COMBINATORIAL ALGORITHMS FOR PROCESS SYNTHESIS

F. Friedler[1], K. Tarjan, Y.W. Huang, and L.T. Fan[2]

Laboratory for Artificial Intelligence in Process Engineering
Department of Chemical Engineering, Kansas State University
Manhattan, Kansas 66506, U.S.A.

ABSTRACT

Analysis of the combinatorial properties of process synthesis has been carried out in the present work. Such analysis has given rise to some efficient combinatorial algorithms. Algorithm MSG generates the maximal structure (super-structure) of a process synthesis problem; it can also be the basic algorithm in generating a mathematical programming model for this problem. Algorithm MSG is effective in synthesizing a large industrial process since its complexity grows merely polynomially with the size of the synthesized process. Another algorithm, algorithm SSG, generates the set of feasible process structures from the maximal structure; it leads to additional combinatorial algorithms of process synthesis including those for decomposition and for accelerating branch and bound search. These algorithms have also proved themselves to be efficient in solving large industrial synthesis problems.

KEYWORDS

Process synthesis; structure generation; maximal structure; combinatorial algorithm.

INTRODUCTION

The mathematical programming approach to process synthesis has two major steps, the generation of the mathematical model and the solution of this model. Nevertheless, the available methods for the first step are restricted to limited classes of homogeneous processes, and those for the second step are capable of solving only the models of relatively small synthesis problems. A homogeneous process comprises operating units of the same type, e.g., heat exchangers. Thus, the process synthesis methods resorting to mathematical programming are not sufficiently mature for industrial application.

Both steps of process synthesis have combinatorial aspects. In the first step, the connections of plausible operating units, i.e., some graph representation of the mathematical model, should be postulated, while in the second step, the model to be solved contains integer (combinatorial) variables. Process synthesis primarily is a combinatorial problem because the complexity of the synthesis is the consequence of its combinatorial nature, and the combinatorial variables affect the objective (cost) function more profoundly than the continuous variables of the model. Since, in practice, process synthesis cannot be separated into combinatorial and continuous parts, it should be solved by taking into account both parts simultaneously. The required combinatorial tools and algorithms for this purpose, however, have been unavailable so far. Process synthesis is defined here as the initial step of process design where the total flowsheet is to be generated (Siirola and Rudd, 1971; Mahalec and Motard, 1977; Douglas, 1988).

---

[1] On leave from Research Institute for Technical Chemistry, Hungarian Academy of Sciences, Veszprém.
[2] Author to whom correspondence should be addressed.

MATHEMATICAL FOUNDATION

Process Graphs

Simple graphs adopted in analyzing process systems are unsuitable for representing the structures of processes in their syntheses since the uniqueness of these graphs cannot be ascertained. For example, it has been found that a simple graph may belong to different processes (Friedler *et al.*, 1992). Hence, a special directed bipartite graph, termed process graph or P-graph in short, has been introduced to alleviate this difficulty.

Let M be a finite set, and let set O satisfy the constraint

$$O \subseteq \wp(M) \times \wp(M), \tag{1}$$

where $\wp(.)$ is the power set and $\times$ is the Cartesian product. Pair (M, O) is defined to be a process graph or P-graph; the set of vertices of this graph is $M \cup O$, and the set of arcs is a where

$$a=\{ (X, Y) \mid Y=(\alpha, \beta)\in O \text{ and } X\in\alpha \} \cup \{ (Y,X) \mid Y=(\alpha, \beta)\in O \text{ and } X\in\beta \}. \tag{2}$$

Since $M \cap O = \emptyset$ and no arc exists between two elements of M or those of O, a process graph is bipartite. The union and intersection of the two P-graphs, (m_1, o_1) and (m_2, o_2), are defined by (m_3, o_3) and (m_4, o_4), respectively, where $(m_3, o_3)=(m_1 \cup m_2, o_1 \cup o_2)$ and $(m_4, o_4)=(m_1 \cap m_2, o_1 \cap o_2)$. P-graph (m_1, o_1) is defined to be a subgraph of P-graph (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 arc $(X, Y) \in a$, then X and Y are said to be the initial and terminal endpoints, respectively, of this arc. If (α, β) is an element of O, then set α is the input-set of (α, β), while set β is its output-set. The input-set and output-set are subsets of M. An arc is defined to be incident into or out of a vertex if this vertex is the terminal or initial endpoint of this arc, respectively. The indegree, d^-, of vertex X is defined to be the number of the elements of the set of arcs incident into vertex X.

