$$I = \int_0^L \frac{G(y^+)}{1/\sigma + \epsilon(y^+)}\, dy^+$$

k = thermal conductivity
K_3, K_4, K_5 = coefficients in Taylor series expansion for ϵ beginning with y^{+3}
L = upper integration limit for I
M_4, M_5, M_6 = coefficients in Taylor series expansion for ϵ beginning with y^{+4}
n = exponent on y^+ in first term of ϵ expansion
$O(\ )$ = order symbol, $A = O(\sigma^a) \rightarrow A \propto \sigma^a$, for $\sigma \rightarrow \infty$
P_k = coefficients in Taylor series expansion for ϵ
q_w = wall heat flux
r^+ = dimensionless radial coordinate, $r^+ = ru^*/\nu$
R^+ = dimensionless tube radius, $R^+ = Ru^*/\nu$
$R_1{}^+$, $R_2{}^+$ = dimensionless radii for annulus, $R_1{}^+ = R_1u^*/\nu$, $R_2{}^+ = R_2u^*/\nu$
Re = Reynolds number, $Re = 2Ru_b/\nu$, for tube, $Re = 2Yu_b/\nu$, for parallel plates, $Re = \dfrac{2(R_2 - R_1)u_b}{\nu}$ for annulus
St = Stanton number, $St = h/\rho C_p u_b$
St_T = Stanton number for constant temperature boundary conditions
u^+ = dimensionless velocity, $u^+ = u/u^*$
u^* = friction velocity, $u^* = u_b\sqrt{f/2}$
u_b = bulk velocity
x^+ = dimensionless axial distance, $x^+ = xu^*/\nu$
y^+ = dimensionless distance from wall, $y^+ = yu^*/\nu$
Y^+ = dimensionless channel half-width, $Y^+ = Yu^*/\nu$
α = constant defined by Equation (10)
ϵ = dimensionless eddy diffusivity, $\epsilon = E_H/\nu$
ν = kinematic viscosity
σ = Prandtl or Schmidt number
θ = temperature
θ_0 = upstream temperature at $x = 0$
θ_1, θ_2 = wall temperatures
θ^+ = dimensionless temperature
θ_b = bulk temperature
$\theta_b{}^+$ = dimensionless bulk temperature
θ_c = centerline temperature

LITERATURE CITED

1. De Bruijn, N. G., "Asymptotic Methods in Analysis," North-Holland, Amsterdam (1961).
2. Deissler, R. G., NACA Tech. Note, 3145 (1954).
3. Elrod, H. G., *J. Aero. Sci.*, **24**, 468 (1957); erratum, **27**, 145 (1960).
4. Friend, W. L., and A. B. Metzner, *AIChE J.*, **4**, 393 (1958).
5. Hanna, O. T., paper presented at Am. Inst. Chem. Engrs. 63rd Annual Meeting, Chicago (1970).
6. Harriott, P., and R. M. Hamilton, *Chem. Eng. Sci.*, **20**, 1073 (1965).
7. Hubbard, D. W., and E. N. Lightfoot, *Ind. Eng. Chem. Fundamentals*, **5**, 370 (1966).
8. Knudsen, J. G., and D. L. Katz, "Fluid Dynamics and Heat Transfer," McGraw-Hill, New York (1958).
9. Lin, C. S., R. W. Moulton, and G. L. Putnam, *Ind. Eng. Chem.*, **45**, 636 (1953).
10. Lvon, R. N., *Chem. Eng. Progr.*, **47**, 75 (1951).
11. Seban, R. A., and T. T. Shimazaki, *Trans. A.S.M.E.*, **73**, 803 (1951).
12. Son, J. S., and T. J. Hanratty, *AIChE J.*, **13**, 689 (1967).
13. Wasan, D. T., C. L. Tien, and C. R. Wilke, *AIChE J.*, **9**, 567 (1963).
14. Wasan, D. T., and C. R. Wilke, *Intern. J. Heat Mass Transfer*, **7**, 87 (1964).

Manuscript received May 27, 1971; revision received October 21, 1971; paper accepted October 26, 1971.

An Efficient Algorithm for Optimum Decomposition of Recycle Systems

RAVINDRA S. UPADHYE and EDWARD A. GRENS II

Department of Chemical Engineering
University of California, Berkeley, California 94720

A method is developed for decomposition of a recycle process so as to minimize the summation of weighting factors for variables torn. This procedure is based on application of dynamic programming to a state space representing combinations of cycles opened. The resulting algorithm is well suited to machine implementation and is more efficient for large problems than alternative procedures that also guarantee decomposition.

Most chemical processes are characterized by recycle of material and/or energy. In addition, the units in a process are in general nonlinear. These characteristics usually combine to preclude direct solution to mathematical models for entire processes in the course of analysis or simulation of the process.

