ELSEVIER

Computers and Chemical Engineering 24 (2000) 1881–1900

Computers & Chemical Engineering

www.elsevier.com/locate/compchemeng

Separation-network synthesis: global optimum through rigorous super-structure

Z. Kovács [a,c], Z. Ercsey [a], F. Friedler [a,b,*], L.T. Fan [b]

[a] *Department of Computer Science, University of Veszprém, Veszprém, Egyetem u. 10., H-8200, Hungary*
[b] *Department of Chemical Engineering, Kansas State University, Manhattan, KS 66506, USA*
[c] *Department of Computer Science, József Attila University, Szeged, Árpád tér 2, H-6720, Hungary*

Received 12 February 2000; received in revised form 24 February 2000; accepted 20 April 2000

Abstract

The available algorithmic methods often fail to yield with certainty the global optima in solving even a relatively simple class of separation-network synthesis problem for which the cost functions are considered to be linear. This is attributable to two complications; firstly the super-structures on which the solutions are based are incomplete; and the secondly, the mathematical programming models derived for the problems are unnecessarily cumbersome. To circumvent these complications, a novel method is proposed here to generate the complete super-structure and the corresponding mathematical programming model necessary for the separation-network synthesis problem with linear cost function. The efficacy of the proposed method is demonstrated by re-examining four published problems for which the optima obtained are claimed to be global. For all the problems re-examined, the costs of the solutions resulting from the present method are the same or as much as 30% lower than those of the published solutions.

Keywords: Process synthesis; Separation network; Rigorous super-structure; Mathematical programming

1. Introduction

The optimal solution of a separation-network synthesis (SNS) problem requires an appropriate mathematical programming model and a global optimization method; solving the former by the latter gives rise to the optimal solution of the original problem. Apparently, no method is available for exactly constructing such a model although some effective global optimization methods have become available recently.

Various unexpected optimal solutions obtained for some simple classes of SNS problems illustrate the difficulty involved in generating a valid mathematical programming model for a process-synthesis problem in general and for a SNS problem in particular (see e.g. Kovács, Friedler & Fan, 1993; Kovács, Ercsey, Friedler & Fan, 1998). The present paper proposes an efficient method to resolve this situation; specifically, it focuses on the solution of a relatively simple class of SNS problems for which the cost functions of the operating units are linear without fixed charges. Unlike the available algorithmic methods not involving model generation for synthesis, the proposed method solves algorithmically every step from the model generation to the optimal solution. The model generation is carried out by a novel systematic technique; moreover, the resultant model is rigorously verified. The efficacy of the present method is illustrated with published examples for ease of comparison.

2. Rigorous super-structure

First, an outline is given of the basic requirements for an algorithmic process-synthesis method and of the concomitant super-structure. This is followed by the mathematical definition of the latter and a description of an approach for systematically generating it.

* Corresponding author. Tel.: +36-88-424483; fax: +36-88-428275.
E-mail address: friedler@dcs.vein.hu (F. Friedler).

0098-1354/00/$ - see front matter
PII: S0098-1354(00)00568-8

2.1. Super-structure

An algorithmic method of SNS should comprise two major steps: (i) The construction of a mathematical model involving the generation of both the complete network structure, i.e. super-structure, and the corresponding mathematical programming model; and (ii) the solution of the resultant mathematical programming model, i.e. the determination of the optimal structure by means of an optimization method. If the mathematical programming model is obtained from an incomplete network structure, it may not yield the optimal solution regardless of the optimization method adopted. In principle, the super-structure may be generated if all potentially optimal structures are known either explicitly or implicitly. Practically speaking, however, this super-structure is unnecessary, if all the potentially optimal structures are explicitly known. Nevertheless, generating them can be exceedingly difficult since the number of different structures that need to be taken into account in solving the problem can be excessive even for the simplest class of problems (see e.g. Thompson & King, 1972). It is, therefore, essential that an algorithm be available for generating the super-structure from the potentially optimal structures that are given only implicitly (see e.g. Friedler, Tarján, Huang & Fan, 1993).

Two vital aspects have long been neglected in solving the problems of separation-network synthesis on the basis of the super-structure. These are: (i) the failure to define mathematically the super-structure; and (ii) the failure to establish a method of analysis to determine if any specific super-structure contains the optimal structures for all instances of a given class of SNS problems. To mitigate these deficiencies, the definition of the rigorous super-structure is introduced below.

Definition. Let a set of operating units and the mathematical model of each operating unit be given. Moreover, a systematic procedure is presumed to be available so that a valid mathematical programming model can be generated for a network of the given operating units. Then, this network is deemed to be a rigorous super-structure for a class of process-synthesis problems if the optimality of the resultant solution cannot be improved for any instance of the class of problems by any other network of operating units and model generation procedure.

The term, rigorous super-structure, is coined to distinguish it from the super-structure which hitherto lacks an exact definition. The optimal solution, therefore, can now be determined with certainty if the mathematical programming model is generated from this rigorous super-structure, and the solution procedure is capable of solving the model. In other words, the generation of the rigorous super-structure should be the first critical step of SNS.

A rigorous super-structure is not unique since two different super-structures may lead to an identical optimal solution for any instance of a class of SNS problems. One of the reasons is the following. Suppose that a rigorous super-structure includes every potentially optimal structure for each instance of a given class of SNS problems; then, every other structure containing this rigorous super-structure also contains every potentially optimal structure. Even for a simple class of SNS problems, it is difficult to determine the rigorous super-structure containing only those operating units and linkages belonging to the optimal structure for at least one instance.

2.2. Rigorous super-structure generation for a class of SNS problems

Since the rigorous super-structure may depend on the type of cost functions of the operating units, we limit our consideration to the following specific class of SNS problems.

A set of multicomponent product-streams is to be generated from multicomponent feed-streams by a network of separators, dividers, and mixers. The models and cost functions of the operating units are as follows:

The components of the streams are in a ranked list, which remains invariant over the entire separation process. Let n be the number of components considered in the separation-network synthesis problem; then, a stream is represented by an n-vector of nonnegative real numbers. It is assumed that the vector representing a feed-stream of the synthesis problem does not contain zero elements between nonzero elements. The number of components of a feed-stream is k, if the number of nonzero elements of the vector representing it is k. Separator S^i performs a sharp separation between components i and $(i+1)$ of its inlet stream, i.e. if its inlet stream contains components 1 through n, then its top outlet stream contains components 1 through i, and its bottom outlet stream, components $(i+1)$ through n. If more than one separator of an identical type appear in the network of interest, they are distinguishable by their subscripts. To facilitate the comparison with the earlier works, the cost of a separator is assumed to be its mass load multiplied by the degree of difficulty of the separation; and the overall cost of the separation-network, to be the sum of the costs of individual separators in the network. In addition, the cost of the dividers and mixers in the separation-network are regarded zero because their contribution to the overall cost is negligible: they are far less costly than the separators. This type of model has been adopted by several authors, e.g. Floudas (1987), Wehe and Westerberg (1987) and Quesada and Grossmann (1995).

Notation

A	Set of directed arcs (a,b), where $a,b \in N$. The direction of the flow represented by arc (a,b) is from a to b.
D	Finite set of dividers.
F	Set of feed-streams where $f \in F$ is an n-vector of nonnegative real values; an input to algorithm SNS-LMSG.
$G(N,A)$	Network where N is the set of nodes, and A is the set of directed arcs; an output from algorithm SNS-LMSG.
M	Finite set of mixers.
N	$N=F \cup P \cup S \cup D \cup M$
P	Set of product-streams where $p \in P$ is an n-vector of nonnegative real values with its i-th component represented by p^i; an input to algorithm SNS-LMSG.
S	Finite set of separators.
TD	Temporary set of dividers.
δ_{fd}	Operator assuming the value of 1 if there exists a path between feed-stream f and divider d; otherwise, 0.
δ^i_{fd}	Operator assuming the value of 1 if there exists a path between feed-stream f and divider d such that every stream of the path contains component i; otherwise, 0.

```
begin
    S = ∅; D = ∅; M = ∅; A = ∅; TD = ∅; I = {1,2,...,n-1};
    for all f ∈ F do begin
        Let d be a new divider
        TD = TD ∪ {d}
        D = D ∪ {d}
        A = A ∪ {(f,d)}
    end
    for all p ∈ P do begin
        Let m be a new mixer
        M = M ∪ {m}
        A = A ∪ {(m,p)}
    end
    while TD ≠ ∅ do begin
        Let d be an element of TD
        TD = TD \ {d}
        Let f be an element of F such that δ_fd = 1
        for all i ∈ I do begin
            if δ^i_fd = 1  and  δ^(i+1)_fd = 1
            then begin
                Let s be a new separator
                S = S ∪ {s}
                Let type of s be i
                A = A ∪ {(d,s)}
                Let d1 and d2 be new dividers
                TD = TD ∪ {d1, d2}
                D = D ∪ {d1, d2}
                A = A ∪ {(s,d1), (s,d2)}
            end
        end
        for all p ∈ P do begin
            if ∀i ∈ I  (if δ^i_fd = 1 then p^i > 0)
            then begin
                Let m be an element of M such that (m,p) ∈ A
                A = A ∪ {(d,m)}
            end
        end
    end
end
```

Fig. 1. Algorithm SNS-LMSG.

To ascertain that a super-structure contains only the necessary operating units and linkages, SNS problems must be examined from the structural point of view. This may lead to the determination of basic structural properties of potentially optimal structures. The following theorem plays a key role in generating a rigorous super-structure to the specific class of SNS problems under consideration.

Theorem 1. *There exists a loopless optimal network for any separation-network synthesis problem with simple and sharp separators, dividers, and mixers, where the cost of a network is the sum of the separators' costs, each of which is proportional to its mass load (see Appendix A for the proof).*

A rigorous super-structure for the class of SNS problems described above can be generated by algorithm SNS-LMSG derivated from Theorem 1 (see Fig. 1). A step-by-step illustration of the algorithm is given in Appendix B. The rigorous super-structure is formed from the structure generated for each feed-stream as follows:

As an initialization of the algorithm, a divider is created to every feed-stream, and each feed-stream is linked with the corresponding divider; moreover, a mixer is created to each product-stream, and the outlet stream from the mixer is linked with the corresponding product-stream. Then, the algorithm considers every divider generated in the structure. One of all types of separators performing a separation between any pair of adjacent components in the inlet stream to the divider

is created, and an outlet stream from the divider is linked with the separators created. Subsequently, dividers are created for each outlet stream of the separators created. Moreover, outlet streams from the divider under consideration are linked to the mixers of the product-streams if plausible. This procedure is repeated until every divider is processed, i.e. single component streams are produced from the feed-stream and bypassed to the product-streams. In other words, any stream is linked to every separator performing a separation on this stream; moreover, it is also linked to the mixers assigned to those product-streams that contain every component of the stream; obviously, no other connection is needed. This can be visualized through the following: a separator performing no separation on the given stream; connecting it to the network simply increases the cost of the overall network. Note that mixers are assigned only to product-streams (see the corollary of Theorem 1 in Appendix A).

Theorem 2. *Algorithm SNS-LMSG generates a rigorous super-structure for any SNS problem with simple and sharp separators, dividers and mixers in a finite number of steps where the cost of a network is the sum of the separators' costs, each of which is proportional to its mass load (see Appendix A for the proof).*

3. Mathematical programming model

The mathematical programming model derived from the rigorous super-structure should be as simple as possible without impairing the optimality of the resultant solution. In any of the available algorithmic methods for SNS, the model derived leads to a nonlinear mathematical programming problem even when the objective function is linear (see, e.g. Quesada & Grossmann, 1995). This nonlinear programming model is formulated in terms of the compositions of each stream. As such, a substantial number of nonconvex terms appear in the expressions of mass balance in the model. Another alternative is to formulate the model in terms of the flows of individual components, where the nonconvexities arise only in the models of dividers. The resultant model is solved, e.g. with a conventional NLP algorithm (Floudas, 1987), Benders decomposition (Floudas & Aggarwal, 1990), or a reformulation-linearization technique (Quesada & Grossmann, 1995).

A linear programming model has been successfully formulated in the present work, which at the very least yields a solution identical with those obtained by the more complex nonlinear models mentioned above if each model is based on a rigorous super-structure. Naturally, the present model can be solved with relative ease to yield the global optimal solution with certainty.

The linear programming model has been formulated in the following manner.

Let F denote the set of feed-streams; P, the set of product-streams; D, the set of dividers; M, the set of mixers; S, the set of separators; and A, the set of linkages among the operating units, i.e. the streams in the super-structure. Variables x's, i.e. the splitting ratios, are assigned only to the streams emerging from dividers of the super-structure. Each x signifies the fraction of the feed-stream to the network that is diverted to a stream exiting from a divider. The given mass of any component remaining in a stream between two successive dividers cannot be dispersed by sharp separators; moreover, no mixer exits between these two dividers in the super-structure constructed rigorously according to algorithm SNS-LMSG. Thus, the x assigned to the divider in front can be associated with any stream between the two dividers; in fact, the x corresponds to the ratio between a component contained in this stream and that in the feed-stream to the entire network; see Fig. 2. To illustrate numerically, suppose that the flow rates or amounts of the six components in

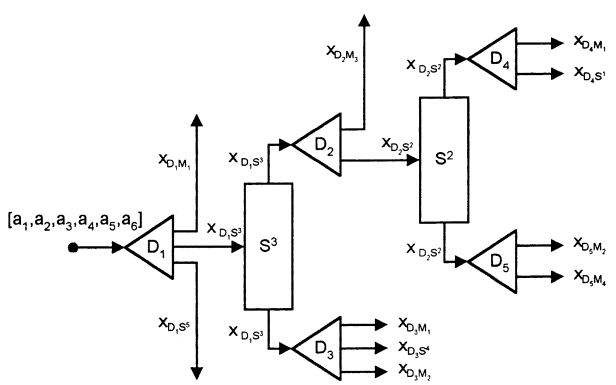

Fig. 2. Illustration of the splitting ratio.

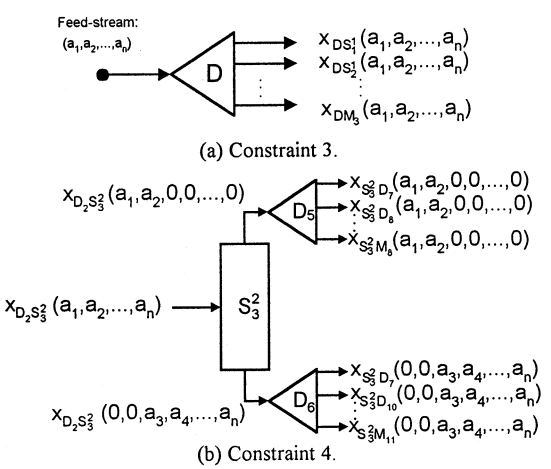

Fig. 3. Illustration of constraints 3 and 4.

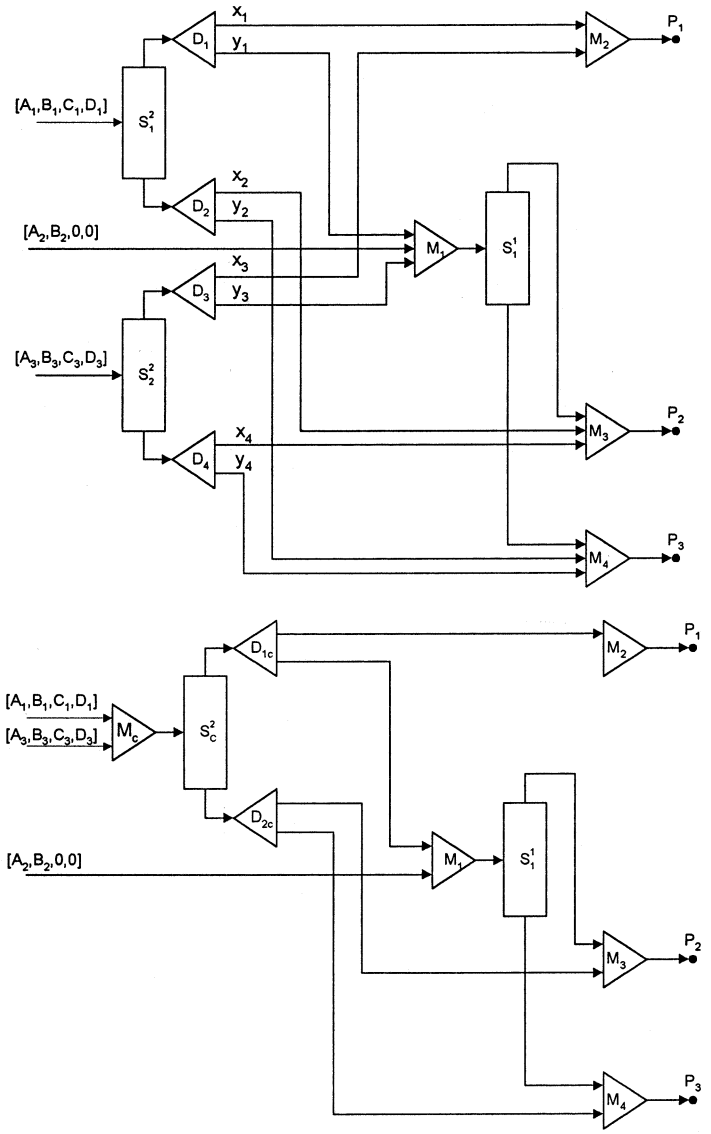

Fig. 4. Illustration of merging separators. (a) Network prior to merging seperators identical in type and with the same cost function. (b) Network resulting from merging the seperators as indicated in (a).

the feed-stream in Fig. 2 are [12, 8, 10, 4, 8, 6], thereby giving rise to the total flow rate of 48. If $x_{D_1M_1} = {}^1/_4$, $x_{D_1S^3} = {}^1/_2$, and $x_{D_1S^5} = {}^1/_4$, the flow rates of the streams around divider D_1 are balanced as given below.

[12, 8, 10, 4, 8, 6]

$$= \frac{1}{4} \cdot [12, 8, 10, 4, 8, 6] + \frac{1}{2} \cdot [12, 8, 10, 4, 8, 6]$$

$$+ \frac{1}{4} \cdot [12, 8, 10, 4, 8, 6]$$

Similarly, the flow rates of the streams around separator S^3 are balanced as shown below:

$$\frac{1}{2} \cdot [12, 8, 10, 4, 8, 6]$$

$$= \frac{1}{2} \cdot [12, 8, 10, 0, 0, 0] + \frac{1}{2} \cdot [0, 0, 0, 4, 8, 6]$$

Now, suppose that the top-outlet stream of separator S^3 is split into 2 by divider D_2, ${}^1/_3$ is being bypassed to mixer M_3 and the remaining ${}^2/_3$ being directly fed to separator S^2. Then, we have:

$$x_{D_2M_3} = \frac{1}{2} \cdot \frac{1}{3} = \frac{1}{6}$$

and

Table 1
Data for Example 1

Component	A	B	C
Feed-stream	10	10	10
Product-stream 1	6	4	2
Product-stream 2	4	6	8

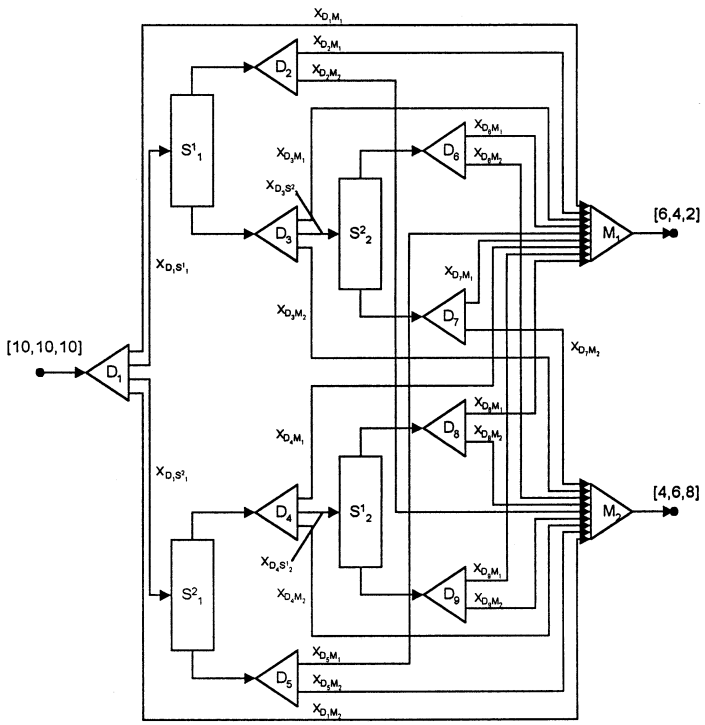

Fig. 5. Rigorous super-structure of Example 1.

$$x_{D_2S^2}=\frac{1}{2}\cdot\frac{2}{3}=\frac{2}{6}.$$

Consequently, the flow rates of the streams around divider D^2 are balanced as

$$\frac{1}{2}\cdot[12, 8, 10, 0, 0, 0]$$

$$=\frac{1}{6}\cdot[12, 8, 10, 0, 0, 0]+\frac{2}{6}\cdot[12, 8, 10, 0, 0, 0]$$

and those around S^2 are balanced as

$$\frac{2}{6}\cdot[12, 8, 10, 0, 0, 0]$$

$$=\frac{2}{6}\cdot[12, 8, 0, 0, 0, 0]+\frac{2}{6}\cdot[0, 0, 10, 0, 0, 0].$$

The resultant linear programming model is given in the following.

$$\min\sum_{i\in S}\left(d_i\sum_{\{j:(j,i)\in A\}}\left(x_{ji}\sum_{c=1}^{n}\sum_{k\in F}\delta_{ki}^{c}f_k^{c}\right)\right) \tag{1}$$

subject to

$$0\le x_{ij}\qquad\forall(i,j)\in A\quad\text{where}\quad i\in D \tag{2}$$

$$\sum_{\{j:(i,j)\in A\}}x_{ij}=1$$

$$\forall i\in D\quad\text{where}\quad\exists l\in F\ \text{such that}\ (l,i)\in A \tag{3}$$

$$\sum_{\{j:(i,j)\in A\}}x_{ij}=x_{kl}$$

$$\forall(k,l)\in A\quad\text{where}\ \exists i\in D\ \text{such that}\ (l,i)\in A \tag{4}$$

$$p_i^c=\sum_{\{(l,j):(j,i)\in A\}}\left(x_{lj}\sum_{k\in F}\delta_{ki}^{c}f_k^{c}\right)$$

$$\forall i\in P\ \text{and}\ c=1,2\ldots,n \tag{5}$$

$$\delta_{ki}^{c}=\begin{cases}1 & \text{there is a path from node } k \text{ to node } i \text{ with component } c\\ 0 & \text{otherwise}\end{cases} \tag{6}$$

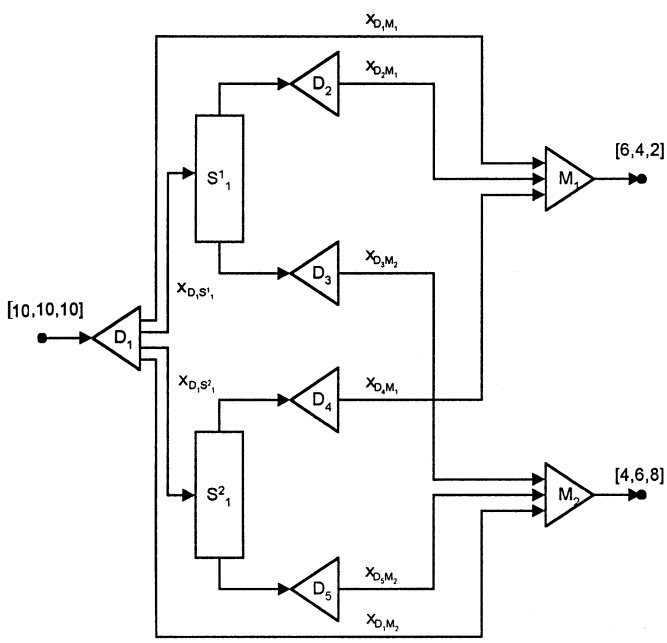

Fig. 6. Optimal structure for example 1 (cost = 12.00).

Table 2
Data for Example 2

Component	A	B	C	D
Feed-stream 1	6	4	0	0
Feed-stream 2	8	6	10	6
Feed-stream 3	0	0	5	5
Degree of difficulty	4	1.5	4	

Product	Sum of components	Component information			
Product-stream 1	15	A≥9	B≤3	C≤3	D=0
Product-stream 2	20	B≥7	C≥7	B=C	
Product-stream 3	15	D≥9	A=0		

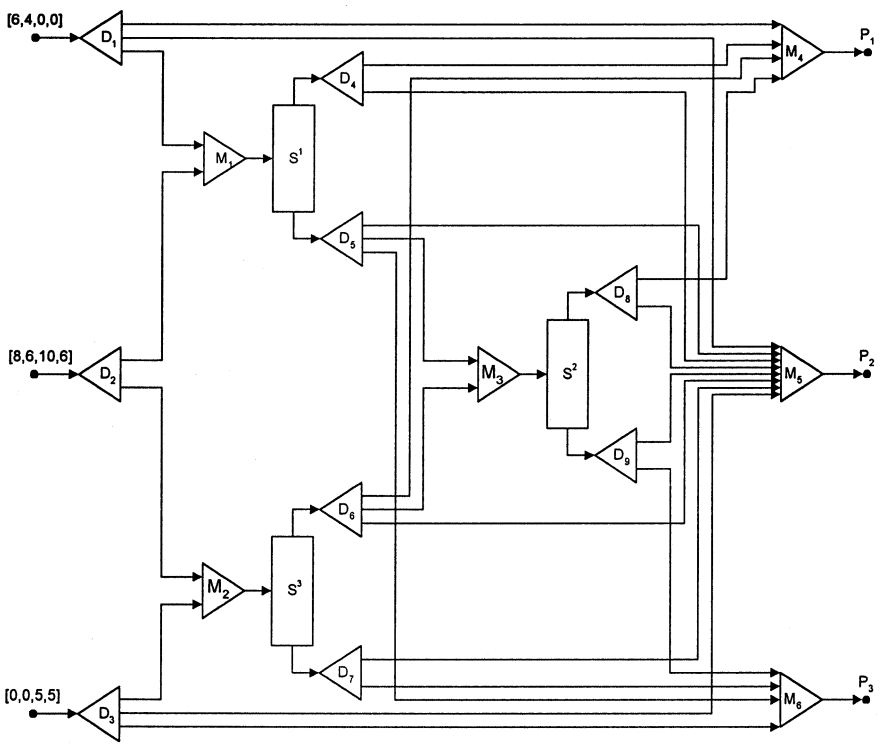

Fig. 7. Super-structure published in Quesada and Grossmann (1995) for Example 2.

In the objective function, expression (1), d_i denotes the degree of difficulty of the separation in separator i. Note that the components in the inlet stream of each separator are determined when the super-structure is generated by algorithm SNS-LMSG; hence, the different degrees of difficulty of the separation can be assigned to separators identical in type but containing different non-key components. For instance, the degree of difficulty for performing separation A/B is, in reality, greater than that for performing separation A/BCD because the relative volatility of A in the latter is greater than that in the former due to the non-key heavy components in the mixture. In the objective function, x_{ji} denotes the variable belonging to the inlet stream to separator i, i.e. the variable belonging to the stream between divider j and separator i; and f_k^c, the amount of component c in feed-stream k. Note that the component composition of every stream is described by the variables, i.e. splitting ratios, x's. Constraints (2)–(4) pertain to the splitting ratios of the model. In constraint (2), a natural assumption for the streams is given; x_{ij} denotes the variable belonging to the outlet streams of the dividers. Constraint (3) a mass balance equation on the splitting ratios of the dividers that are assigned to feed-streams is given; i.e. x_{ij}'s are the variables of the outlet streams from divider i belonging to feed-stream l; it is illustrated in Fig. 3(a). In constraint (4), a mass-balance equation on the splitting ratios of the dividers that are assigned to separators is given; i.e. x_{ij}'s are the variables of the outlet streams from divider

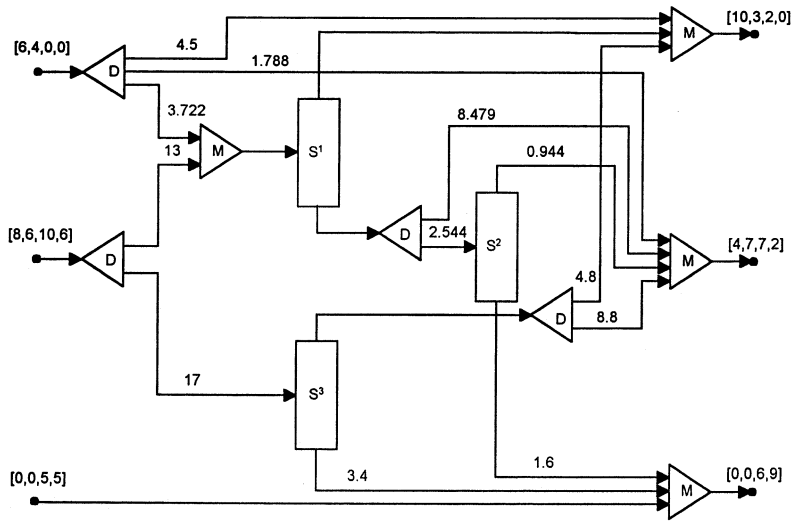

Fig. 8. Solution published in Quesada and Grossmann (1995) for Example 2 (cost = 138.18).

i belonging to separator l, and x_{kl}, the variable of the inlet stream to separator l; it is illustrated in Fig. 3(b). Constraint (5) results from the mass balance of the product-streams, where p_i^c denotes the amount of component c of product-stream i; f_k^c, component c in feed-stream k; and x_{lj}, variables belonging to input streams of mixer j assigned to product-stream i. Constraint (6) refers to the connections between the operating units appearing in the super-structure, where $\delta_{ki}^c = 1$ indicates that each stream, i.e. each arc of the path, contains component c.

The mathematical programming model given in expressions (1)–(6) has the following properties. When a feed-stream is split, the sum of variables x's, i.e. the splitting ratio, must be equal to one. When an intermediate stream between a pair of the feed-stream and a product-stream is split, the sum of the splitting ratios must be equal to the splitting ratio of the inlet stream to the separator to which the divider is assigned. This mathematical programming model is also capable of handling SNS problems where the component compositions of the product-streams are not explicitly given, but given as a set of inequality constraints. This requires a slight modification of the constraints imposed on the splitting ratios and to the product-streams; specifically, the equality constraints are relaxed to inequality constraints. The present mathematical programming model is strictly linear; it can be algorithmically generated from the rigorous super-structure. Thus, the model always yields the optimal solution.

Any of the optimal separation-networks obtained in the SNS problems under consideration may include two or more separators identical in type and with the same cost function. Obviously, some of such separators can be merged, thereby reducing their number; this practice

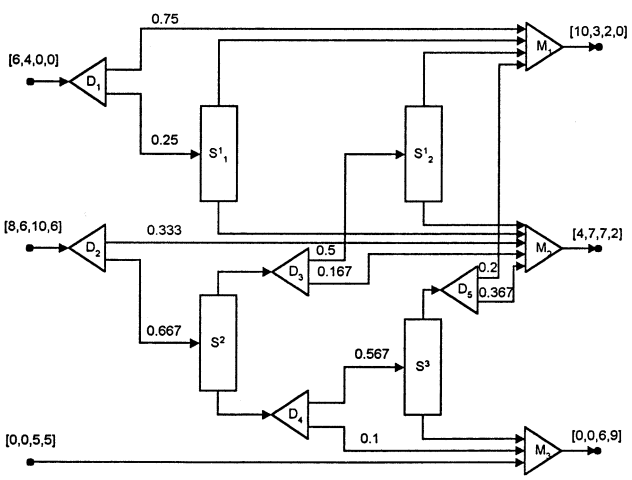

Fig. 9. Optimal structure for Example 2 (cost = 104.26).

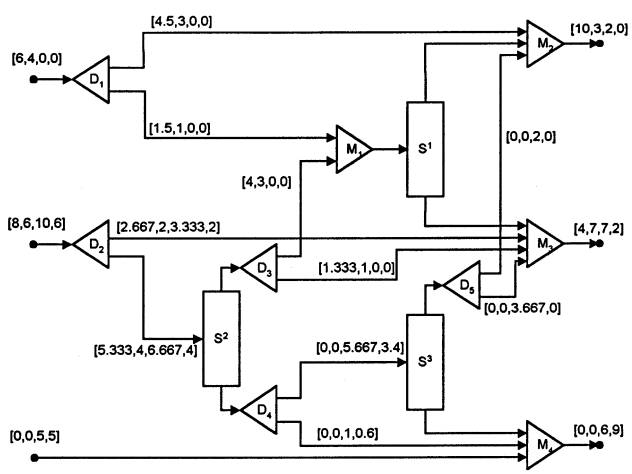

Fig. 10. Optimal structure for Example 2 with combined seperators (cost = 104.26).

Table 3
Data for Example 3

Component	A	B	C	D
Feed-stream	15	20	10	15
Product-stream 1	5	10	4	10
Product-stream 2	10	10	6	5
Degree of difficulty	2.5	3.0	1.5	

should be implemented whenever it does not increase the network's cost. The resultant network is naturally also optimal while the number of its separators and consequently, its complexity are reduced. Specifically, two separators identical in type should be merged and replaced with a single separator if (i) they have the same cost function, i.e. their degrees of difficulty of the separation are identical; (ii) the dividers for the top-outlet streams of the separators are linked to the same mixers with identical splitting ratios; and (iii) the dividers for the bottom-outlet streams of the separators are linked to the same mixers with identical splitting ratios.

Naturally, the flow rates of the resultant separator's inlet and outlet streams are equal to the sum of the flow rates of the two merged separators' inlet and outlet streams, respectively. Moreover, this tends to alter the numbers of mixers and dividers and their linkages in the network. For example, consider separators S_1^2 and S_2^2 in Fig. 4(a); components A, B, C, and D are in both separators' inlet streams; moreover, they are of the type 2, separating between components A and B, and therefore, their degrees of difficulty of the separation are identical. Their bottom-outlet streams are linked to the dividers; the streams exiting from these dividers are subsequently linked to the mixers assigned to product streams P_2 and P_3. In addition, all the top-outlet streams from both separators are linked to the dividers whose exiting streams are, in turn, linked to the same mixers, i.e. the mixers assigned to separator S_1^1 and product stream P_1 with the specific splitting ratio. These separators, i.e. separators S_1^2 and S_2^2, can only be combined if the corresponding splitting ratios are identical, i.e. if $x_1/y_1 = x_3/y_3$ and $x_2/y_2 = x_4/y_4$. Under any other circumstance, these separators must not be merged if the optimality of the solution is to be maintained. The optimal network resulting from the merging is illustrated in Fig. 4(b), where separator S_C^2 represents the combined separator. Note that the flow rate of the inlet stream to this separator is equal to the sum of the flow rates of the inlet streams to the two separators merged and that M_c represents a new mixer for the inlet

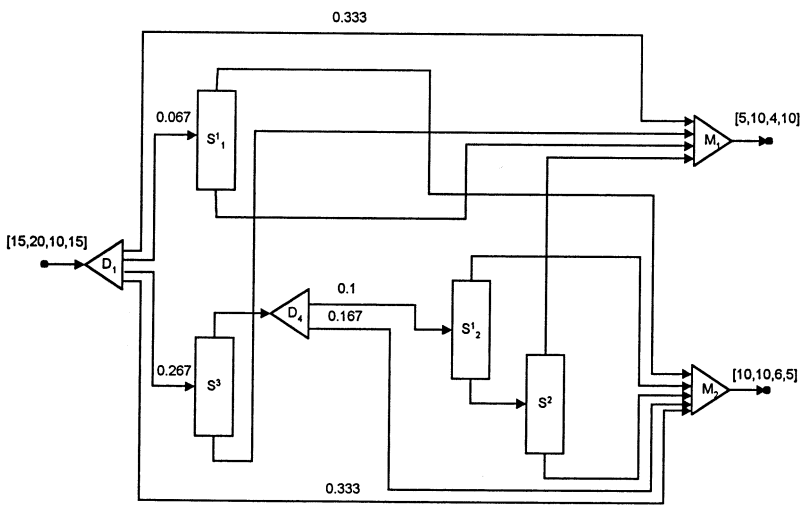

Fig. 11. Optimal structure for Example 3 (cost = 54.25).

Table 4
Data for Example 4

Component	A	B	C	D	E	F
Feed-stream	23	19	25	21	26	26
Product-stream 1	3	2	6	8	4	10
Product-stream 2	8	10	8	8	6	5
Product-stream 3	5	4	10	3	11	4
Product-stream 4	7	3	1	2	5	7
Degree of difficulty	1.5	3.0	2.0	2.5	4.0	

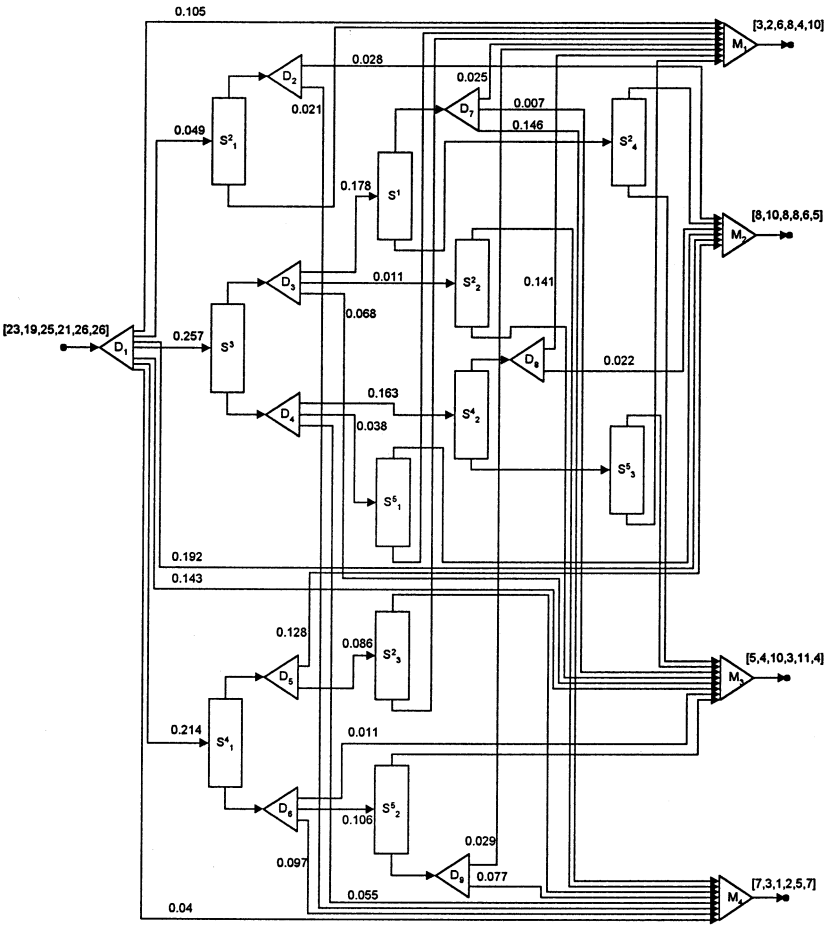

Fig. 12. Optimal structure for Example 4 (cost = 330.76).

stream, and D_{1c} and D_{2c} are new dividers for the outlet streams of the new separator.

A software package has been developed to illustrate the present method; it is available at http://www.dcs.vein.hu/capo/demo. The software includes the generation of the rigorous super-structure, the mathematical programming model, and the solution of the model.

4. Examples

Four examples taken from the literature (Quesada & Grossmann, 1995) are elaborated in detail to explicitly and clearly illustrate the present method. The results of two of the examples are superior to the previously known best solutions, and those of the remaining examples are slightly better or equivalent in terms of the objective function.

4.1. Example 1

Suppose that a three-component equimolar feed-stream is to be separated into two multicomponent

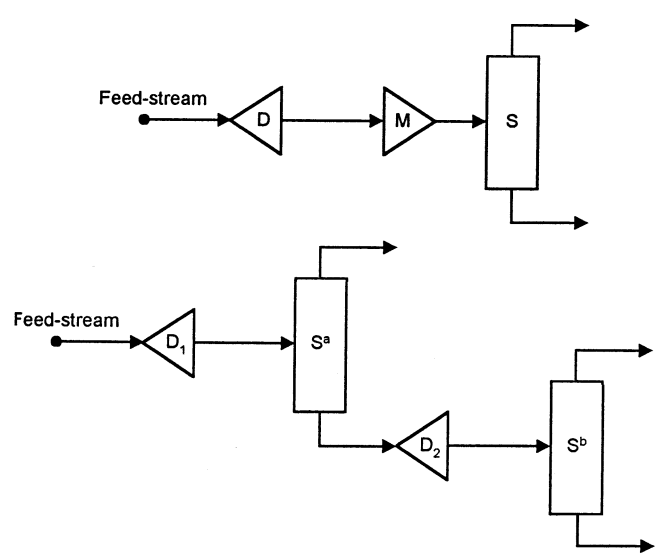

Fig. 13. Examples of a sample and a non-simple path. (a) Simple path: from Feed-stream to S. (b) Non-simple path: from feed-stream to S^b.

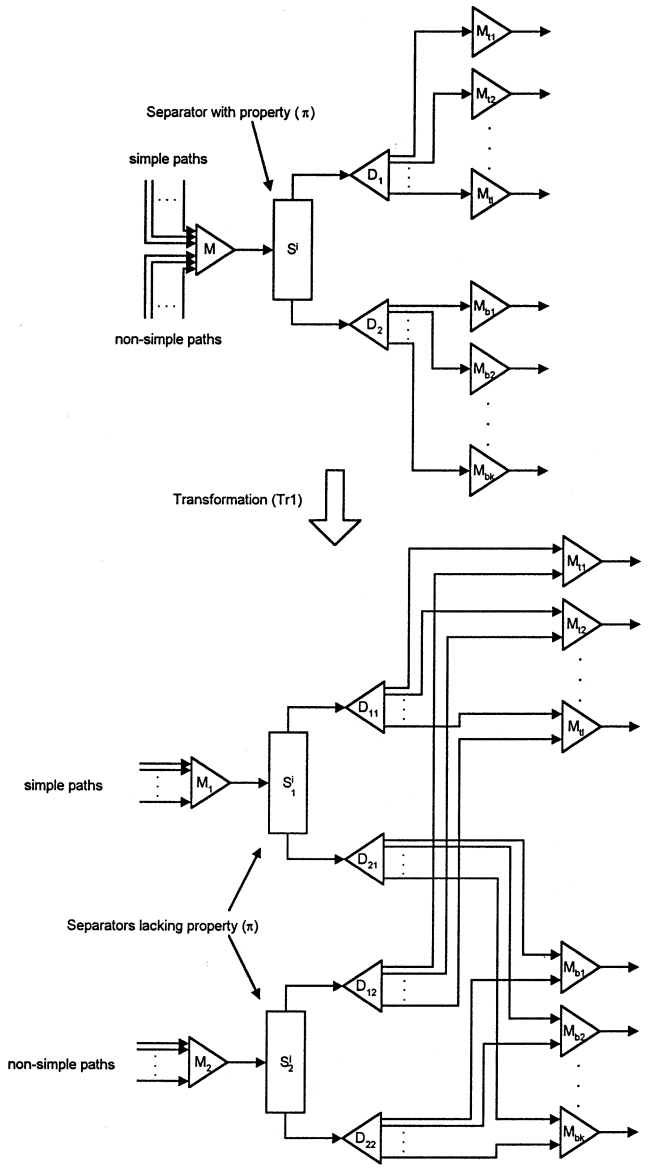

Fig. 14. Transformation (Tr1).

product-streams of different compositions as specified in Quesada and Grossmann (1995) where the objective function to be minimized is the sum of the total mass

Table 6
Pertinent information for the step-by-step illustration

	Components		
Feed-stream 1	A	B	C
Feed-stream 2	A	B	C
Product-stream 1	A	B	–
Product-stream 2	–	B	C
Product-stream 3	–	–	C

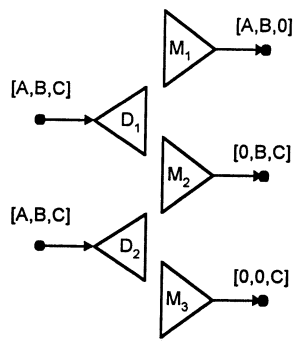

Fig. 16. Creating and linking a divider to each feed-stream and a mixer to each product-stream.

loads of the separators; this is equivalent to assuming that the degree of difficulty of every separation is 1 in Eq. (1). The pertinent data are summarised in Table 1.

The rigorous super-structure generated by algorithm SNS-LMSG is given in Fig. 5. The three-component feed-stream with components *A*, *B*, and *C* is split into four streams; two of them are fed into separators performing separation of two different types, and the remaining two streams are bypassed to the product-streams. Note that a variable is assigned to each of these streams since they emerge from a divider. Specifically, variables $x_{D_1M_1}$ and $x_{D_1M_2}$ are assigned to the streams that are bypassed to product-streams 1 and 2, whereas variables $x_{D_1S_1^1}$ and $x_{D_1S_1^2}$ are assigned to the inlet streams to separators S_1^1 and S_1^2, respectively. The top stream from separator S_1^1, containing component *A* only, is then split into two streams that are merged with the product-streams; these two streams are assigned

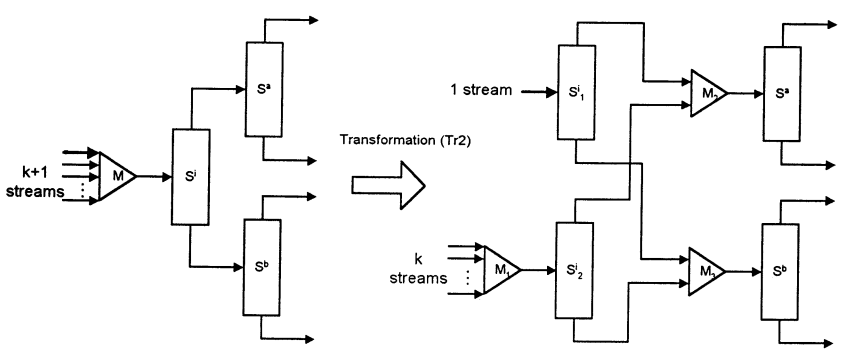

Fig. 15. Transformation (Tr2).

with variables $x_{D_2M_1}$ and $x_{D_2M_2}$. The bottom stream from separator S_1^1, containing components B and C, is split into three; two of them are merged with the product-streams and the remaining one is sent to separator S_2^2. Appropriate variables are assigned to these streams as indicated in Fig. 5. The top stream from separator S_1^2, containing components A, and B, is split into three streams; two of them are merged with the product-streams and the remaining one is fed to separator S_2^1. The bottom stream from the same separator, containing component C only, is split into two streams that are merged with the product-streams. A new variable is assigned consecutively to each stream resulting from splitting, as indicated.

According to expressions (1)–(6), the LP model for this separation-network synthesis problem can be expressed as follows:

$$\min (30 \cdot x_{D_1S_1^1} + 30 \cdot x_{D_1S_1^2} + 20 \cdot x_{D_3S_2^2} + 20 \cdot x_{D_4S_2^1})$$

from Eq. (1)

subject to:

$$0 \le x_{ij}\ (i, j) \in \{D_1M_1, D_1M_2, D_1S_1^1, D_1S_1^2, D_2M_1, D_2M_2,$$
$$D_3M_1, D_3S_2^2, D_3M_2, D_4M_1, D_4S_2^1, D_4M_2,$$
$$D_5M_1, D_5M_2, D_6M_1, D_6M_2, D_7M_1, D_7M_2,$$
$$D_8M_1, D_8M_2, D_9M_1, D_9M_2\} \qquad \text{from Inequality (2)}$$

$$x_{D_1M_1} + x_{D_1S_1^1} + x_{D_1S_1^2} + x_{D_1M_2} = 1 \qquad \text{from Eq. (3)}$$

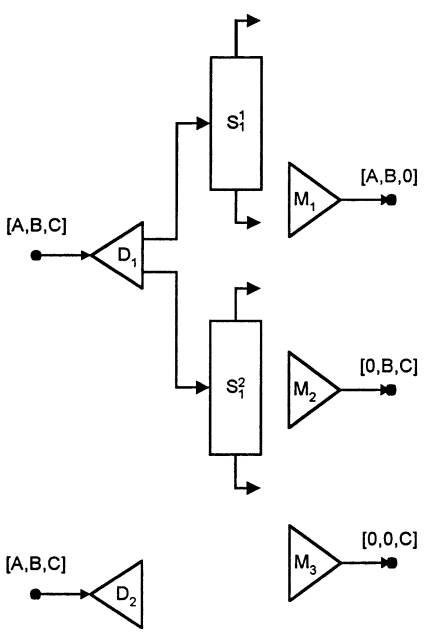

Fig. 17. Creating seperators S_1^1 and S_1^2 and linking both of them to the outlet of divider D_1.

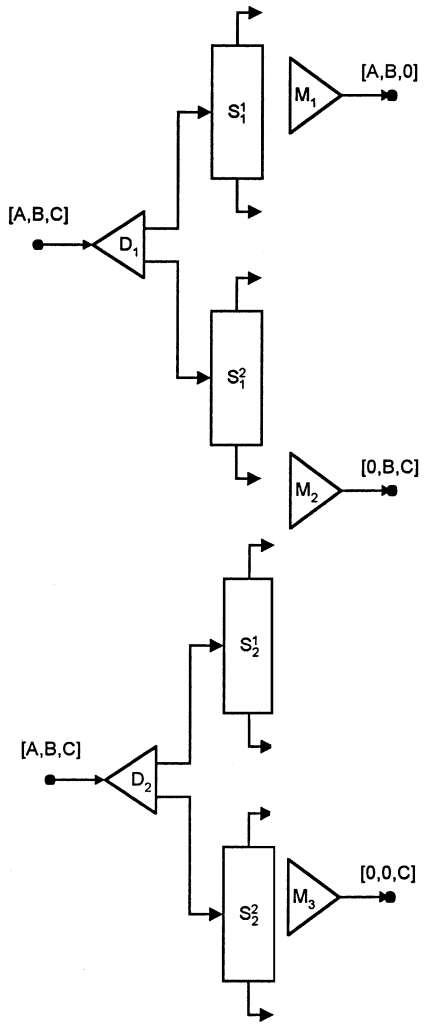

Fig. 18. Creating seperators S_2^1 and S_2^2 and linking both of them to the outlet of divider D_2.

$$\left.\begin{aligned}
x_{D_2M_1} + x_{D_2M_2} &= x_{D_1S_1^1} \\
x_{D_3M_1} + x_{D_3S_2^2} + x_{D_3M_2} &= x_{D_1S_1^1} \\
x_{D_4M_1} + x_{D_4S_2^1} + x_{D_4M_2} &= x_{D_1S_1^2} \\
x_{D_5M_1} + x_{D_5M_2} &= x_{D_1S_1^2} \\
x_{D_6M_1} + x_{D_6M_2} &= x_{D_3S_2^2} \\
x_{D_7M_1} + x_{D_7M_2} &= x_{D_3S_2^2} \\
x_{D_8M_1} + x_{D_8M_2} &= x_{D_4S_2^1} \\
x_{D_9M_1} + x_{D_9M_2} &= x_{D_4S_2^1}
\end{aligned}\right\} \qquad \text{from Eq. (4)}$$

$$\left.\begin{aligned}
6 &= 10 \cdot (x_{D_1M_1} + x_{D_2M_1} + x_{D_4M_1} + x_{D_8M_1}) \\
4 &= 10 \cdot (x_{D_1M_1} + x_{D_3M_1} + x_{D_4M_1} + x_{D_6M_1} + x_{D_9M_1}) \\
2 &= 10 \cdot (x_{D_1M_1} + x_{D_3M_1} + x_{D_5M_1} + x_{D_7M_1}) \\
4 &= 10 \cdot (x_{D_1M_2} + x_{D_2M_2} + x_{D_4M_2} + x_{D_8M_2}) \\
6 &= 10 \cdot (x_{D_1M_2} + x_{D_3M_2} + x_{D_4M_2} + x_{D_6M_2} + x_{D_9M_2}) \\
8 &= 10 \cdot (x_{D_1M_2} + x_{D_3M_2} + x_{D_5M_2} + x_{D_7M_2})
\end{aligned}\right\}$$

from Eq. (5)

The resultant mathematical programming model gives rise to a linear programming problem. Note that the model proposed by Quesada and Grossmann (1995) is nonlinear. The solution of the LP problem yields 12.00 as the minimal value of the objective function with

$$x_{D_1M_1}=0.2 \quad x_{D_1S_1^1}=0.2 \quad x_{D_1S_1^2}=0.2 \quad x_{D_1M_2}=0.4$$

$$x_{D_2M_1}=0.2 \quad x_{D_3M_2}=0.2 \quad x_{D_4M_1}=0.2 \quad x_{D_5M_2}=0.2$$

Fig. 6 is the corresponding optimal structure obtained, which is identical to that of Quesada and Grossmann (1995).

4.2. Example 2

Example 3 of Quesada and Grossmann (1995) is repeated here; three multicomponent product-streams

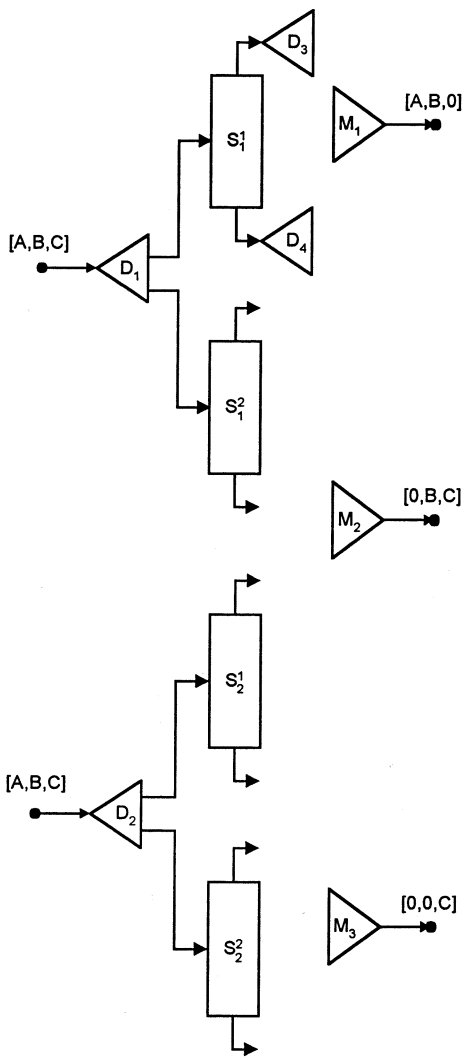

Fig. 19. Creating seperators D_3 and D_4 and linking each of them to each of the outlets of separator S_1^1.

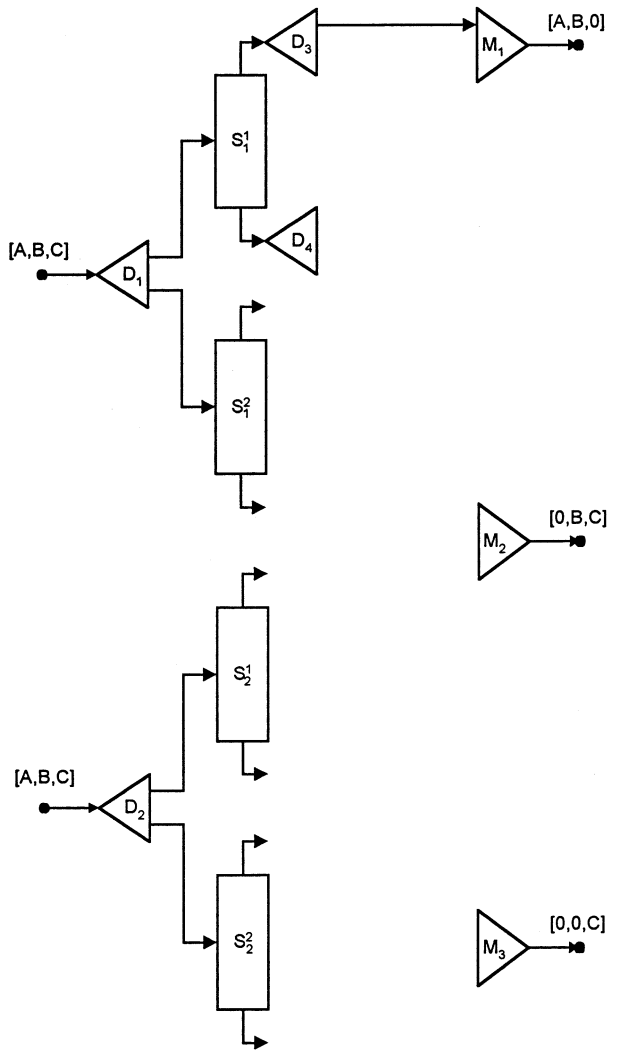

Fig. 20. Establishing a bypass from the outlet of divider D_3 to the inlet of mixer M_1 for product-stream [A, B, 0].

are obtained from three feed-streams. The objective function to be minimized is the sum of the costs of the individual separators in the network, where the cost of a separator is its mass load multiplied by the degree of difficulty of the separation. The pertinent data are given in Table 2.

Quesada and Grossmann (1995) considered the super-structure given in Fig. 7, and generated the optimal structure based on this super-structure given in Fig. 8; the value of the objective function obtained from an NLP model with 113 variables, with an execution time 0.74 s (given by the authors) on an IBM RS600/530, is 138.70.

The rigorous super-structure is generated with algorithm SNS-LMSG. The mathematical programming model derived from it is linear and contains 90 variables, of which 80 variables are assigned to the network itself, i.e. to the dividers. Ten variables are assigned to

the product-streams since the product-streams are specified by constraints instead of by their compositions in the problem specification. For example, component A must be greater than or equal to 9, while components B and C must be less than or equal to 3, and component D must be negligibly small in product-stream 1.

The optimal structure based on the proposed method, given in Fig. 9, has been obtained with an execution time of 0.55 s on a 100 MHz Pentium PC by means of the LP solver BDMLP of GAMS 2.25 (Brooke, Kenderick & Meeraus, 1996); the value of the corresponding objective function is 104.26.

Fig. 10 illustrates the optimal structure constructed by combining separators performing an identical separation according to the present method. Although this structure differs from that in Fig. 9, their total costs are identical because of the linearity of the cost functions. Note that although they are not specified numerically but defined to satisfy inequalities, the compositions of the product-streams in the optimal solution are unexpectedly identical to those given in Quesada and Grossmann (1995).

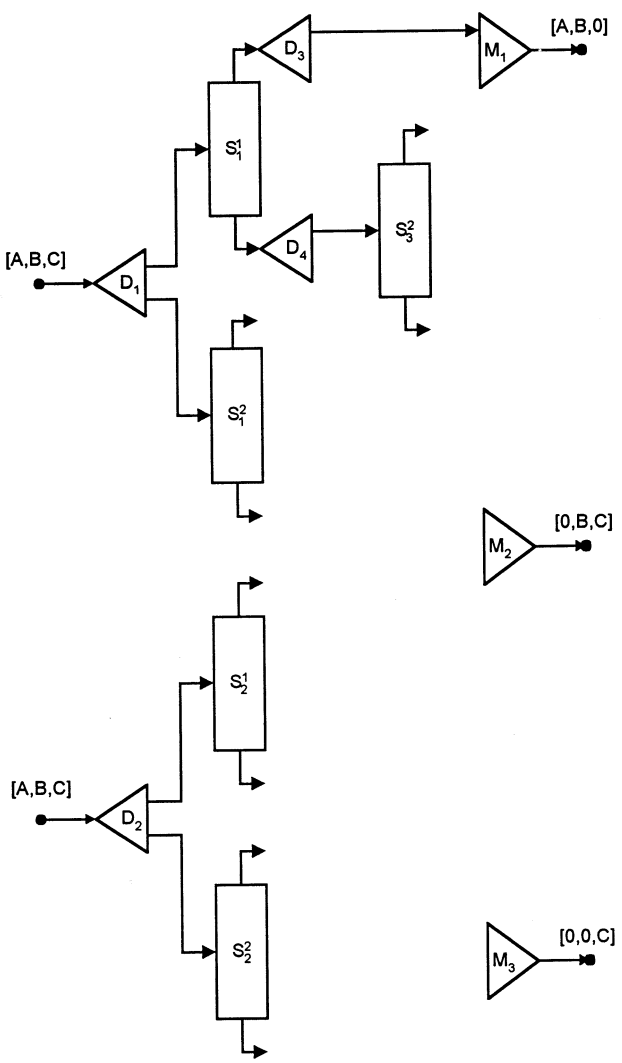

Fig. 21. Creating seperator S_3^2 and linking it to the outlet of divider D_4.

4.3. Example 3

This example is Example 6 of Quesada and Grossmann (1995) in which one four-component feed-stream is to be separated into two four-component product-streams; the pertinent data are listed in Table 3. Again, the proposed mathematical programming model is derived from the rigorous super-structure constructed by algorithm SNS-LMSG. The resultant optimal structure of four separators costing 54.25 depicted in Fig. 11 differs from that of Quesada and Grossmann (1995) costing 55.50 and including three separators.

4.4. Example 4

Repeating Example 12 of Quesada and Grossmann (1995) with our proposed method has yielded four six-component product-streams from a six-component feed-stream; the pertinent data are listed in Table 4. The resultant optimal structure including 11 separators is depicted in Fig. 12. The corresponding value of the objective function is 330.76, which is 15% less than the optimal value of 388.00 obtained by Quesada and Grossmann (1995); their work has yielded a system with five separators. Our execution time was 1.65 s on a 100 MHz Pentium PC in contrast to their 33 s on an IBM RS600/530.

Table 5 summarises the results obtained in the present work. These results are also compared with those of Quesada and Grossmann (1995). The solver and the examples are available at http://www.dcs.vein.hu/capo/demo.

5. Conclusions

Algorithmic methods have been extensively explored because of the importance of attaining the optimality of the solutions in separation-network synthesis (SNS). It appears, however, that hitherto no method has been available to algorithmically and rigorously solve every step of an SNS problem and at the same time is capable of ensuring the optimality of the solution. This is attributable to the difficulty in systematically generating appropriate mathematical programming models.

A novel algorithmic method is proposed in the present work for SNS problems with linear cost functions; the method is totally algorithmic throughout the solution. With this method, the optimal solution is obtainable with certainty for any instance of this class of SNS problems. It is demonstrated that the genera-

Table 5
Summary and comparison of the results

Example number	Number of components	Number of products	Number of variables		Time(s)		Optimal value of the cost function	
			Quesada and Grossmann, 1995	Present work	Quesada and Grossmann, 1995*	Present work**	Quesada and Grossmann, 1995	Present work
1	3	2	65	22	0.13	0.22	12.00	12.00
2	4	3	113	90	0.74	0.55	138.18	104.26
3	4	2	107	67	1.37	0.26	55.50	54.25
4	6	4	430	1094	33.00	1.65	388.00	330.76

* IBM RS600/530.
** 100 MHz Pentium PC.

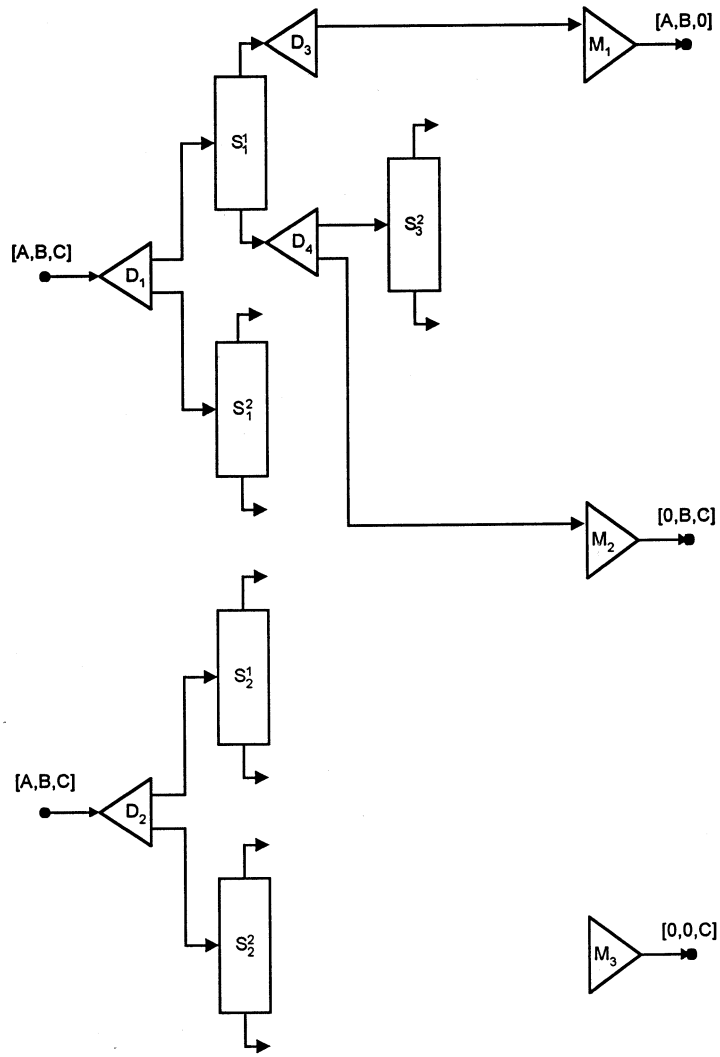

Fig. 22. Establishing a bypass from the outlet of divider D_4 to the inlet of mixer M_2 for product-stream [0, B, C].

tion of the rigorous super-structure is the crucial step of SNS. The fact that the mathematical programming model inherent in the proposed method is linear insures that the globally optimal solution can be generated; the same fact renders it possible to apply the method in conjunction with any available linear programming code to problems much larger than those unsolved so far. The efficacy of the present method is demonstrated by solving four examples taken from the literature. It should be cautioned, however, that a variety of practical considerations, such as the costs of mixers and dividers, piping complexity, pumping energy, maintainability, and controllability, may overshadow the desirability of a separation-network.

The present algorithmic method has been proposed for SNS problems with linear cost functions. Nevertheless, it should also be useful for determining initial or approximate feasible solutions when the cost functions involved are nonlinear.

Acknowledgements

This research was partially supported by the Hungarian Science Foundation Grant No. T-014212. This is contribution # 99-224-J, Department of Chemical Engineering, Kansas Agricultural Experiment Station, Kansas State University, Manhattan, KS 66506.

Appendix A. Proofs of Theorems

Proof of Theorem 1. The theorem will be proved by induction on the maximum of the numbers of components in the feed-streams. The simplest case involves two components in each feed-stream; consequently, every outlet stream of any separator contains a single component. Obviously, no separation can be performed on this outlet stream, thereby resulting in a loopless

network. Now, the theorem is presumed to be valid for any SNS problem for which at most k (>2) components are in each of its feed-streams.

Suppose that the optimal network is given for an arbitrary SNS problem for which the maximum of the numbers of components in its feed-streams is $(k+1)$. If none of the operating units is situated between two successive mixers or two successive dividers, then, these mixers or dividers are combined; naturally, it does not affect the optimality. The resultant optimal network is designated as N; and the SNS problem under consideration, as P. If network N is loopless, then Theorem 1 holds.

Let a path of the network be termed simple if it begins at one of the feed-streams and passes through operating units other than separators before reaching its end which is a separator; see the example in Fig. 13(a). On the other hand, a path is termed non-simple if it starts at one of the feed-streams and leads to a separator containing at least one separator between the beginning and the end of the path; see the example in Fig. 13(b). If a separator is linked to both a simple path and a non-simple path, it is deemed to possess property (π); such a separator is depicted in the top half of Fig. 14.

If network N contains a loop, the following is executed.

First, suppose that there exists a separator with property (π) in network N; then, transformation (Tr1) is executed on the network as illustrated in Fig. 14. Note that if a separator does not have property (π), transformation (Tr1) will not furnish it with property (π); in fact, transformation (Tr1) reduces the number of separators with property (π) by one. The cost of the resultant network is identical to that of network N because the individual separator's cost is regarded as linearly proportional to its mass load. Hence, the transformed network is also optimal. Transformation (Tr1) is repeated until no separator with property (π) remains in the network. This procedure terminates in a finite

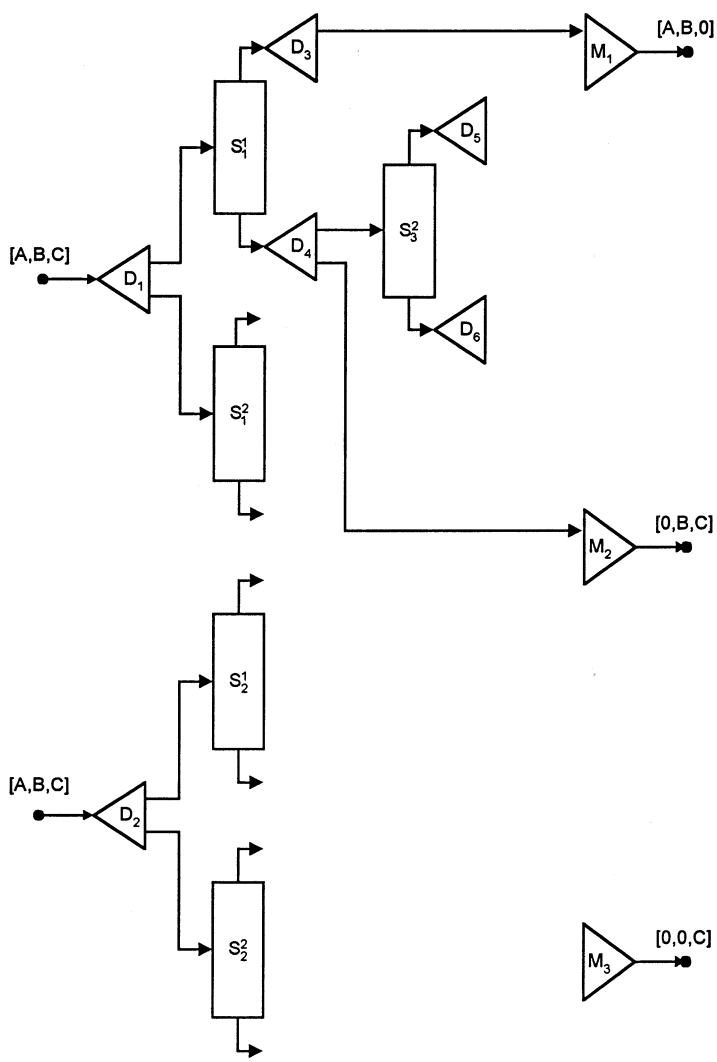

Fig. 23. Creating dividers D_5 and D_6 and linking each of them to each of the outlets of separator S_3^2.