Process Structures

Let M be a given set of objects, usually material species or materials, that are transformed in the process under consideration. M can be expressed as a set of names or vectors of characteristics of these objects (materials). Transformation between two subsets of M occurs in an operating unit of the process, which is linked to other operating units of the process through the elements of these two subsets of M. If O is the set of operating units, it satisfies constraint (1). If (α, β) is an operating unit, then set α denotes its inputs while set β denotes its outputs. The structure of a process, given by sets M and O, is defined to be P-graph (M, O). The union and intersection operations of process structures are defined by their P-graph operations.

Example 1

Suppose that set M_1 of materials and set O_1 of operating units are given by $M_1=\{A, B, C, D, E, F, G, H, I, J\}$, and $O_1=\{ (\{B\}, \{A, E\}), (\{C\}, \{A, J\}), (\{D, E\}, \{B\}), (\{E, F\}, \{B\}), (\{F, G\}, \{C\}), (\{H\}, \{E\}), (\{I, J\}, \{G\}) \}$. It is not difficult to validate that sets M_1 and O_1 satisfy constraint (1), i.e., (M_1, O_1) is a P-graph, as depicted in Fig. 1.

Decision Mappings

To generate a certain class of substructures of a process structure, e.g., a set of feasible process structures, a special technique, decision mapping, is required to organize the system of decisions. Decision mapping is a special mathematical tool to render our decisions consistent and complete in dealing with complex decision problems, such as those encountered in process synthesis. The most essential definitions and theorems of decision-mappings have been listed here; further details and the proofs of theorems will be given elsewhere (Friedler *et al.*, 1991c).

Let us suppose that for finite sets M and O, $O\subseteq\wp(M)\times\wp(M)$ holds; for $X\in M$, let us define set o(X) by $o(X)=\{ (\alpha, \beta) \mid (\alpha, \beta)\in O \text{ and } X\in\beta \}$.
Definition. Let us suppose that set m is a subset of M. For $X\in m$, let $\delta(X)$ be a subset of o(X); then, $\delta[m]=\{ (X, \delta(X)) \mid X\in m \}$ is defined to be a *decision-mapping* on its domain m.
Definition. The *complement* of decision-mapping $\delta[m]$ is defined by $\delta^*[m]=\{ (X, Y) \mid X\in m \text{ and } Y=o(X)\backslash\delta(X) \}$. Thus, for $X\in m$, $\delta^*(X)=o(X)\backslash\delta(X)$.

Definition. Decision-mapping $\delta[m]$ is said to be *consistent* if $|m| \le 1$, or $(\delta(X) \cap \delta(Y)) \cup (\delta^*(X) \cap \delta^*(Y)) = o(X) \cap o(Y)$ for any X, Y∈m.
Definition. m' is said to be an *active domain* of decision mapping $\delta[m]$, if $m' \subseteq m$,

$$\bigcup_{X \in m'} \delta(X) = \bigcup_{X \in m} \delta(X), \text{ and } \bigcup_{X \in m'} \delta^*(X) = \bigcup_{X \in m} \delta^*(X). \quad (3\text{-}4)$$

Note that m is always an active domain of decision-mapping $\delta[m]$, and a decision mapping can have multiple active domains.

Decision-mapping of a P-graph. Let P-graph (m, o) be a subgraph of P-graph (M, O).
Definition. m' is an *active set* of P-graph (m, o), if $m' \subseteq m$ and $\beta \cap m' \neq \varnothing$ for any $(\alpha, \beta) \in o$.
Definition. Let m' be an active set of P-graph (m, o); then, $\delta[m']$ is defined to be a *decision-mapping of P-graph* (m, o), if

$$\delta[m'] = \{(X, Y) \mid X \in m' \text{ and } Y = \{(\alpha, \beta) \mid (\alpha, \beta) \in o \text{ and } X \in \beta\}\}. \quad (5)$$

Theorem. The decision-mappings of a P-graph are consistent.

