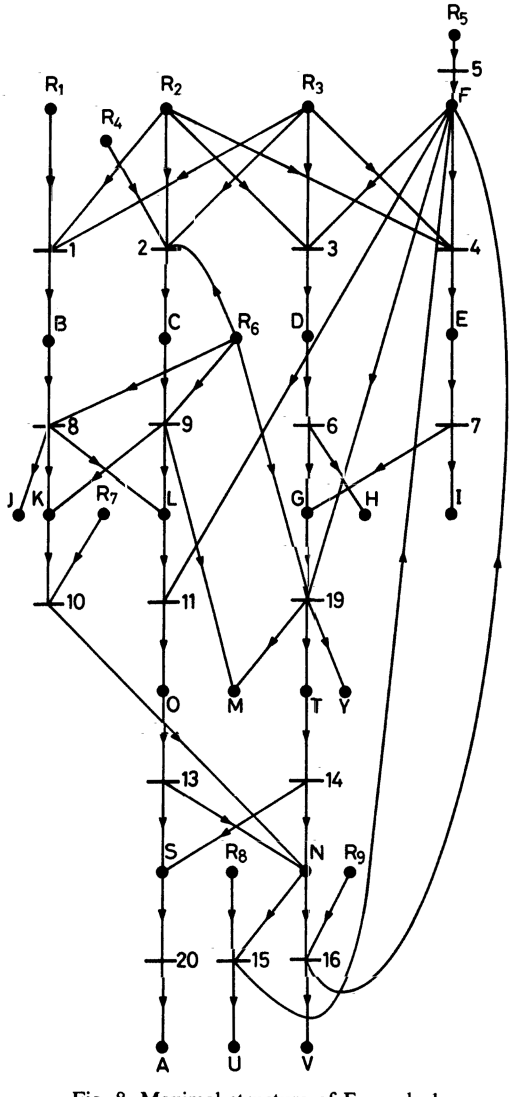


Fig. 8. Maximal structure of Example 1.

Example 2 revisited. Let us generate the maximal structure of Example 2 by algorithm MSG. In the input step of the algorithm M, P, R and O denote the following sets:

$$M = \{A, B, C, D, E, F, G, H, I, J, K, L, M, N, Q, T, U, V\},$$

$$P = \{B\},$$

$$R = \{F, H, M, T\}$$

$$O = \{(\{C, D, F\}, \{A\}), (\{D\}, \{B, G\}), (\{E\}, \{B, U\}), (\{F, G\}, \{C, D\}), (\{G, H\}, \{D\}), (\{H, I\}, \{E\}), (\{J, K\}, \{E\}), (\{M\}, \{G\}), (\{N, Q\}, \{H\}), (\{T, U\}, \{I\}), (\{V\}, \{J\})\}.$$

Structure (M, O) is given in Fig. 9a. Operating units generating raw material are identified in statement st1 by:

$$\varphi^{-}(R) = \{(\{N, Q\}, \{H\})\}.$$

This operating unit is excluded from further consideration after statements st1 and st2 are executed, where:

$$O = O \backslash \varphi^{-}(R) = \{(\{C, D, F\}, \{A\}), (\{D\}, \{B, G\}), (\{E\}, \{B, U\}), (\{F, G\}, \{C, D\}), (\{G, H\}, \{D\}), (\{H, I\}, \{E\}), (\{J, K\}, \{E\}), (\{M\}, \{G\}), (\{T, U\}, \{I\}), (\{V\}, \{J\})\}$$

and

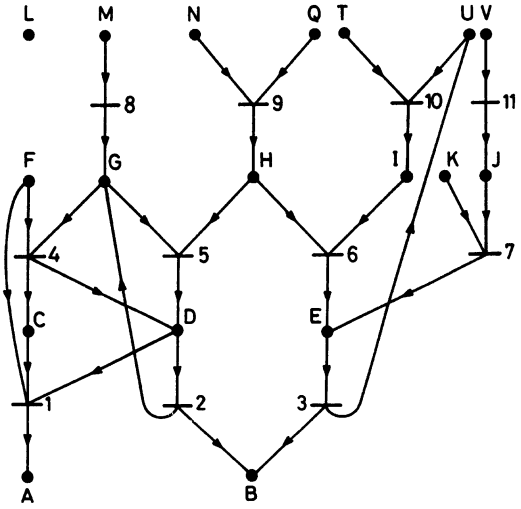
$$M = \Psi(O) = \{A, B, C, D, E, F, G, H, I, J, K, M, T, U, V\}$$

This gives rise to the structure given in Fig. 9b. By definition, we now have:

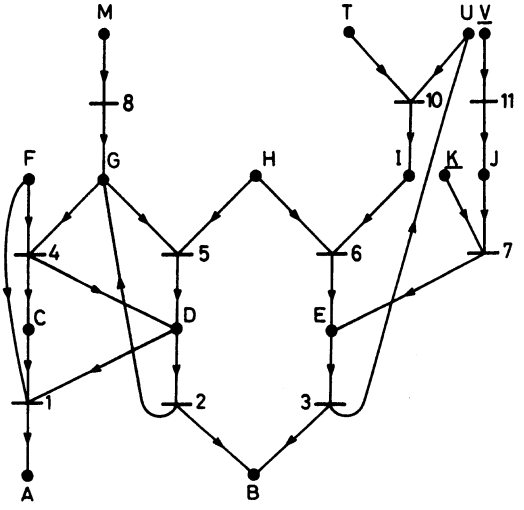
$$\Psi^{-}(O) = \{C, D, E, F, G, H, I, J, K, M, T, U, V\}$$

and

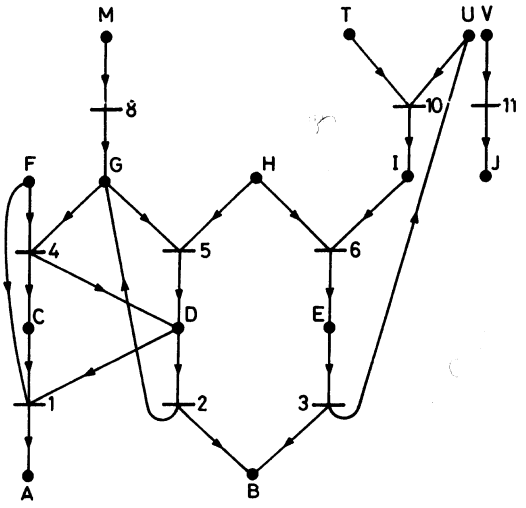
$$\Psi^{+}(O) = \{A, B, C, D, E, G, I, J, U\}.$$



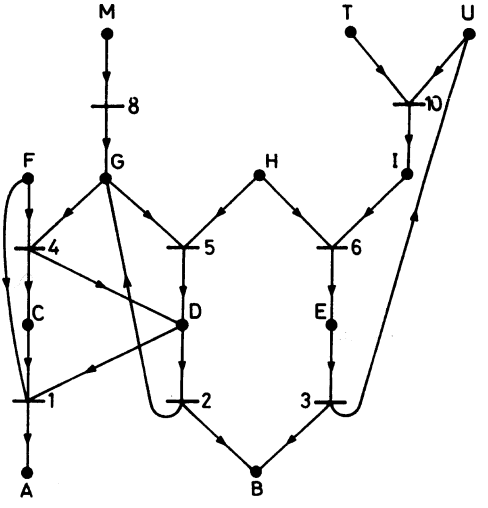
(a) Input structure for the algorithm.



(b) Structure generated by statements st1 and st2; materials belonging to set r of st3 are underlined.



(c) Structure generated after the first iteration of loop lp4.



(d) Structure generated after the third iteration of loop lp4.

Fig. 9. Structures generated in the reduction part of algorithm MSG for Example 2.

Hence, in statement st3, we obtain:

$$r = \Psi^-(O)\backslash(\Psi^+(O)\cup R) = \{K, V\}.$$

This set of materials is strictly consumed by the operating units although they are not raw materials. Thus, they must be excluded from the maximal structure.

Since r is not empty, the algorithm enters loop lp4. At the end of the first iteration of the loop, we have:

$$M = M\backslash\{K\} = \{A, B, C, D, E, F, G, H, I, J, M, T, U, V\},$$

$$o = \varphi^+(\{K\}) = \{(\{J, K\}, \{E\})\},$$

$$O = O\backslash o = \{(\{C, D, F\}, \{A\}), (\{D\}, \{B, G\}), (\{E\}, \{B, U\}), (\{F, G\}, \{C, D\}), (\{G, H\}, \{D\}), (\{H, I\}, \{E\}), (\{M\}, \{G\}), (\{T, U\}, \{I\}), (\{V\}, \{J\})\},$$

$$\Psi^+(o) = \{E\},$$

$$\Psi^+(O) = \{A, B, C, D, E, G, I, J, U\}$$

and

$$r = (r\cup(\Psi^+(o)\backslash\Psi^+(O)))\backslash\{K\} = \{V\},$$

where r is an updated set of materials which are strictly consumed by the operating units while they are not raw materials. Structure generated at the end of this iteration is given in Fig. 9c. Naturally, all elements in set r remain to be excluded from the

maximal structure. The algorithm exits loop lp4 after three iterations; in the third iteration, we have the following steps:

$$M = M\backslash\{J\} = \{A, B, C, D, E, F, G, H, I, M, T, U\},$$

$$o = \varphi^{+}(\{J\}) = \emptyset,$$

$$O = O\backslash o = \{(\{C, D, F\}, \{A\}), (\{D\}, \{B, G\}), (\{E\}, \{B, U\}), (\{F, G\}, \{C, D\}), (\{G, H\}, \{D\}), (\{H, I\}, \{E\}), (\{M\}, \{G\}), (\{T, U\}, \{I\})\},$$

$$\Psi^{+}(o) = \emptyset,$$

$$\Psi^{+}(O) = \{A, B, C, D, E, G, I, U\}$$

and

$$r = (r \cup (\Psi^{+}(o)\backslash\Psi^{+}(O)))\backslash\{J\} = \emptyset.$$

The resultant structure is given in Fig. 9d. Since:

$$P = \{B\} \subset M = \{A, B, C, D, E, F, G, H, I, M, T, U\},$$

the maximal structure exists for this problem; it is constructed starting from product B, by adding operating units one by one to the existing structure in the iterations of loop lp7. For example, the first iteration of loop lp7 yields the following:

$$m = m \cup \{B\} = \{B\},$$

$$o_x = \varphi^{-}(\{B\}) = \{(\{D\}, \{B, G\}), (\{E\}, \{B, U\})\},$$

$$o = o \cup o_x = \{(\{D\}, \{B, G\}), (\{E\}, \{B, U\})\}$$

and

$$p = (p \cup \Psi^{-}(o_x))\backslash(R \cup m) = \{D, E\}.$$

The structures after each iteration of loop lp7 are given in Fig. 10, the last of which is the maximal structure for the problem where its materials are determined by statement st8.

4. PROOF OF THE VALIDITY OF ALGORITHM MSG

Since the generation of maximal structure is a fundamentally important problem in process synthesis, its algorithm needs to be validated. Let synthesis problem (P, R, O) be given on M and the maximal structure $\mu(P, R, O)$ of the synthesis problem be denoted by $(\bar{m}, \bar{o})$, if the structure exists. Let us also define the sets $O' = O\backslash\varphi^{-}(R)$ and $M' = \Psi(O')$. The proofs of the following theorems can be found in Appendix B.

Theorem 4.1. If $S(P, R, O) \neq \emptyset$, then: (i) $\mu(P, R, O) \subseteq (M', O')$; and (ii) for $x \in (M' \cap R)$, $d^{-}_{(M', O')}(x) = 0$.

Theorem 4.2. Let us suppose that $(m, o) \in S(P, R, O)$ and that P-graph (m', o') is a subgraph of (M, O). If $(m, o) \subseteq (m', o')$ and $d^{-}(x) = 0$ for $x \in (m'\backslash R)$, then $x \notin m$, i.e. $(m'\backslash\{x\}) \supseteq m$.

Theorem 4.3. If $x \notin m$ and $(\alpha, \beta) \in \varphi^{+}_{(M', O')}(\{x\})$, i.e. $(\alpha, \beta) \in O'$ and $x \in \beta$, for P-graph $(m, o) \subseteq (M', O')$, then $(\alpha, \beta) \notin o$.

Let sets M and O of algorithm MSG be denoted by M'' and O'', respectively, immediately after loop lp4 is completed. P-graph (M'', O'') satisfies Axiom (S2); an element of set M'' has no input if and only if it represents a raw material as described by the next theorem.

Theorem 4.4. (i) (M'', O'') is a P-graph, i.e. $O'' \subseteq (\mathcal{P}(M'') \times \mathcal{P}(M''))$; and (ii) P-graph (M'', O'') satisfies Axiom (S2) of (P, R, O).

Theorems 4.2, 4.3 and 4.4 gives rise to the next corollary.

Corollary 4.1. If $S(P, R, O) \neq \emptyset$ and $(m, o) \in S(P, R, O)$, then $(m, o) \subseteq (M'', O'')$ and P-graph (M'', O'') satisfies Axiom (S2).

Theorem 4.5. Let P-graph $\sigma = (m, o)$ be given and $p \subseteq m$. Also let $o' = \{y \mid y \in o$ and there exists path $[y, z]$ in σ such that $z \in p\}$, $m' = \Psi(o')$ and $\sigma' = (m', o')$. Then, $d^{-}_{\sigma}(x) = 0$ if and only if $d^{-}_{\sigma'}(x) = 0$ for any element x of m'.

Let us define the sets $o^{*} = \{y \mid y \in O''$ and there exists path $[y, z]$ in (M'', O'') such that $z \in P\}$ and $m^{*} = \Psi(o^{*})$.

Theorem 4.6. If $P \subseteq M''$, then $(m^{*}, o^{*}) \in S(P, R, O)$.

Theorem 4.7. $S(P, R, O) \neq \emptyset$ if and only if $P \subseteq M''$.

Theorem 4.8. If $P \subseteq M''$, then $\bar{o} = o^{*}$.

Let o' and m' denote sets o and m, respectively, defined in loop lp7 of algorithm MSG immediately after the looping is completed.

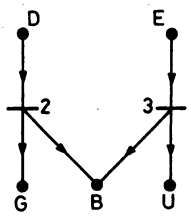
Theorem 4.9. If $P \subseteq M''$, then $o^{*} = o'$.

Theorem 4.10. Algorithm MSG determines the maximal structure of a synthesis problem in a finite number of steps, if it exists.

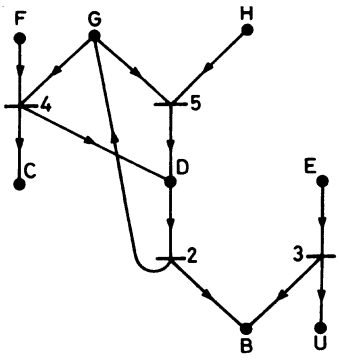
5. COMPLEXITY ANALYSIS OF ALGORITHM MSG

Even a validated algorithm might be useless if its execution or computer realization is excessively tedious for a problem encountered in practice. Theoretically, complexity analysis can yield information on the algorithm's behavior as the size of the problem increases (see, e.g. Hartmanis, 1989). In the worst case, an upper bound is obtained for the number of elementary steps of the algorithm as a function of the size of the problem. It is usually sufficient to have an upper estimate for this function; this is why the complexity analysis of an algorithm adopts the so-called "big-oh" notation of the mathematics (see, e.g. Rosen, 1990) to represent the order of magnitude of the complexity.

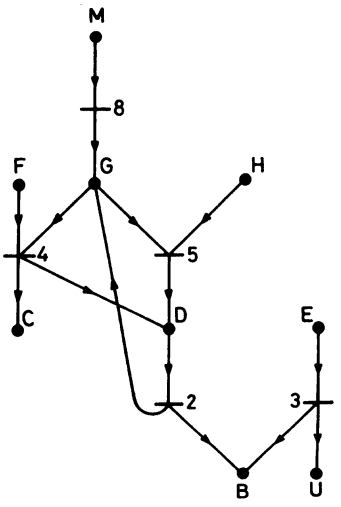
Definition 5.1. For two functions of positive integers f and g, $f(x) = O(g(x))$ if there exists a positive



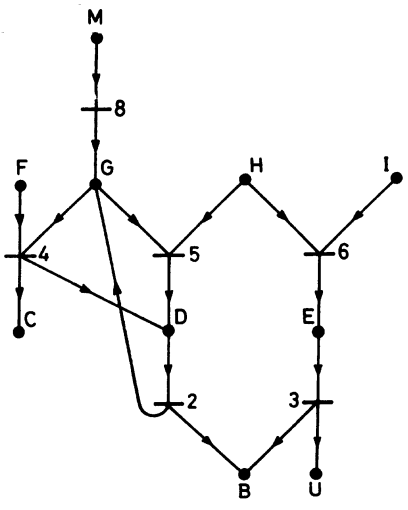
(a) Structure after the first iteration of loop lp7.



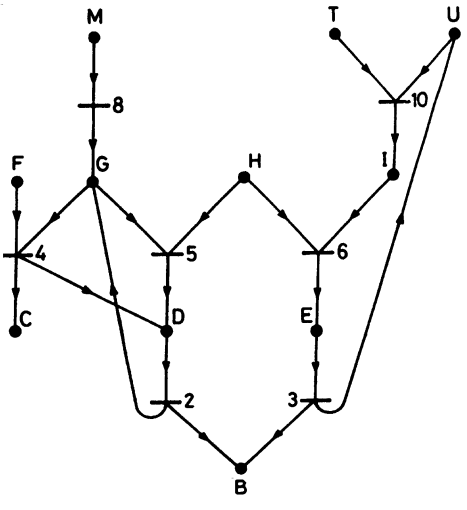
(b) Structure after the second iteration of loop lp7.



(c) Structure after the third iteration of loop lp7.



(d) Structure after the fourth iteration of loop lp7.



(e) Structure after the fifth iteration of loop lp7.

Fig. 10. Structures generated in the composition part of algorithm MSG for Example 2.

constant C such that $|f(x)| \leqslant C|g(x)|$ for all but finitely many positive integers.

Note that the big-oh notation is not unique, the function $g(x)$ is usually chosen from well-known functions such as x^n. If an algorithm solves a problem of size x by $f(x)$ elementary steps and $f(x) = O(g(x))$, then the complexity of this algorithm is $g(x)$. For the most effective class of combinatorial algorithms, $g(x)$ is a polynomial, i.e. the complexity of these algorithms is polynomial. The complexity of the majority of the combinatorial algorithms, however, is greater than polynomial; e.g. it can be exponential.

Given synthesis problem (P, R, O) on material set M, let us denote the number of elements of set M as l and that of set O, as n, i.e. $|M| = l$ and $|O| = n$. Furthermore, let d_m and d_o be the maximal degrees of vertices of the M- and the O-types, respectively, of P-graph (M, O). The complexity of algorithm MSG will be analyzed on the basis of the following data structures.

Let us define matrix $\mathbf{M}$ of $l \times d_m$, matrix $\mathbf{O}$ of $n \times d_o$, vector $\mathbf{i}_m$ of l, vector $\mathbf{i}_o$ of n and vector $\mathbf{i}_s$ of $(n + l)$. P-graph (M, O) is stored in matrices $\mathbf{M}$ and $\mathbf{O}$ in the following manner. Each row of matrix $\mathbf{M}$ or $\mathbf{O}$ corresponds, respectively, to a vertex of the M-type or the O-type of P-graph (M, O). Suppose that for $x \in O$ and $y \in M$, arc(x, y) exists from vertex x to vertex y. Then, this arc is represented as a pointer

with a positive sign in the row of vertex x in matrix **O** and with a negative sign in the row of vertex y in matrix **M**. An arc from a vertex of the M-type to that of the O-type is represented similarly.

The complexity of statement st1 of algorithm MSG in Fig. 7 is $O(l \cdot d_m \cdot d_o)$. This is determined as follows: there can be at most l materials defined in set R for a synthesis problem. For each material in the set, there can be at most d_m operating units producing this material. Thus, statement st1 needs to manipulate matrix **M** at most $(l \cdot d_m)$ times to calculate $\varphi^-(R)$. In each of the $(l \cdot d_m)$ manipulations, matrix **O** will be modified d_o times, and matrix **M** will be modified one time to effect the calculation of $[O \backslash \varphi^-(R)]$. As a result, statement st1 can involve at most $[l \cdot d_m \cdot (d_o + 1)]$ matrix manipulations, thereby giving rise to the complexity of $O(l \cdot d_m \cdot d_o)$. Similarly, the complexity of statement st2, statement st3, loop lp4, loop lp7 or statement st8 can be estimated, respectively, to be $O(l \cdot d_m)$, $O(l \cdot d_m)$, $O(l \cdot d_m \cdot d_o)$, $O(l \cdot d_m \cdot d_o)$ or $O(l \cdot d_m)$; hence, the overall complexity of algorithm MSG is $O(l \cdot d_m \cdot d_o)$; in fact, it is polynomial, thereby indicating that it is an efficient algorithm.

6. OTHER APPLICATIONS OF ALGORITHM MSG

Algorithm MSG can be adopted in the preliminary step in solving the MINLP model of process synthesis, i.e. in the step of model generation, whenever the mathematical programming approach is applicable. Moreover, this algorithm can validate existing "superstructures" or can reveal that the solution is unattainable for the synthesis problem under consideration. From the available databases and knowledge bases, algorithm MSG selects the set of those processing schemes, such as reaction paths and separation sequences, which can be incorporated into a process to be synthesized, if the set of products and the set of raw materials are specified.

7. CONCLUDING REMARKS

The term, maximal structure, of process synthesis is defined mathematically; it is the union of all combinatorially feasible process structures and is also the combinatorially minimized superstructure of a synthesis problem. The mathematical model of a synthesis problem should be based on this maximal structure. Moreover, a novel, exacting, efficient algorithm for generating the maximal structure, algorithm MSG, is presented.

Acknowledgements—The authors thank Professor B. Imreh of Department of Computer Science, University of Szeged, Hungary, for critically reviewing our manuscript and independently verifying its mathematical contents. Although the research in this article has been funded in part by the U.S. Environmental Protection Agency under assistance agreement R-815709 to the Great Plains/Rocky Mountain Hazardous Substance Research Center with headquarters at Kansas State University, it has not been subjected to the Agency's peer and administrative review and therefore may not necessarily reflect the views of the Agency and no official endorsement should be inferred. This research was partially supported by the Hungarian Academy of Sciences and Kansas State University Center for Hazardous Substance Research.

NOMENCLATURE

A = Set of arcs of a P-graph
d^- = In-degree of a vertex
(m, o) = P-graph
m^*, m, M = Set of materials
o^*, o, O = Set of operating units
O = Order of magnitude
P = Set of products
(P, R, O) = Synthesis problem defined by the specific sets of products (P), raw materials (R) and operating units (O)
R = Set of raw materials
$S(P, R, O)$ = Set of solution-structures for synthesis problem (P, R, O)
V = Set of vertices of a P-graph
$[y_i, y_j]$ = Path in a P-graph
$n \times m$ = Size of a matrix with n rows and m columns
$X \times Y$ = Cartesian product of sets X and Y
α, β = Set of materials
(α, β) = Operating unit
$\mu(P, R, O)$ = Maximal structure for synthesis problem (P, R, O)
σ = Solution-structure
Ψ, Ψ^-, Ψ^+ = Structural mapping from a set of operating units to a set of materials
$\varphi, \varphi^-, \varphi^+$ = Structural mapping from a set of materials to a set of operating units
$\varnothing$ = Empty set
$\mathscr{P}$ = Power set
$\forall$ = For all
$\exists$ = There exists
$\{\}$ = Set
$||$ = Cardinality of a set or the absolute value of a real function
$\backslash$ = Set difference
$(\subset)\subseteq$ = (Proper) subset or subgraph
$\in$ = Element
$\notin$ = Not an element
$\cap$ = Intersection of sets or graphs
$\cup$ = Union of sets or graphs
$\bigcup_{i \in I} x_i$ = Union of all the elements of sets $x_i (i \in I)$

REFERENCES

Aho A. V., J. E. Hopcroft and J. D. Ullman, *The Design and Analysis of Computer Algorithm*, Section 1.8, pp. 33–39. Addison Wesley, Reading, MA (1974).

Cantone D., A. Ferro and E. Omodeo, *Computable Set Theory*, pp. 102–103. Oxford University Press, New York (1989).

Douglas J. M., *Conceptual Design of Chemical Processes*. McGraw-Hill, New York (1988).

Floudas C. A. and S. H. Anastasiadis, Synthesis of distillation sequences with several multicomponent feed and product streams. *Chem. Engng Sci.* **43**, 2407–2419 (1988).

Friedler F., K. Tarjan, Y. W. Huang and L. T. Fan, Graph-theoretic approach to process synthesis: axioms and theorems. *Chem. Engng Sci.* **47**, 1973–1988 (1992).

Grossmann I. E., Mixed-integer programming approach for the synthesis of integrated process flowsheets. *Computers chem. Engng* **9**, 463–482 (1985).
Grossmann I. E., MINLP optimization strategies and algorithms for process synthesis. In *Foundations of Computer-Aided Process Design* (J. J. Siirola, I.E. Grossmann and G. Stephanopoulos, Eds), pp. 105–132. Elsevier, New York (1990).
Hartmanis J., Computational complexity theory. In *AMS Short Course Lecture Notes: Proceedings of Symposia in Applied Mathematics*, Vol. 38 (J. Hartmanis, Ed.), pp. 1–17. American Mathematical Society, Providence, RI (1989).
Ichikawa A. and L. T. Fan, Optimal synthesis of process systems. *Chem. Engng Sci.* **28**, 357–373 (1973).
Lu M. D. and R. L. Motard, Computer-aided total flowsheet synthesis. *Computers chem. Engng* **9**, 431–445 (1985).
Mah R. S. H., *Chemical Process Structures and Information Flows*. Butterworth, Stoneham, MA (1990).
Papadimitriou C. H., *Combinatorial Optimization: Algorithms and Complexity*, pp. 24–25. Prentice-Hall, Englewood Cliffs, NJ (1982).
Rosen K. H., *Discrete Mathematics and Its Applications*. McGraw-Hill, New York (1990).
Siirola J. J. and D. F. Rudd, Computer-aided synthesis of chemical process designs. *Ind. Engng Chem. Fundam.* **10**, 353–362 (1971).
Umeda T., A. Hirai and A. Ichikawa, Synthesis of optimal processing system by an integrated approach. *Chem. Engng Sci.* **27**, 795–804 (1972).

APPENDIX A

Short Summary of Pidgin Algol

Pidgin Algol is a high-level language whose purpose is to describe algorithms for publication and mathematical examination (Aho *et al.*, 1974; Papadimitriou, 1982). This language uses traditional mathematical and programming language constructs, such as expressions, conditions, statements and procedures. It does not have a fixed set of data types.

Statements

variable: = expression;
if condition **then** statement **else** statement;
while condition **do** statement;
repeat statement **until** condition;
for variable: = initial-value **step** step-size **until** final-value **do** statement;
for all $x \in X$ **do** statement;
label: statement;
goto label;
begin
  statement;
  statement;
  statement;
end;
procedure name (list of parameters): statement
return;
return expression;
procedure-name (arguments);
read variable;
write expression;
comment comment;
any other miscellaneous statements

APPENDIX B

Proof of Theorems

The following obviously satisfied propositions have been used in the proof of the theorems:

Proposition A1. If $x \in m$ for P-graph (m, o), then $d^-(x) = |\varphi^-(\{x\})|$.

Proposition A2. Let synthesis problem (P, R, O) be given; then

(i) Axiom (S5) is satisfied by P-graph (m', o') if and only if $m' = \Psi(o')$ and
(ii) If P-graph (m', o') satisfies Axiom (S5), then Axiom (S1) is satisfied by (m', o') if and only if $P \subseteq \Psi(o')$.
(iii) If $(m', o') \in S(P, R, O)$, then $m' = \Psi(o')$.

Proof of Theorem 4.1

(i) If $S(P, R, O) \neq \varnothing$, then, there exists the maximal structure $(\bar{m}, \bar{o})$ of synthesis problem (P, R, O). Furthermore, according to Theorem 2.4.1, $(\bar{m}, \bar{o}) \in S(P, R, O)$, the maximal structure must satisfy the axioms of solution-structures. Axiom (S3) implies that $\bar{o} \subseteq O$, and Axiom (S2) asserts that if $x \in R$ and $(\alpha, \beta) \in \varphi^-(\{x\})$, then $(\alpha, \beta) \notin \bar{o}$. In other words, $\bar{o} \cap \varphi^-(R) = \varnothing$. Thus, $\bar{o} \subseteq O \backslash \varphi^-(R) = O'$. It follows from Proposition A2 that $\bar{m} = \Psi(\bar{o}) \subseteq \Psi(O') = M'$. With $\bar{m} \subseteq M'$ and $\bar{o} \subseteq O'$, we have $(\bar{m}, \bar{o}) = \mu(P, R, O) \subseteq (M', O')$.

(ii) For $x \in (M' \cap R)$, $\varphi_{(\bar{M}', O')}(\{x\}) \subseteq \varphi_{(\bar{M}', O')}(M' \cap R) \subseteq \varphi^-(M' \cap R) \subseteq \varphi^-(R)$, $\varphi_{(\bar{M}', O')}(\{x\}) \subseteq O'$ and $O' \cap \varphi^-(R) = \varnothing$; thus, $\varphi_{(\bar{M}', O')}(\{x\}) = \varnothing$. Since $d^-_{(\bar{M}', O')}(x) = |\varphi_{(\bar{M}', O')}(\{x\})|$ according to Proposition A1, $d^-_{(\bar{M}', O')}(x) = 0$.

Proof of Theorem 4.2

Since (m, o) is a solution-structure, Axiom (S2) implies that if $x \in m$ and $d^-_{(m, o)}(x) = 0$, then $x \in R$. Consequently, if $d^-_{(\bar{m}', o')}(x) = 0$ and $x \notin R$, then $x \notin m$.

Proof of Theorem 4.3

(m, o) is a P-graph; therefore, $o \subseteq (\mathscr{P}(m) \times \mathscr{P}(m))$. This implies that for $(\alpha, \beta) \in o$, $\beta \in \mathscr{P}(m)$, which, in turns, implies that $\beta \subseteq m$. Stated differently, if β is not a subset of m, then $(\alpha, \beta) \in o$.

Proof of Theorem 4.4

(i) It will be proved first that (M, O) of algorithm MSG is a P-graph in every iteration of loop lp4. At the initialization of loop lp4 (statements st1, st2 and st3 of Fig. 6), that (M, O) is a P-graph holds trivially. Let us assume that (M, O) being a P-graph holds at the end of a certain iteration of loop lp4. It will be proved that (M, O) being a P-graph also holds at the end of the next iteration. For $x \in r$ (r is defined in loop lp4 of algorithm MSG as a set of materials which are strictly consumed by the operating units while they are not raw materials), $\forall (\alpha, \beta) \in O$, $x \notin \beta$. Thus, if $\beta \in \mathscr{P}(M)$, then $\beta \in \mathscr{P}(M \backslash \{x\})$. In other words, if for $(\alpha, \beta) \in O$, $\beta \in \mathscr{P}(M)$ at the end of a certain iteration of loop lp4, then $\beta \in \mathscr{P}(M)$ at the end of the next iteration. For $x \in r$, $\forall (\alpha, \beta) \in O \backslash o$ (o is defined in loop lp4 of algorithm MSG as a set of operating units each of which has x as part of its inlet materials), $x \notin \alpha$. This implies that if $\alpha \in \mathscr{P}(M)$, then $\alpha \in \mathscr{P}(M \backslash \{x\})$.

(ii) Let $\gamma_1 = (M', O')$ and $\gamma_2 = (M'', O'')$. Since $\gamma_2 \subseteq \gamma_1$, for $x \in M''$, $d^-_{\gamma_2}(x) \leqslant d^-_{\gamma_1}(x)$. According to Theorem 4.1, $d^-_{\gamma_1}(x) = 0$ for $x \in (M' \cap R)$; hence, if $x \in (M'' \cap R)$, then $d^-_{\gamma_2}(x) = 0$. Conversely, in every iteration of loop lp4, if $d^-(x) = 0$, then $x \in R$ for $x \in M \backslash r$.

Proof of Theorem 4.5

From $o' \subseteq o$ and $m' = \Psi(o')$, it follows that $m' = \Psi(o') \subseteq \Psi(o) \subseteq m$; thus, $\sigma' \subseteq \sigma$. Moreover:

(i) $\sigma' \subseteq \sigma$ implies that if $d^-_\sigma(x) = 0$, then $d^-_{\sigma'}(x) = 0$ for any element x of m'.

(ii) Since $\Psi^-(o') \supseteq \{x | x \in m' \text{ and } d^-_{\sigma'}(x) = 0\}$, it is sufficient to prove that for any element x of $\Psi^-(o')$, $d^-_\sigma(x) =$

$d_{o'}^{-}(x)$. Let $m'' = \Psi^{-}(o')$ and $o'' = \varphi_{o'}^{-}(\{x\})$ for some $x \in m''$. Then, $o'' \subseteq \varphi_{o'}^{-}(m'') = \varphi_{o'}^{-}(\Psi^{-}(o'))$, and from the definition of (m', o'), $\varphi_{o'}^{-}(\Psi^{-}(o')) \subseteq o'$. Furthermore, $o'' \subseteq o$ holds trivially. Thus, we obtain $d_{o'}^{-}(x) = |o''| = d_{o'}^{-}(x)$ by resorting to Proposition A1.

Proof of Theorem 4.6

(i) Axioms (S4) and (S5) are trivially satisfied.

(ii) Now, according to Proposition A2, Axiom (S1) is satisfied if and only if $P \subseteq \Psi(o^*)$. By Theorems 4.4 and 4.5, $d^{-}(x) = 0$ implies that $x \in R$ for any element x of M''. Since $P \subseteq M''$ and $P \cap R = \emptyset$, the definition of o^* implies that for any $x \in P$, there exists $(\alpha, \beta) \in o^*$ such that $x \in \beta$. This implies that $P \subseteq \Psi(o^*)$.

(iii) For $x \in M''$, $d^{-}(x) = 0$ if and only if $x \in R$; therefore, Theorem 4.5 implies that for any $x \in m^*$, $d_{(m^*, o^*)}^{-}(x) = 0$ if and only if $x \in R$. Axiom (S2), therefore, is satisfied.

(iv) Since $o^* \subseteq O'' \subseteq O$, Axiom (S3) is satisfied.

Proof of Theorem 4.7

(i) From Corollary 4.1, if $S(P, R, O) \neq \emptyset$ and $(m, o) \in S(P, R, O)$, then $(m, o) \subseteq (M'', O'')$. Axiom (S1) implies that $P \subseteq m$, which in turn implies that $P \subseteq M''$.

(ii) If $P \subseteq M''$, then according to Theorem 4.6, $(m^*, o^*) \in S(P, R, O)$.

Proof of Theorem 4.8

(i) Theorem 4.6 implies that $o^* \subseteq \bar{o}$.

(ii) $\bar{o} \subseteq O''$; thus, it follows from Axiom (S4) that if $y \in \bar{o}$, then there exists path $[y, z]$ such that $z \in P$. This implies that $\bar{o} \subseteq o^*$.

Proof of Theorem 4.9

The statements that if $x \in p$ in any iteration of loop lp7, then $\varphi^{-}(x) \subseteq o'$, that m' is the set of x where $x \in p$ in any iteration of loop lp7, and that if $y \in o'$, then $\Psi^{-}(\{y\}) \subseteq (m' \cup R)$ are trivially satisfied by sets m' and o'.

(i) It will be proved by induction on the number of vertices of the O-type in paths of (M'', O'') that if $y \in o^*$, then $y \in o'$, i.e. $o^* \subseteq o'$. Let us first examine the path $[y_1, x_1] = (y_1, x_1)$, where $x_1 \in p$ and $y_1 \in O''$. Since $x_1 \in p$ as defined in statement st6 of Fig. 7 and $y_1 \in \varphi^{-}(\{x_1\}) \subseteq o'$, $y_1 \in o'$. Suppose that for any path $[y_{n-1}, x_1] = (y_{n-1}, x_{n-1})|(x_{n-1}, y_{n-2})| \cdots |(x_2, y_1)|(y_1, x_1)$ of (M'', O''), $y_i \in o'$ $(i = 1, 2, \ldots, n-1)$ if $x_1 \in P$. Now, let $[y_n, x_1] = (y_n, x_n)|(x_n, y_{n-1})|[y_{n-1}, x_1]$ be a path in (M'', O'') and $x_1 \in P$. Since $y_{n-1} \in o'$, $\Psi^{-}(\{y_{n-1}\}) \subseteq (m' \cup R)$. Nevertheless, $x_n \in \Psi^{-}(\{y_{n-1}\})$ and $x_n \notin R$, and therefore, $x_n \in m'$. This implies that $\varphi^{-}(\{x_n\}) \subseteq o'$, but $y_n \in \varphi^{-}(\{x_n\})$ and thus, $y_n \in o'$.

(ii) At the outset of loop lp7, set o is empty and element $(\alpha, \beta) \in (M'', O'')$ has been substituted into o', if for an actual element of p, x is an element of β. For $o^* \supseteq o'$, it is sufficient to prove that for any element x of p in loop lp7, $x \in P$ or there exists path $[x, z]$ such that $z \in P$ or $x \in P$. At the initialization of loop lp7, this is trivially satisfied. Let us suppose that the preceding statement is satisfied in n steps of loop lp7. If $p \neq \emptyset$, then for $x \in p$, $o_x = \varphi^{-}(\{x\})$ and the possible new elements of p are the elements of $\Psi^{-}(o_x)$. But, for any element x' of $\Psi^{-}(o_x)$, there exists an $(\alpha, \beta) \in o_x (\subseteq o')$ such that $x' \in \alpha$ and $x \in \beta$. Consequently, there exists a path, $[x', z] = [x', x]|[x, z]$.

Proof of Theorem 4.10

From Theorem 4.7, there exists the maximal structure if $P \subseteq M''$. Suppose that $P \subseteq M''$; then $(\bar{m}, \bar{o}) \subseteq (M'', O'')$ according to Corollary 4.1. From Theorems 4.8 and 4.9, $\bar{o} = o^* = o'$ and $\bar{m} = m^* = m'$. This proof will be complete after it is proved that algorithm MSG terminates in a finite number of steps. The finiteness of the algorithm is proved in Section 5 along with an analysis of its complexity.