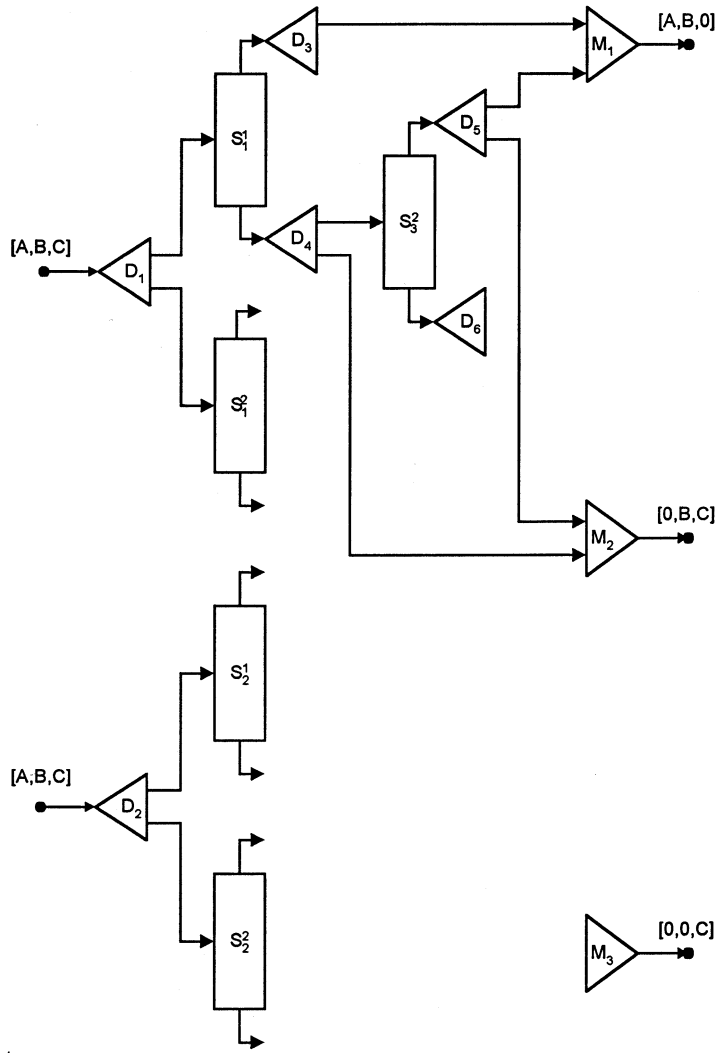

Fig. 24. Establishing two bypasses from the outlet of divider D_5, one to mixer M_1 for product-stream [A, B, 0] and the other to mixer M_2 for product-stream [0, B, C].

number of steps because the number of separators with property (π) is finite in network N; the resultant network is denoted as network N^*, which is obviously an optimal network of SNS problem P.

Second, if no separator with property (π) is in network N, this network is renamed as N^*. This is followed by the construction of SNS problem P_1. This problem has the same product-streams as SNS problem P, but its feed-streams are identical to the outlet streams of the separators that are endpoints of simple paths in N^*. Let N_1^* denote the subnetwork of N^* corresponding to problem P_1, and N_2^* denote the remaining subnetwork of N^*. Note that the maximum of the numbers of components in the feed-streams of problem P_1 is at most k.

According to the hypothesis, an SNS problem with at most k components in each of the feed-streams, e.g. SNS problem P_1, gives rise to a loopless optimal network; this loopless optimal network is denoted by N_1. Then, subnetwork N_1^* of network N^* is replaced with N_1 (i.e. connecting networks N_2^* and N_1), therefore resulting in another loopless network, the cost of which cannot exceed that of network N_1^* since network N_1 is an optimal network of SNS problem P_1. Thus, the modified N^* is also optimal, thereby verifying the theorem.

Corollary. *There exists a loopless optimal network where mixers are assigned only to product-streams for any separation-network synthesis problem with simple and sharp separators, dividers and mixers, provided that the cost of a network is the sum of the separators' costs, each of which is proportional to its mass load.*

Proof. There exists a loopless optimal network to any given SNS problem according to Theorem 1. Network transformation (Tr2) illustrated in Fig. 15 is repeated as long as a separator exists in the network on which this

transformation can be performed. The number of these transformations is finite since the network is loopless. The cost is not increased during the transformation, thus preserving the optimality. For simplicity, separators identical in inlet stream and type are combined during the transformation.

Proof of Theorem 2. The algorithm SNS-LMSG generates the union of networks described in the corollary for a given SNS problem, i.e. all potential operating units and their linkages appear in the structure generated, thus the theorem is obvious.

The finiteness is a critical issue for any algorithm. In this regard, the present algorithm terminates in a finite number of steps. This can be elucidated as follows:

The numbers of feed-streams and product-streams are finite; as a result, the first part of the algorithm is finite. Single-component streams are fed into the mixers from which the product-streams emerge; thereafter, no further separation is performed. The outlet streams from separators contain fewer components than their inlet streams; hence, single-component streams can be generated in a finite number of steps. Moreover, since the number of product-streams is finite, the number of streams between all the dividers accompanying the feed-streams and all the mixers concomitant with the product-streams is finite, thereby ascertaining the finiteness of the algorithm.

Appendix B. Step-by-step illustration of algorithm SNS-LMSG

The example to illustrate algorithm SNS-LMSG is specified in Table 6. Figs. 16–24 depict the early steps of algorithm SNS-LMSG in solving the example; the rigorous super-structure given in Fig. 25 is generated by repeating these steps. Every step of the example should be self-evident from the detailed caption provided to each figure.

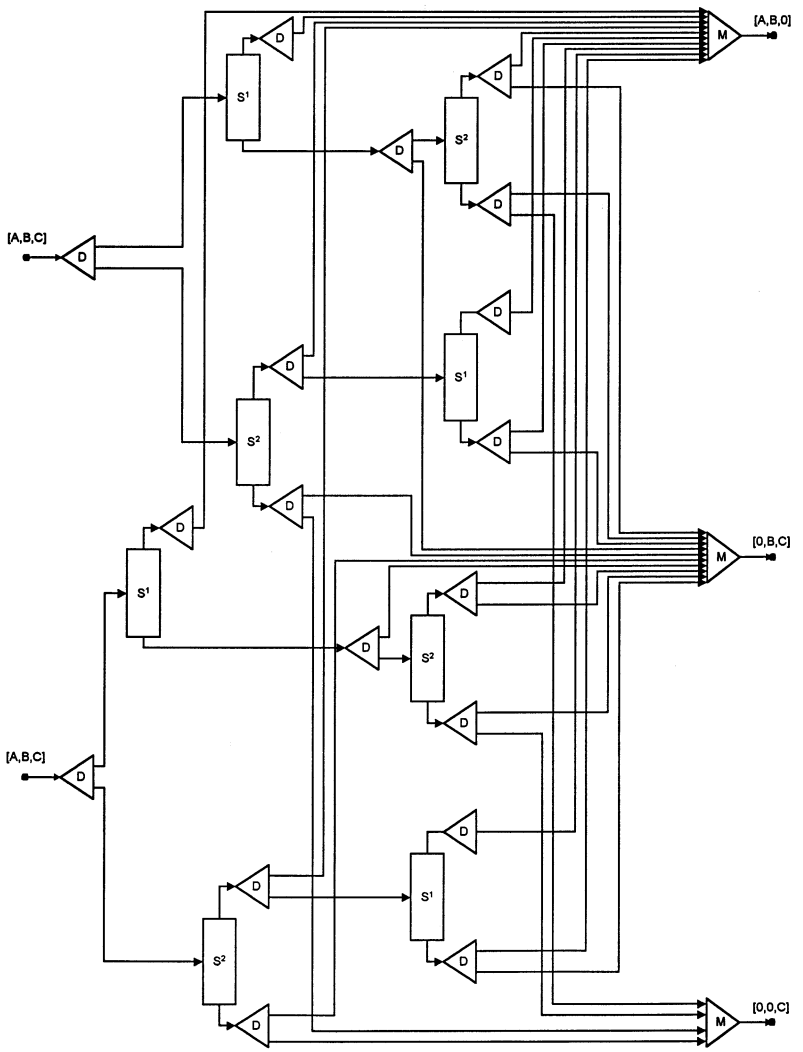

Fig. 25. Resultant rigorous super-structure.

References

Brooke, A., Kenderick, D., & Meeraus, A. (1996). *GAMS: a user's guide*, Release 2.25. GAMS Development Corporation.

Floudas, C. A. (1987). Separation synthesis of multicomponent feed streams into multicomponent product streams. *American Institute of Chemical Engineering Journal*, *33*, 540–550.

Floudas, C. A., & Aggarwal, A. (1990). A decomposition strategy for global optimum search in the pooling problem. *ORSA Journal of Computers*, *2*, 225–235.

Friedler, F., Tarján, K., Huang, Y. W., & Fan, L. T. (1993). Graph-theoretic approach to process synthesis: polynomial algorithm for maximal structure generation. *Computers & Chemical Engineering*, *17*(9), 929–942.

Kovács, Z., Ercsey, Z., Friedler, F., & Fan, L. T. (1998). Redundancy in a separation-network. *Hungarian Journal of Industrial Chemistry*, *26*(3), 213–219.

Kovács, Z., Friedler, F., & Fan, L. T. (1993). Recycling in a separation process structure. *American Institute of Chemical Engineering Journal*, *39*(6), 1087–1089.

Quesada, I., & Grossmann, I. E. (1995). Global optimization of bilinear process networks with multicomponent flows. *Computers & Chemical Engineering*, *19*(12), 1219–1242.

Thompson, R. W., & King, C. J. (1972). Systematic synthesis of separation schemes. *American Institute of Chemical Engineering Journal*, *18*(5), 941–948.

Wehe, R. R., & Westerberg, A. W. (1987). An algorithmic procedure for the synthesis of distillation sequences with bypass. *Computers & Chemical Engineering*, *11*, 619–627.