P-graph of a decision-mapping. The definition of the P-graph of a decision-mapping is based on the following theorem.
Theorem. Let $\delta[m']$ be a consistent decision-mapping,

$$o = \bigcup_{X \in m'} \delta(X), \text{ and } m = \bigcup_{(\alpha, \beta) \in o} (\alpha \cup \beta). \quad (6\text{-}7)$$

Then, (m, o) is a P-graph, m' is an active set of P-graph (m, o), and $\delta[m']$ is a decision-mapping of P-graph (m, o).
Definition. The *P-graph of consistent decision-mapping* $\delta[m']$ is defined to be (m, o), where o and m are determined by formulas (6) and (7).

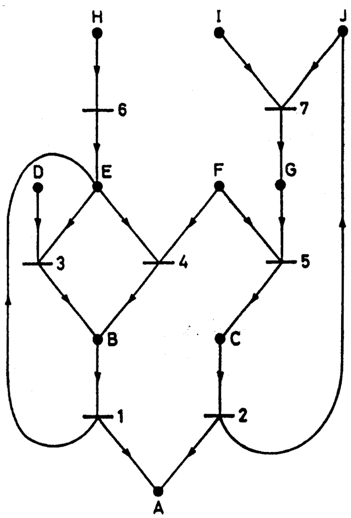

Fig. 1. P-graph (M_1, O_1).

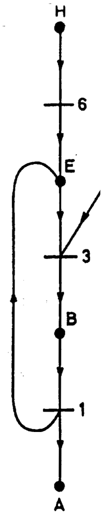

Fig. 2. P-graph representing decision-mapping δ_2.

Example 1 Revisited

Let decision-mapping δ_1 be defined by $\delta_1[\{A, B, E\}] = \{ (A, \{1\}), (B, \{3, 4\}), (E, \{6\}) \}$, where the operating units are labelled as in Fig. 1. Obviously, this decision-mapping is not consistent; therefore, it does not define a P-graph. However, decision-mapping $\delta_2[\{A, B, E\}] = \{ (A, \{1\}), (B, \{3\}), (E, \{1, 6\}) \}$ is consistent and defines P-graph (M_1', O_1'), where $M_1' = \{ A, B, D, E, H \}$, and $O_1' = \{ 1, 3, 6 \}$ (see Fig. 2.).

COMBINATORIAL PROPERTIES OF PROCESS STRUCTURES IN PROCESS SYNTHESIS

Suppose that the sets of the products, P, the raw materials, R, and the operating units, O, define synthesis problem (P, R, O). A set of axioms has been constructed to express the necessary and sufficient combinatorial properties to which a feasible process structure should conform (Friedler *et al.*, 1992). Structure (m, o) is such a feasible structure, i.e., solution-structure, of synthesis problem (P, R, O) if it satisfies axioms (S1) through (S5) given below.

(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 this 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 $y_1 \in P$; and
(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 $x \in (\alpha \cup \beta)$.

The set of solution-structures is denoted by S(P, R, O). An interesting property of this set is that it is closed under union.

The maximal structure, defined below, plays an important role in process synthesis.

Definition. The union of all solution-structures, μ(P, R, O), is defined to be its maximal structure, i.e.,

$$\mu(P,R,O) = \bigcup_{\sigma \in S(P,R,O)} \sigma \, . \tag{8}$$

The maximal structure of P-graphs corresponds to the "super-structure" of simple directed graphs; however, the former is mathematically defined rigorously, but the latter is not. Moreover, since each solution-structure is a substructure of the maximal structure, a solution-structure can be given by a decision-mapping of the maximal structure.

Table 1. Plausible Operating Units of Example 2.

No.	Type	Inputs	Outputs
1.	Feeder	A1	A5
2.	Reactor	A2, A3, A4	A9
3.	Reactor	A3, A4, A6, A11	A10
4.	Reactor	A3, A4, A5	A12
5.	Reactor	A3, A4, A5	A13
6.	Reactor	A7, A8, A14	A16
7.	Reactor	A8, A14, A18	A16
8.	Separator	A9, A11	A21, A22, A24
9.	Separator	A10, A11	A22, A24, A37
10.	Separator	A12	A25, A26
11.	Separator	A13	A25, A31
12.	Dissolver	A15, A16	A32
13.	Reactor	A14, A17, A18, A19, A20	A33
14.	Reactor	A6, A21	A35
15.	Washer	A22, A23	A48
16.	Washer	A5, A24	A36
17.	Separator	A5, A11, A25	A37, A38, A39
18.	Separator	A11, A26	A40, A42
19.	Reactor	A14, A27, A28, A29, A30	A41
20.	Separator	A11, A31	A40, A42
21.	Centrifuge	A32	A44, A45
22.	Washer	A33, A34	A46
23.	Separator	A36	A14, A48
24.	Separator	A38	A14, A48
25.	Filter	A41	A50, A51
26.	Washer	A43, A44	A53
27.	Filter	A46	A55, A56
28.	Separator	A47, A48	A5, A57
29.	Separator	A48, A49	A5, A58
30.	Separator	A50	A59, A60
31.	Dryer	A51, A54	A61
32.	Dryer	A52, A53	A61
33.	Dryer	A54, A55	A61
34.	Distillation	A59	A62, A63
35.	Separator	A60	A64, A65

Example 1 Revisited

Let $P_1 = \{A\}$ be the set of products, $R_1 = \{D, F, H, I\}$ be the set of raw materials; then, Fig. 2. represents a solution-structure, and Fig. 1. shows the maximal structure of synthesis problem (P_1, R_1, O_1).

Example 2. Synthesis of an Industrial Process

The Folpet (N-(trichloromethylthio)phthalimide) process is synthesized in this example. Although the synthesis of the total flowsheet has been carried out, only the combinatorial part of the synthesis is discussed here. Experimental investigations have given rise to a set of plausible operating units and a set of possible raw materials to produce a given product, A61, i.e., $P_2 = \{A61\}$. We have set $M_2 = \{A1, A2, A3, \ldots, A64, A65\}$ as the set of materials, and set $R_2 = \{A1, A2, A3, A4, A6, A7, A8, A11, A15, A17, A18, A19, A20, A23, A27, A28, A29, A30, A34, A43, A47, A49, A52, A54\}$ as the set of possible raw materials; O_2 is the set of plausible operating units listed in Table 1. P-graph (M_2, O_2) is not the maximal structure of synthesis problem (P_2, R_2, O_2). Since an operating unit, for example, operating unit # 14, does not satisfy axiom (S4), it can not be an element of neither a solution-structure nor the maximal structure. Without an algorithmic approach, it is very difficult to determine the maximal structure of this example or any other example of similar size. The maximal structure of Example 2 will be determined later by algorithm MSG.

ALGORITHMIC GENERATION OF THE MAXIMAL STRUCTURE

Algorithms given here have been written in Pidgin Algol. This high level language has been introduced by Aho *et al.* (1974) for describing algorithms for publication and mathematical examination.

Main Steps of Algorithm MSG

The algorithm for generating the maximal structure of synthesis problem (P, R, O) is presented

```
input: sets P, R, O;
comment: P ⊆ M, R ⊆ M, O ⊆ ℘(M) × ℘(M), and P ∩ R = ∅;
output: (m, o), the maximal structure of synthesis problem (P, R, O), if it exists;
      begin
st1:  O := O \ {(α,β) | (α,β) ∈ O & β ∩ R ≠ ∅};
st2:  M := ⋃_{(α,β)∈O} (α ∪ β);
st3:  r := {x | x ∈ M \ R & ∀(α,β) ∈ O, x ∉ β};
st4:  while r is not empty do
          begin
          let x be any element of r;
          M := M \ {x};
          o := {(α,β) | (α,β) ∈ O & x ∈ α};
          O := O \ o;
          r := (r ∪ {y | ∃(α,β) ∈ o such that y ∈ β & ∀(γ,δ) ∈ O, y ∉ δ}) \ {x}
          end
st5:  if P ⊄ M then stop; comment: there is no maximal structure;
      p := P; m := ∅; o := ∅;
st6:  while p is not empty do
          begin
          let x be any element of p;
          m := m ∪ {x};
          o := {(α,β) | (α,β) ∈ O & x ∈ β};
          o := o ∪ o;
          p := p ∪ (⋃_{(α,β)∈o} α) \ (R ∪ m);
          end
st7:  m := ⋃_{(α,β)∈o} (α ∪ β)
      end
```

Fig. 3. Algorithm MSG.

in Fig. 3. In this algorithm, statements st1 and st2 exclude operating units producing raw materials that violates axiom (S2). In statement st3, set r is defined as the set of materials that are not raw materials, but are consumed and never produced by any operating unit. In loop st4, these materials are excluded from set M, and the concomitant operating units are excluded from set O; the resulting P-graph (M, O) satisfies axiom (S2). Statement st5 examines if axiom (S1) is satisfied by P-graph (M, O). If this condition is not satisfied, the maximal structure does not exist. Otherwise, the maximal structure is constructed stepwisely by collecting the operating units satisfying axioms (S3) and (S4). Set M of materials defined by statement st7 assures that axiom (S5) is satisfied by P-graph (M, O). It has been proved that algorithm MSG always generates the maximal structure of the synthesis problem in a finite number of steps, if the maximal structure exists (Friedler *et al.*, 1991b).

Complexity Analysis of Algorithm MSG

It is often crucial to know the number of elementary steps required by a combinatorial algorithm as a function of the size of the problem under consideration. For the best combinatorial algorithms, this number can be bounded by a polynomial function. However, the complexity of most combinatorial algorithms is higher than polynomial, e.g., exponential, or factorial (see, e.g., Hartmanis, 1989). The complexity of algorithm MSG has been proved to be polynomial (Friedler *et al.*, 1991b).

Example 2 Revisited

The maximal structure of this example determined by algorithm MSG is given in Fig. 4. Five operating units in set O_2 do not belong to the maximal structure. As a result, the number of binary variables of the MINLP model of this example is reduced by five, thereby attaining the minimum.

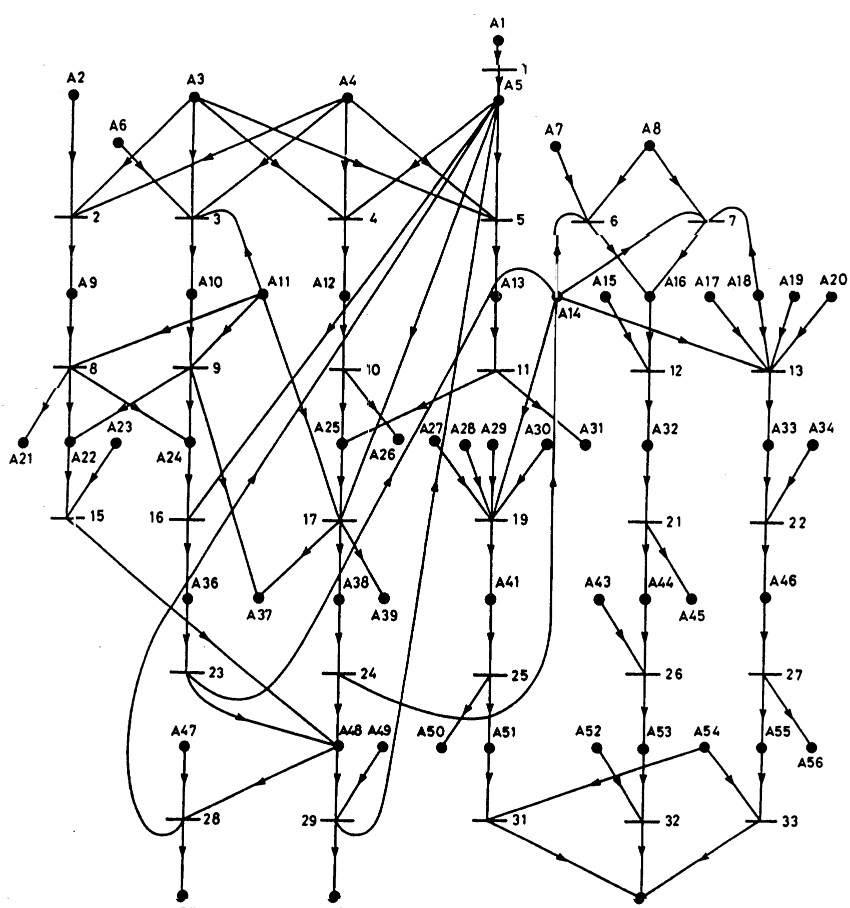

Fig. 4. Maximal structure of Example 2 generated by Algorithm MSG.

GENERATION OF THE SOLUTION-STRUCTURES

The axiom system containing axioms (S1) through (S5) defines the set of combinatorially feasible process structures, i.e., the set of solution-structures. Although the size of this set is usually excessively large to have its elements enumerated in practice, the availability of an algorithm to generate the set is essential. Such an algorithm constitutes the major building block for a mathematical programming approach to process synthesis, e.g., the accelerated branch and bound method of process synthesis (Friedler *et al.*, 1991a). The axiom system renders it possible to determine whether a P-graph represents a combinatorially feasible process structure; nevertheless, it is useless in directly generating the set of solution-structures. Thus, algorithm SSG has been developed for this purpose. This algorithm is based on the mathematical study of the axiom system and the solution-structures and is also based on the decision-mappings.

Algorithm SSG

Algorithm SSG, given in Fig. 5, is recursive because it invokes itself. This algorithm

```
input: P , R , M , o(x)  ( x ∈ M ) ;
comment: P , R , o(x) belong to synthesis problem ( P, R, O ), where
P ⊆ M , R ⊆ M , P ∩ R = ∅ , o(x) = { (α,β) | (α,β) ∈ O  &  x ∈ β } , o(x) = ∅ ⇔ x∈R,
δ[m] is a decision-mapping on M ;
output: all solution-structures of synthesis problem ( P, R, O ) ;
global variables:  R , o(x)  ( x ∈ M ) ;

begin
if  P = ∅   then  stop ;
SSG( P, ∅, ∅ ) ;
end

procedure SSG( p, m, δ[m] ):
begin
if p = ∅ then begin write δ[m] ; comment: δ[m] defines a solution-structure ; return end
let  x ∈ p ;
C := ℘ ( o(x) ) \ {∅} ;
For all c ∈ C  do
            begin
            if  ∀ y ∈ m,   c ∩ (o(y) \ δ(y)) = ∅    &    (o(x) \ c) ∩ δ(y) = ∅
               then
                  begin
                  δ[m∪{x}] := δ[m] ∪ {(x,c)} ;
                  SSG( p∪(     ∪      α ) \ (R∪m∪{x}) , m∪{x} , δ[m∪{x}] )
                           (α,β)∈c
                  end
            end
return
end
```

Fig. 5. Algorithm SSG.

determines the set of solution-structures as the decision-mappings of the maximal structure. In the list of parameters for procedure SSG, p is the set of materials that have not been but should be produced in the process partially defined by decision-mapping δ[m]. These parameters are updated recursively until all possible consistent extensions of δ[m] are examined. It has been proved that this algorithm generates each and every solution-structure exactly once, and it generates solution-structures only.

Example 2 Revisited

Algorithm SSG has generated all the 3465 different solution-structures of this industrial problem in less than 1 min. on a PC/AT; Fig. 6. shows one of them. This solution-structure represents the optimal process minimizing the cost function; it is determined by the accelerated branch and bound method based on algorithm SSG.

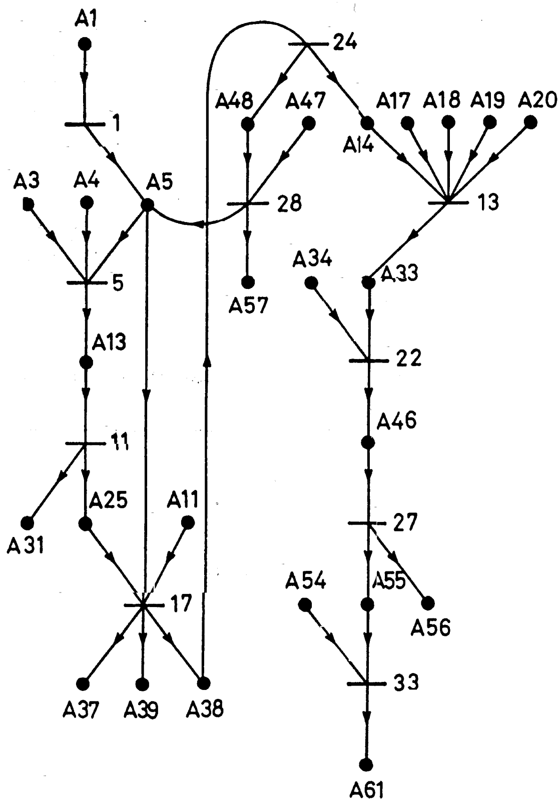

Fig. 6. Solution-structure of Example 2.

ACKNOWLEDGEMENT

Although the research in this article has been funded in part by the U.S. Environmental Protection Agency under assistance agreement R-815709 to Hazardous Substance Research Center for U.S. EPA Regions 7 and 8 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 Kansas State University Center for Hazardous Substance Research.

REFERENCES

Aho, A.V., J.E. Hopcroft and J.D. Ullmann (1974). *The Design and Analysis of Computer Algorithms*. Addison-Wesley, London.

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

Huang and L.T. Fan (1991a). An Accelerated Branch and Bound
hesis. Presented at the Fourth World Congress of Chemical
rmany, June 16-21; paper 12.2-9.

. Huang and L.T. Fan (1991b). Graph-Theoretic Approach to
nial Algorithm for Maximal Structure Generation. Submitted for
hem. Engng.

Huang and L.T. Fan, (1991c). Decision-Mapping: Representation
ecisions. Manuscript under preparation.

. Huang and L.T. Fan (1992). Graph-Theoretic Approach to
and Theorems. *Chem. Eng. Sci.* (in press)

'omputational Complexity Theory. Proceedings of Symposia in
AMS, Providence, RI.

(1977). Evolutionary Search for an Optimal Limiting Process
n. Engng., 1, 149.

971). Computer-Aided Synthesis of Chemical Process Designs.
0, 353.

Friedler, F., K. Tarjan, Y.W.
Method for Process Synt
Engineering, Karlsruhe, Ge

Friedler, F., K. Tarjan, Y.W
Process Synthesis: Polynor
publication to *Computers c*

Friedler, F., K. Tarjan, Y.W.
of Systems of Consistent D

Friedler, F., K. Tarjan, Y.W
Process Synthesis: Axioms

Hartmanis, J. (Ed.) (1989). *C*
Applied Mathematics, 38, A

Mahalec, V. and R.L. Motard
Flowsheet. *Computers chen*

Siirola, J.J. and D.F. Rudd (1
Ind. Eng. Chem. Fundam. 1

Computers chem. Engng, Vol. 16, Suppl., pp. S1–S548, 1992
Printed in Great Britain.
0098-1354/92 $5.00+0.00

European Symposium on Computer Aided Process Engineering—1

ESCAPE-1

24–28 May 1992 Elsinore, Denmark

23rd European Symposium of the Working Party on Computer Aided Process Engineering

463rd Event of the European Federation of Chemical Engineers (EFChE)

Editor
R. GANI
Technical University of Denmark
Lyngby, Denmark

PERGAMON PRESS
Oxford · New York · Seoul · Tokyo