The presence of recycle streams requires the equation system representing several or all process units to be treated simultaneously. It is not possible to proceed directly with the calculation of units in any order, since the output of some units will be inputs to previous units in the sequence. Moreover, since the large number of equations that must be solved simultaneously are not linear, it is necessary to resort to iterative methods for the calculation of the variables involved in the process streams. These

Correspondence concerning this paper should be addressed to E. A. Grens II.

methods involve the decomposition of the process by assumption of values for certain of the stream variables in such a manner that sequential calculations become possible. The assumed (or "torn") variables must be corrected on the basis of the completed calculation of the sequence, and this procedure iterated until corrections become negligible.

The efficiency of such a process simulation can be considerably influenced by the choice of the decomposition scheme, that is, the selection of a set of variables to be torn. This set must lead to a valid decomposition, one that allows a direct calculation through resulting sequence (or sequences) of units. There are many choices that satisfy this requirement, and several criteria have been developed for the selection of the best set. Procedures have been developed that lead to tears in the minimum number of streams (*1*) and in the minimum number of variables (*1, 2, 3*). Actually, minimization of computational effort is desired. This can be approached through minimization of the summation over torn streams of some form of weightings (difficulty parameters) associated with the stream variables (*4*). These weightings are of course a function of the variables torn and must initially be estimated, with subsequent iterative revision. Thus, for a given process the decomposition procedure must be used a number of times to achieve an optimum decomposition. Procedures that lead to a minimum number of torn variables most often do not depend on the fact that the number of variables for any stream is a positive integer; these procedures can, therefore, also be regarded as accomplishing this minimization of weightings by consideration of the weighting factors as representing numbers of variables. Thus the work of Sargent and Westerberg (*3*), Lee and Rudd (*1*), Christensen and Rudd (*2*), and Christensen (*5*) can be regarded as methods for such minimization.

Sargent and Westerberg (*3*) developed a procedure for arranging the order of units in the process calculations so as to minimize the number of torn variables. After the process system is simplified to the extent possible by the merging of simply connected adjacent units and the elimination of recycle loops of length two, the procedure resorts to a systematic search of all permutations of units to find the one involving the smallest number of torn variables. Instead of generating and tearing all possible orders (permutations), dynamic programming is used to define recursive relationships for generation of the best orders for combinations of 1,2,3,. . .k units until all combinations of units are considered. This procedure suffers from the disadvantage that an excessive number of calculations may be required, since the search of orderings for any block of units is not guided by the structure of the block, as might be described by the cycle/stream matrix.

Christensen and Rudd (*2*) introduced the concepts of index nodes and ineligible edges in developing their decomposition procedure. According to this approach, certain streams are found to be ineligible to be torn and thus are eliminated from further consideration, and an index node is defined as one where all inputs, or outputs, or both are eligible to be recycle streams. As these authors mentioned, if more than one index node is required to produce an ordering, then optimality can only be guaranteed by examination of all permutations of index nodes. Christensen and Rudd also presented heuristic algorithms which however do not guarantee ordering in all cases (*2*). Such algorithms may be very efficient in many circumstances but are not easily compared with exact algorithms, which guarantee optimum decomposition. Christensen (*5*) later developed the concept of indexing and presented an algorithm, partly based on the earlier work, that can be used to minimize the number of recycle parameters. His basic concepts remained essentially unchanged, except that bipartite graphs were emphasized instead of directed graphs. The necessity of examination of all permutations of index nodes to assure optimality, therefore, still remained and could lead in certain cases to the calculations becoming quite cumbersome.

Lee and Rudd (*1*) had earlier developed an algorithm for minimization of torn variables based on the concept of streams (or sets of streams) that are contained in other streams. This approach takes into account the structure of the system (as given by the cycle/stream matrix). For systems with many streams, a large number of combinations of streams must be examined for containment, resulting in a severe combinatorial problem. This procedure is thus unsuited for large systems, especially if machine implementation is desired.

In many instances process simulations are required for optimization and comparison studies, where they will be utilized repeatedly. In these situations only moderate numbers of significant units and streams need be considered, but very efficient simulations are necessary. The decomposition employed must approach optimum in the real sense of computational effort. The present work has been directed to development of an efficient exact method for determination of the valid decomposition that minimizes the sum, over the torn streams, of weightings of the streams. It is assumed that all the variables in a stream are torn when the stream is torn. An algorithm based on the application of dynamic programming to the cycle/stream matrix has been formulated. This algorithm takes the form of a recursive operation, starting from the original cyclic process and ending in the desired acyclic process by systematic tearing of streams. It appears very favorable when compared for efficiency to the exact algorithms of Sargent and Westerberg (*3*) or Lee and Rudd (*1*).

BACKGROUND

The formulation of an efficient procedure for the decomposition of recycle systems requires consideration of their fundamental properties. A basis for understanding of such systems is provided by the theory of graphs (*7*).

A recycle process can always be represented by a directed graph containing circuitous paths or cycles. There are many methods available for the location of these cycles in a graph (*8* to *10*). The streams that constitute a cycle are said to be included in that cycle; tearing any one of these streams opens the cycle. A necessary and sufficient condition for a decomposition scheme to be valid is that the set of torn streams be such that for any cycle there is at least one stream in the set that opens the cycle. A cycle contains two or more streams, and a stream may open more than one cycle. It is, therefore, obvious that there usually exists more than one set of torn streams that constitute a valid decomposition.

These concepts are illustrated in the simple process shown in Figure 1. The nature of connections between units of this process can be represented by a cycle/stream matrix as shown in Table 1. Here rows represent cycles and columns represent streams. An element in the matrix is unity if the corresponding stream is included in the cycle concerned and is otherwise blank. This matrix need be determined for a given process only once, even though repeated decompositions of the process are required. The last row represents the summation of weightings of the variables for that stream. For this particular example the valid decompositions are given by the sets of torn streams: {1, 3, 4}, {1, 4, 7}, {3, 4, 5}, and {4, 5, 6, 7}.

Once the cycle/stream matrix is obtained, the problem

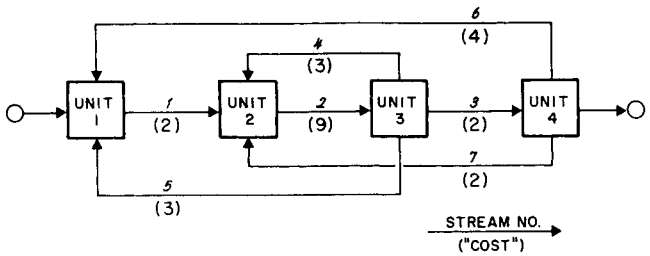

Fig. 1. Process flow diagram for simple process example.

TABLE 1. CYCLE/STREAM MATRIX FOR PROCESS SHOWN IN FIGURE 1

	Stream						
Cycle	1	2	3	4	5	6	7
1		1		1			
2	1	1			1		
3	1	1	1			1	
4		1	1				1
"Cost", $B(l)$	2	9	2	3	3	4	2

of selection of a valid decomposition reduces to that of choice of a set of columns (corresponding to streams), the union of which results in a column with all unit elements. The best decomposition is then the valid decomposition with the minimum total of stream weightings, summed over the torn streams.

The concept of the cycle/stream matrix had been used by Lee and Rudd in their algorithm to find the best decomposition based on minimum torn variables (*1*). Their approach to the problem is based on finding streams or sets of streams contained in other streams or sets (in so far as the decomposition problem is concerned). If a stream j is included in all cycles in which stream k is included and if the number of variables in stream j is greater than or equal to the number of variables in stream k, then stream j is said to be contained in stream k. Since stream k can open all cycles that are opened by stream j with the same or fewer torn variables, Lee and Rudd eliminated stream j from further consideration for inclusion in the optimum tearing. The same argument was applied to containment of sets of streams in other sets of streams. During this process of elimination, if at any time any cycle includes only one stream, that stream is selected to be torn. All the cycles opened by tearing this stream are also removed and the process continued till all cycles are opened.

As mentioned earlier there are several disadvantages to this procedure associated with the combinatorial problems of examination of sets of streams for containment in other sets. This procedure is impractical for machine implementation with systems of any appreciable size, although it may well be efficient for hand computation of systems small enough for the cycle/stream matrix to be easily scanned visually.

A DYNAMIC PROGRAMMING ALGORITHM

A more efficient algorithm can be formulated for selection of the best decomposition by consideration of the properties of the cycle/stream matrix for a process. The problem can be viewed as having two definite terminal states, initial and final. The initial state corresponds to the original system, with no cycles open. The final state corresponds to the acyclic system, with all cycles open. In between these two terminal states there exist many intermediate states, corresponding to various combinations of cycles opened. The solution to the problem may be considered to progress through these states. If these states are represented as nodes on a graph, then the streams selected to be torn may be represented by arcs joining the nodes. With each arc is associated a cost—the sum of weightings of variables included in the corresponding stream. In general, there exist many paths reaching the final state from the initial state. The optimum path is that path with the least sum of costs for its arcs, that is, the least total cost. For any intermediate state reached by a path, the path may be subdivided into two subpaths. One subpath connects the initial state and the intermediate state and the other subpath connects the intermediate state and the final state. