The Chemical Engineering Journal, Elsevier Sequoia S.A., Lausanne. Printed in The Netherlands

Decomposition of Nets*

R. W. BARKLEY† and R. L. MOTARD

Department of Chemical Engineering, University of Houston, Houston, Texas 77004 (USA)

(Received 12 December, 1970; in final form 24 June, 1971)

freepaper.me FREE paper

Abstract

An improved mode of representation for networks resulting from process simulation flowcharts is presented based on the signal flowgraph concept. Decomposition of maximal cyclical nets in the digraph by a minimal cut-vertex set becomes a simple three-step procedure of reduction to intervals, elimination of self-loops and processing of node pairs associated with two-way edges. Identification of cyclical nets is obtained as a by-product of decomposition. The procedure is applicable to equation-ordering problems if the output set is selected a priori.

1. INTRODUCTION

Many generalized systems programs have been developed during the past decades to compute the steady-state material and energy balances for broad classes of chemical process configurations[4,8,9,13,17]. Evans *et al.*[3] have recently reviewed the state of such programs and their characteristics. In general, these systems are organized in modular fashion, where the modules are the basic unit computations of a typical process such as distillation, absorption, reaction, heat exchange, condensation, reboiling, expansion, compression, pumping, decanting, flashing, etc. Supervising the progress of computation on the entire process network is the executive program which orders the execution sequence according to the details of the process flowsheet supplied as data. The distinction between simulation programs and design programs has been discussed by Forder and Hutchison[5] and the systems we refer to above fall into both classes. While design programs are not suitable for simulation, simulation programs may be used for design, though somewhat inefficiently, by case study analysis. Our concern is primarily with simulation programs.

The unit computation modules in all simulation programs, such as the CHESS system developed at the University of Houston[13], do not lend themselves easily to being decomposed into sets of equations. These modules frequently contain sub-iteration processes and special algorithms which have been highly refined for a specific application to a given unit operation. Our reason for stressing this point is to distinguish the problem of decomposition of networks from the equation-ordering algorithms of Steward[18] and Himmelblau[6,7].

The network problem deals with the interconnection of modules, where each transforms its input data into a set of output results. The computational sequence can readily be managed by the executive program for arbitrary process configurations. The basic problem in network analysis as addressed by Sargent and Westerberg[16], Lee and Rudd[11], Christensen and Rudd[2] and Forder and Hutchison[5] is the structuring of recycle calculations or the management of information feedback in an efficient manner so as to reduce computational time. The optimal sequence is obtained by cutting or tearing a selected set of recycle streams, *i.e.*, edges in the graph of the process flowsheet, to reduce a cyclic directed graph or digraph to an acyclic one. The cut edges become the iteration parameters in forcing the convergence of the iterative computation by either direct substitution or by an algorithm such as the bounded Wegstein procedure[10], which lends itself easily to the solution of a mathematically equivalent system of equations of the form

$$x_{i+1} = f(\mathbf{x}_i) \tag{1}$$

where i is the iteration count and $\mathbf{x}$ is the vector of recycle parameters contained in the edges cut by the decomposition process. The Wegstein procedure is a one-dimensional concept and should be best suited

* This work was supported by the Office of Naval Research, Contract N00014-68-A-0151.

† Present address: Texaco, Inc., Houston, Texas.

for systems such as eqn. (1) where the diagonal terms of the Jacobian matrix of eqn. (1) are dominant since the forcing process is applied to each recycle parameter independently. A number of multidimensional convergence-forcing algorithms are conceivable but have not yet proved practically efficient within the limited context of system simulation programs.

The equation-ordering procedures, on the other hand, are primarily dedicated to identifying minimal sets of simultaneous equations for a given set of output variables or maximum cyclical nets in the context of network analysis. The simultaneous equations are then solved in groups using techniques such as the Marquardt algorithm[12] or steepest-descent methods. The identification of cyclical nets is also important in the network problem, but the network analysis is then more concerned with the strategy of decomposing the separate nests by tearing. Both points of view have the same mathematical model, *i.e.*, eqn. (1), where the **x**-vector in equation ordering is the set of variables selected *a priori* as the output set of a system of simultaneous equations. Himmelblau[7] has presented a heuristic procedure for the decomposition of cyclical nets by ordering their occurrence matrices. Westerberg and Edie[19] have extended the equation-ordering procedure by including the choice of the optimum output set concurrently with the ordering process for additional computational efficiency.

2. CONCEPTUAL MODELS FOR NETWORK DECOMPOSITION

Two basic traditions have developed for the analysis of networks and the ordering of equations. These and the new approach that we are proposing can all be used interchangeably in either application but with varying degrees of facility. All have foundations in graph theory but some concentrate primarily on the Boolean properties of the connection matrices of the graphs[5,6,7,14,15,18], while others deal with the identification of key graph theoretic elements[2,11,16].

All have used directed graph formation except Westerberg and Edie[19], whose approach implies bidirectional graphs, since the output variables are selected in the analysis. The latter approach should offer considerable advantage in developing design programs rather than simulation programs where the structure of the computational process and the choice of module inputs are dictated by specifications of an arbitrary set of output variables. In relation to simulation, a design program is frequently analogous to running the computations of the simulation program in reverse. The simulation inputs become the design outputs.

In any case, the choice of conceptual framework has made the decomposition task more difficult than it should be and the literature in the field is filled with heuristic strategies that are involuted and difficult to program.

3. A NEW MODEL

The conceptual model that we propose[1] is the directed graph of the flowsheet as a signal flowgraph. The new graph is the dual of the traditional flowsheet graph used most recently by Christensen and Rudd[2]. The simplicity of the resulting decomposition analysis will become obvious. Graph theoretic elements identical to those used by the latter and Sargent and Westerberg[16] are also recognized in this new formulation of the problem, but additional elements of great analytic power also emerge, especially the notion of reducibility and the occurrence of self-loops.

The basic distinction between the two representations lies in recognizing the streams or edges in the flowsheet graph as signals or nodes (vertices) in the signal flowgraph. The nodes of the flowsheet graph, *i.e.*, the process transformations, are devoid of useful information and become the edges of the signal flowgraph. The relationship between the two is shown in Fig. 1 for a typical cyclical net.

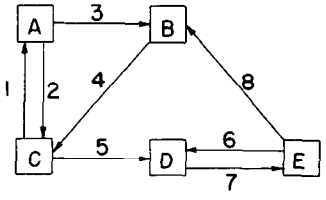

Fig. 1a. Flowsheet graph[2]. Nodes are the process modules.

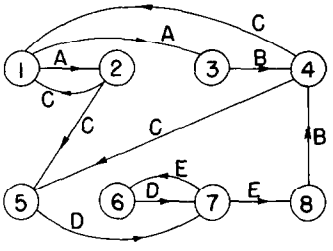

Fig. 1b. Signal flowgraph. Nodes are the streams of information.

freepaper.me FREE paper

The decomposition procedure on the signal flow-graph is effected by tearing or cutting nodes or vertices rather than by cutting edges as in the flow-sheet graph. Questions of optimality are deferred for later discussion. For the moment, no weighting of the nodes is assumed, despite the fact that each node includes all of the significant variables or parameters associated with each stream or flow of information, which vary in number from node to node. The first objective is to find the minimum cut-vertex set which

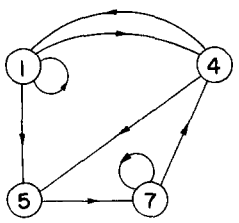

Fig. 2a. Graph of intervals.

Fig. 2b. After cutting self-loop node 7.

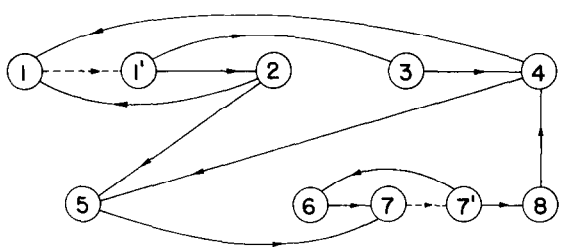

Fig. 2c. Decomposed graph after cutting nodes 1 and 7.

reduces the cyclic net to an acyclic net. A brief but incomplete summary of the method and the resulting effect on the graph of Fig. 1b leads successively to Figs. 2a, 2b, and 2c and the tabulation in Table 1:

(1) Identify all intervals in the graph and reduce the graph to interval header nodes only (Fig. 2a).
(2) Cut self-loop node 7 in Fig. 2a and repeat step (1) (Fig. 2b).
(3) Cut self-loop node 1 and complete the graph reduction summarized in Fig. 2c.

The primed nodes of Fig. 2c are the assumed inputs at the cut nodes and the dotted edges imply the introduction of a convergence-forcing computation at the end of each complete cycle. The ordering of the computation process, module-wise, by the program executive is relatively simple once streams 1 and 7 have been identified as assumed inputs and the transformations would be executed in the order A, E, B, C, D. Each cycle through the sequence would then be followed by a convergence analysis of output streams 1 and 7 and computation repeated if necessary.

TABLE 1

Summary of decomposed net in Fig. 1b

Intervals	*Minimum cut-vertex set*	*Output Intervals*
1,2,3	7,1	4,5
4		
5		
7,6,8		

4. PROPERTIES OF GRAPHS

The properties of the signal flowgraphs of recycle processes used in the analysis are applied in the following order, from the strongest to the weakest condition:

(1) Identification of maximal cyclical nets.
(2) Reduction of the graph to intervals.
(3) Elimination of self-loop nodes.
(4) Processing of nodes associated with two-way edges.
(5) Cut node with maximum number of output edges.

No examples have been found which cannot be decomposed by the routine application of these five steps. At the outset, we assert that no node of the graph will be linked to any other individual node by more than one input edge and one output edge. Any influences of node i on node j, as represented by parallel directed edges, are combined into one edge.

The identification of maximal cyclical nets can be done efficiently by Himmelblau's reachability matrix technique[6]. A considerable saving in computer storage requirements can be realized, however, if this is actually done after the signal flowgraph is reduced to intervals in step (2). The reachability matrix method works equally well on the flowchart graph formulation or signal flowgraph model. The algorithm used to reduce the graph eliminates all intervals that are headed by external input streams and merges all nodes

freepaper.me FREE paper

on paths that terminate on external outputs into their headers which are members of cyclical nets. Only linked cyclical nets remain after graph reduction. An alternative method of cyclical net identification within the present method is proposed below.

An interval in the graph is the maximal single entry subgraph for which h is the entry node and in which all closed paths contain h. No vertex of the graph can be included in the interval headed by h unless its precursors are already in the interval. By identifying all intervals in the graph it can be partitioned into subgraphs, and further processing occurs only on the reduced graph containing only the headers of the subgraphs, *i.e.*, the intervals. All edges linking any node in the interval with nodes outside the interval become properties of the header node in the reduced graph Referring to Figs. 1b, 2a and Table 1, an eight-node graph is reduced to four intervals; nodes 2 and 3 belong to the interval headed by node 1 and nodes 6 and 8 belong to the interval headed by 7. The graph of Fig. 2a is irreducible without cutting vertices. The algorithm for identifying intervals begins with two lists: a list of all nodes and their precursor nodes, *i.e.*, those nodes with edges directed toward a node. Any node with a single precursor is said to belong to that precursor and is eliminated; where such a node previously appeared as a precursor it is replaced by its header node in the precursor list. Table 2 summarizes the reduction to intervals of Fig. 1b. Repeated passes are made through the precursor list until all nodes with one precursor have been eliminated.

TABLE 2

Reduction to intervals

Node List	*Precursors*	*Intervals*	*Precursors*
1	2,4	1 (2,3)	1,4
2	1		
3	1		
4	3,8	4	1,7
5	2,4	5	1,4
6	7		
7	5,6	7 (6,8)	5,7
8	7		

Note that 6 may be included in the interval headed by 7 because the closed path 7-6-7 terminates on the header. The reduction algorithm takes care of such a path without special consideration.

The actual cutting process begins with the appearance of self-loops in the reduced graph. These must be members of the minimum cut-vertex set since there is no alternative which will reduce cyclicity. Whenever a node is to be cut it is eliminated from the graph and added to a list of cut streams. Where the cut stream appears in a precursor list it is also eliminated and this may render the graph reducible again, *i.e.*, nodes may now appear with single precursors. After any cutting process additional passes are made through the graph reduction algorithm to find any intervals which may be extended as the result of cutting. Table 3 indicates the effect of cutting self-loop node 7.

TABLE 3

Further reduction after cutting 7

Intervals	*Precursors*	*Intervals*	*Precursors*
1 (2,3)	1,4	1 (2,3) (4,5)	1
4	1		
5	1,4		

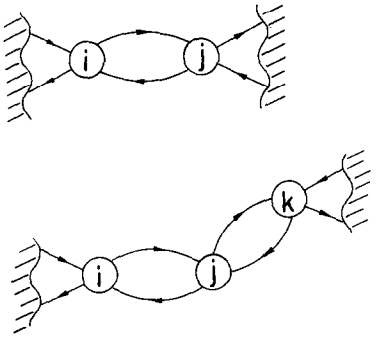

Fig. 3a and b. Two-way edge elements in a graph.

The processing of two-way edges is the next step in the algorithm. In most applications, self-loops and two-way edges are the only elements which can prevent the complete reduction of a cyclical net to one interval. The two-way edge is illustrated in Fig. 3 and was recognized by earlier investigators[2,16] as identifying a pair of streams of which one must necessarily belong to the cut set.

In Fig. 3a either node i or node j will belong to the cut-vertex set. Where the set of nodes in two-way edge elements includes a common node such as in Fig. 3b, the common node is immediately assigned to the cut-vertex set. If all of the two-way edge vertex pairs are disjoint, a selection is made of the node with the maximum number of output edges, since this offers

freepaper.me FREE paper

a greater possibility of affecting more nodes in the remaining graph. In the event of a tie an arbitrary choice is made. The final and weakest condition in the algorithm is invoked if all other conditions fail to appear. The node with the maximum number of output edges is selected for cutting for the same reason as stated above. In the event of a tie an arbitrary choice is made.

This completes the verbal description of the algorithm.

5. APPLICATION OF THE ALGORITHM

Examples of the application of the decomposition procedure are taken from Sargent and Westerberg[16] and Christensen and Rudd[2]. The first example is shown in Fig. 4. The steps in the algorithm will be illustrated in list format rather than graphically.

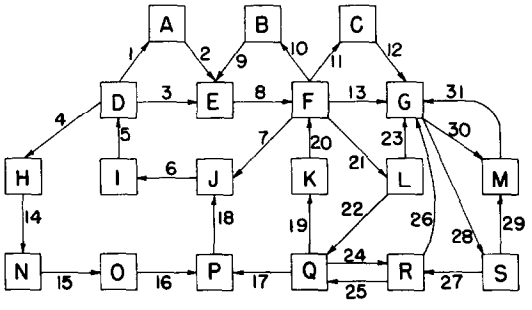

Fig. 4. Sargent and Westerberg (S & W) Example.

The nodes with only one precursor node have been eliminated (*) and any occurrences of these nodes in the complete list of precursors have been replaced by their precursors. The 31-node graph has been reduced to 16 intervals, as shown in Table 4. In detail, the beginning of the reduction process proceeds as follows:

(1) Node 1 belongs to node 5
(2) A 1 appearing as a precursor to 2 is replaced by 5
(3) Node 2 belongs to node 5
(4) Precursors of node 8 are now 5,3,9
(5) Node 3 belongs to node 5
(6) Precursors of node 8 are now 5,9
(7) Node 4 belongs to node 5
(8) Precursor of node 14 is not 5
(9) Node 5 belongs to node 6
(10) Precursors of node 8 are now 6,9
(11) Precursor of node 14 is now 6, etc.

The procedure for identification of pairs determines the occurrence of mutual precursors. For this example 7 pairs are found (6-18, 8-10, 19-21, 24-25, 26-28, 28-31, 30-31). Two common pair nodes are found, 28 and 31. Since one of the pairs includes both 28 and 31 the evidence indicates that the final cut set will include six nodes, one from each of the first four pairs and two from the last three. Node 28 is cut, since it has the largest number of output edges, and eliminated from the list of precursors. The graph is reduced further as nodes become single precursor nodes. In the process of reduction node 31 becomes part of the interval (30,31) and self-loops are generated at nodes 24 and 30 both of which are cut.

After nodes 28,24,30 are added to the cut set (Table 5) a new self-loop appears at node 21 and further reduction occurs as shown in Table 6. Cutting

TABLE 4

Initial reduction of S & W Example

	Reduction to intervals			*Two-edge pairs*
Nodes	*Precursors*	*Intervals*		*Precursors*
1	5			*
2	1			*
3	5			*
4	5			*
5	6			*
6	7,18	(6,1,2,3,4,5,14, 15,16)	(1)	7,18
7	8,20			8,19
8	2,3,9		(2)	6,10
9	10			*
10	8,20	(10,9)	(2)	8,19
11	8,20	(11,12)		8,19
12	11			*
13	8,20			8,19
14	4			*
15	14			*
16	15			*
17	22,25			21,25
18	16,17		(1)	6,17
19	22,25	(19,20)	(3)	21,25
20	19			*
21	8,20	(21,22,23)	(3)	8,19
22	21			*
23	21			*
24	22,25		(4)	21,25
25	24,27		(4)	24,28
26	24,27		(5)	24,28
27	28			*
28	12,13,23,26,31	(28,27,29) 6	(5)	11,13,21, 26,31
29	28			*
30	12,13,23,26,31		(7)	11,13,21, 26,31
31	29,30	6	(7)	28,30

freepaper.me FREE paper

21 finally reduces the graph to two self-loops at 6 and 8, both of which are cut. The final cut set is (6,8,21,24,28,30).

TABLE 5

S & W Example

Cut node 28		*Cut node 24 and 30*	
Intervals	*Precursors*	*Intervals*	*Precursors*
6	7,18	6	7,18
7	8,19	7	8,21
8	6,10	8	6,10
10	8,19	10	8,21
11	8,19	11	8,21
13	8,19	13	8,21
17	21,24		
18	6,17	18	6,21
19	21,24		
21	8,19	21	8,21
24	21,24	24	c
28	c	28	c
30	11,13,21,24,30	30	c

TABLE 6

S & W Example

Intervals	*Precursors*	*Intervals*	*Precursors*
6	8,6	6	c
8	6,8	8	c
21	c	21	c
24	c	24	c
28	c	28	c
30	c	30	c

In some cases the interval reduction algorithm is extremely efficient. Christensen and Rudd's[2] example number 1 is such a case, as shown in Fig. 5. Three self-loops at nodes 12,16,30 identify these nodes as the minimum cut set as shown in Table 7.

Another example presented by Christensen and Rudd is their example number 2. The process flowsheet is shown in Fig. 6 and the reduced signal flowgraph is shown in Table 8.

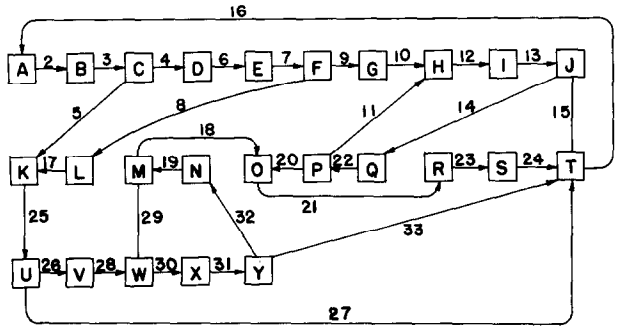

Fig. 5. Christensen and Rudd (C & R) Example No. 1.

TABLE 7

Initial reduction of C & R Example No. 1

Intervals	*Precursors*
(16,2,3,4,5,6,7,8,17,9,10,25,26,27,28)	12,16,21,30
(12,13,14,15,22,11,20)	12,16
(21,23,24)	12,30
(30,31,32,33,19,18,29)	16,30

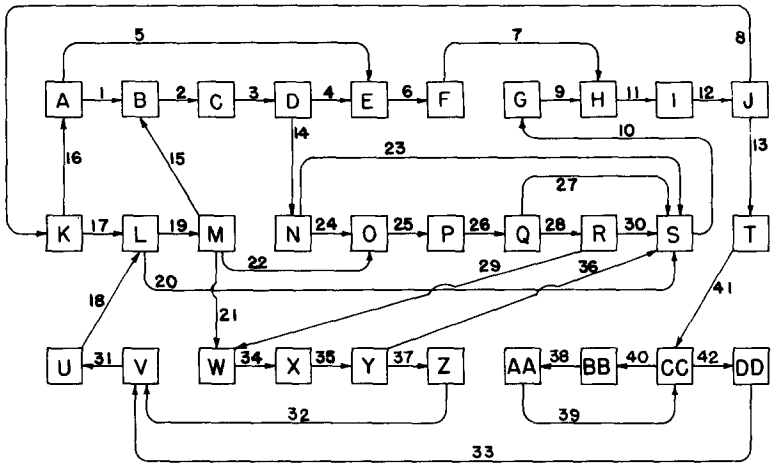

Fig. 6. C & R Example No. 2.

freepaper.me FREE paper

TABLE 8

Reduced graph of C & R Example No. 2
Final cut set = (40,11,34)

Intervals	*Precursors*	*After cutting nodes 40,11* *Intervals*	*Precursors*
(2,3,4,14,23,24)	11,19	*	
(6,7)	(1) 11,2	*	
(10,9)	6,20,25,34	*	
(11,12,13,8,41,16, 17,1,5)	(1) 6,10	11	c
(19,15,22,21)	11,31	*	
(20)	11,31	*	
(25,26,27,28,29,30)	2,19	*	
(31,18)	34,42	*	
(34,35,36,37,32)	19,25	34	34
(40,38,39)	40,11	40	c
(42,33)	40,11	*	

The reduced graph in Table 8 has one self-loop at node 40 which is cut. There remains one two-edge pair (6,11) and node 11 is cut because it has the larger number of output edges. Cutting 40 and 11 reduces the graph to a self-loop at 34 which is the final cut stream.

As a final example, the sulfuric acid plant problem of Shannon *et al.*[17] is analyzed in Fig. 7 and Table 9.

TABLE 9

Initial reduction of sulfuric acid plant model
Final cut set = (58,31,46,65,60)

Intervals	*Precursors*	*Pair nodes*
(4,5,6,8,25,26,27,28,29,30)	31,65	
9	4,58	
(10,11)	4,9	
12	10,58	
(13,14)	4,12	
15	13,60	
(16,17)	4,15	
18	16,60	
(19,20)	4,18	
(21,22,48,49)	19,58	
23	21,50	
(31,32,33,35,36,38,39)	34,40,46	(1) (2)
34	23,31	(1)
40	31,65	(2)
45	31,46	(3)
(46,42,43,47)	21,45	(3)
50	21,45	
53	19,58	(4)
54	4,58	(5)
55	10,58	(6)
(58,51,52,56,57,59,67,68,69)	53,54,55	(4) (5) (6)
(60,61,62)	53,54,55	
63	13,60	
64	16,60	
(65,2)	63,64	

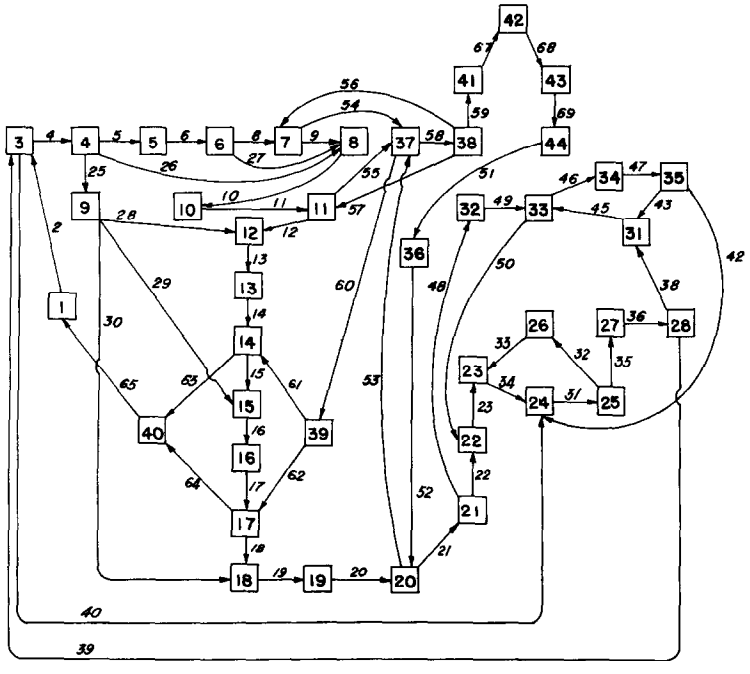

Fig. 7. Sulfuric acid plant model, Shannon *et al.*[17]

freepaper.me FREE paper

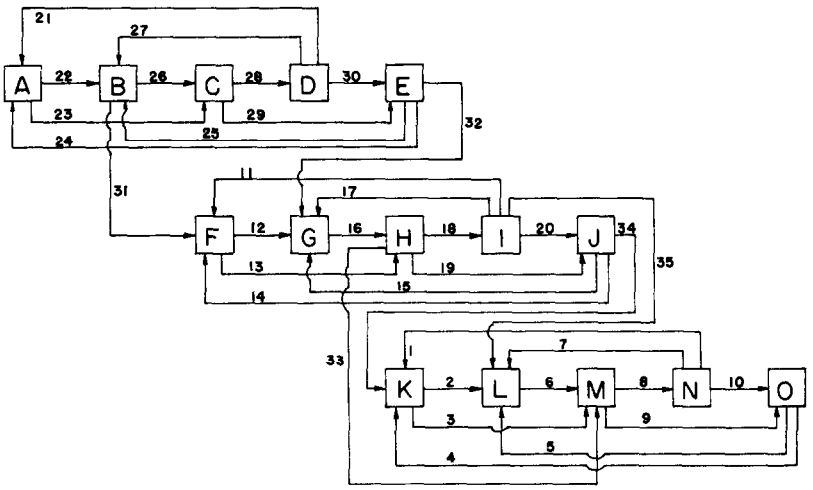

Fig. 8. Example with linked cyclical nets.

TABLE 10

Analysis of linked cyclical nets

Intervals	*Precursors*		*Cut-vertex precursors*	*Intervals*	*Cut-vertex precursors*
		after 28 is cut		Final results	
2		8,4,34	28		
3	(1)	8,4,34	28		
4		9,8	28		
5		9,8	28		
6	(2)	2,5,8,18	28		
(8,1,7,10)	(2)(1)	3,6,33	28	(8)	28,29,18,19,9,8
9		3,6,33	28	(9)	28,29,18,19,9,8
12		18,14,31	28		
13	(3)	18,14,31	28		
14		19,18	28		
15		19,18	28		
16	(4)	12,15,18,32	28		
(18,11,17,20,35)	(4)(3)	13,16	28	(18)	28,29,18,19
19		13,16	28	(19)	28,29,18,19
22		28,24	28		
23	(5)	28,24	28		
24		29,28	28		
25		29,28	28		
26	(6)	22,25,28	28		
(28,21,27,30)	(6)(5)	23,26	28	(28)	28,29
29		23,26	28	(29)	28,29
31		22,25,28	28		
32		29,28	28		
33		13,16	28		
34		19,18	28		

freepaper.me FREE paper

6. IDENTIFICATION AND ORDERING OF CYCLICAL NETS

While the reachability matrix technique of Himmelblau is an efficient method for identifying maximal cyclical nets, it is possible to accomplish the same objective with the new modelling method by adding a simple heuristic during the decomposition process. To illustrate, we choose an example from Christensen and Rudd (ref. 2, Fig. 15a) and replicate three times (Fig. 8) with additional feed-forward streams.

The basic cyclical net is represented by three groups of modules (A,B,C,D,E), (F,G,H,I,J), (K,L,M,N,O) with streams (1-10), (11-20), (21-30) respectively. Feed-forward streams linking the nets are 31-35. The reduced graph of the entire network is shown in Table 10. During the decomposition procedure a third list is maintained in addition to the interval list and the precursor list which we call the cut-vertex precursor list. If at any stage in the algorithm a node changes status by having its number of precursors reduced due to cutting a vertex, or by having a node added to its cut-vertex precursor list, the number of the latest cut node is added to the cut-vertex precursor list for this node and also wherever such a node appears in the precursor list. This process is carried on for cut nodes as well.

In Table 10 no self-loops are found; however, six two-edge node pairs are identified (3-8, 6-8, 13-18, 16-18, 23-28, 26-28) containing 3 common nodes (8,18,28). Since 28 has the maximum number of output edges it would be cut first, resulting in a reduction of precursors (crossed-out) for nodes 22,24,25,26,31 and 32. Node 28 is added to the cut-vertex precursor list for these nodes as well as for those nodes where these streams appear in the precursor list. This process is repeated for any nodes whose cut-vertex precursor list has been incremented as they appear as precursors to other nodes. The result of cutting node 28 is that all cut-vertex precursor lists are affected. Had node 8 been chosen as the initial cut only nodes 2,3,4,5,6,8 and 9 would have been so affected. Continuing the decomposition procedure in this manner leads to the final result shown in Table 10 and in matrix form in Fig. 9. The maximal cyclic nets appear as square sub-matrices along the main diagonal, and the order in which the cyclic nets should be processed is obvious: nodes 28 and 29 depend only on each other whereas nodes 18 and 19 depend on 28 and 29, etc. The cyclic nets are outlined in heavy lines and the matrix is constructed by ordering the columns in ascending numbers of cut-vertex precursors.

Another example of this phase of the algorithm is obtained by combining Fig. 1a with Fig. 5 with a feed-forward connection between module E (Fig. 1a) and module K (Fig. 5). Streams of Fig. 1a are renumbered 41-49, with 49 being the connection to Fig. 5.

cut-vertex precursor

	28	29	18	19	8	9
28	1	1	1	1	1	1
29	1	1	1	1	1	1
18			1	1	1	1
19			1	1	1	1
8					1	1
9					1	1

Fig. 9. Identification and order of cyclical nets.

cut-vertex precursor

	41	47	16	12	30
41	1	1	1	1	1
47	1	1	1	1	1
16			1	1	1
12			1	1	1
30			1	1	1

Fig. 10. Combined Fig. 1a and Fig. 5.

freepaper.me FREE paper

TABLE 11

Reduction of combined Figs. 1a and 5

Intervals	*Precursors*
41	41,44
44	41,47
45	41,44
47	45,47
16	12,16,21,30
25	16,47
12	12,16
21	12,30
30	25,30

Because of the additional stream into module K the initial graph reduction is affected. Final results are shown in Fig. 10 and Table 11.

7. OPTIMAL CUT SETS

Some of the workers[2] in the field have modified the decomposition process by weighting each stream with the number of parameters being transmitted between modules. The objective in the decomposition procedure is then revised to minimize the number of cut parameters rather than cut streams. This criterion of optimality is admittedly neither necessary nor sufficient since the improvement in computational efficiency in cutting one stream relative to another depends on the complexity of the unit computation modules as well as the number of parameters in their input streams. In fact, the advantage of one cut stream relative to another is a function of the global sensitivity of the cyclical net of modules to either stream and one would choose the stream with the larger sensitivity coefficient, which in itself is a composite effect of all the parameters in the stream. Truly optimal decomposition in the latter sense is a problem of much larger magnitude and is not attempted here. Instead, we offer the following observations on alternate cut sets and preferred streams.

8. ALTERNATE CUT SETS AND PREFERRED STREAMS

The cut set obtained by the proposed method is minimal in the number of elements but is not unique. Every header node of the sub-graphs generated during the initial reduction which ultimately appears in the cut set is potentially replaceable by members of its interval. In some cases the number of cut vertices will remain the same, in others, it will not. Identifying those members of the set of cut intervals which do not change the number of cut streams offers some flexibility in improving computational efficiency while not increasing the number of cut streams.

No property of graphs or sub-graphs has been discovered to identify alternate cut sets easily. The only recourse is to cut each member of an interval in turn prior to graph reduction and then to apply the decomposition algorithm. The same approach would be employed if the user wishes to specify preferred streams. Thus, in the problem shown in Fig. 6, cutting node 8 prior to reduction leads to the alternate cut set 8,40,31 instead of 40,11,34. Optimization should then be attempted over the set of alternate cut sets after these have all been identified.

9. ORDERING OF UNIT COMPUTATIONS

The ordering of unit computations for any cut set is a simple procedure. Given a cut set, the streams are cut one at a time (assumed known) and the unit computations enabled sequentially as their inputs are generated. The order of cutting will change the ordering of the computations but the results will be equivalent. Thus, in the problem shown in Fig. 6, if 40,11,34 are cut in that order the computation sequence will be:

BB,AA,I,J,T,K,A,CC,DD,X,Y,Z,V,U,L,M,B,C,D,E,
F,N,O,P,Q,R,S,W,G,H

If 31,8,40 are cut, the computational sequence will be:

U,K,A,L,M,B,C,D,E,F,N,O,P,Q,R,W,X,Y,Z,S,G,H,
I,J,T,BB,AA,CC,DD,V

These sequences form a cycle and at the end of each cycle convergence forcing or direct substitution may be applied to the cut streams as the user chooses. No sub-iteration cycles are employed since the entire cycle nest of computations is a function of the parameters of the cut streams as defined in eqn. (1).

REFERENCES

1 R. W. Barkley, Information structures and order concepts for chemical process computations, *Ph.D. Dissertation*, University of Houston, 1970.

freepaper.me FREE paper

2 J. H. Christensen and D. F. Rudd, *A.I.Ch.E.J., 15* (1969) 1.
3 L. B. Evans, D. G. Steward and C. R. Sprague, *Chem. Eng. Progr., 64* (4) (1968) 39.
4 FLOWTRAN, Monsanto Company, St. Louis, Missouri.
5 G. J. Forder and H. P. Hutchison, *Chem. Eng. Sci., 24* (1969) 771.
6 D. M. Himmelblau, *Chem. Eng. Sci., 21* (1966) 425.
7 D. M. Himmelblau, *Chem. Eng. Sci., 22* (1967) 883.
8 R. R. Hughes, E. Singer and M. Souders, *Proc. Sixth World Petroleum Congr., Frankfurt, 1963*, Vol. 6, Section VII, p. 93.
9 L. N. Kenny and J. W. Prados, A generalized digital computer program for performing process energy and material balances, *M.S. Thesis*, University of Tennessee, 1966.
10 Howard Carl Kliesch, An analysis of steady state process simulation, *Ph.D. Dissertation*, Tulane University, 1967.
11 W. Lee and D. F. Rudd, *A.I.Ch.E.J., 12* (1966) 6.
12 D. W. Marquardt, *J. Soc. Ind. Appl. Math., 11* (1963) 431.
13 R. L. Motard, H. M. Lee and R. W. Barkley, *CHESS (Chemical Engineering Simulation System) User's Guide*, University of Houston, Department of Chemical Engineering, Report RE 9-69, 1969, Defense Documentation Center AD-693522.
14 A. E. Ravicz and R. L. Norman, *Chem. Eng. Progr., 60* (May) (1964) 71.
15 D. I. Rubin, *Chem. Eng. Progr. Symp. Ser. No. 37, 58* (1962) 54.
16 R. W. H. Sargent and A. W. Westerberg, *Trans. Inst. Chem. Engrs., 42* (1964) 190.
17 P. T. Shannon, A. I. Johnson, C. M. Crowe, T. W. Hoffman, A. E. Hamielec and D. R. Woods, *Chem. Eng. Progr., 62* (6) (1966) 49.
18 D. V. Steward, *SIAM (Soc. Ind. Appl. Math.) J. Appl. Math.*, (Numer. Anal. Ser. B) *2* (1965) 345.
19 A. W. Westerberg and F. C. Edie, *Chem. Eng. J.*, 2 (1971) 9, 17.

freepaper.me FREE paper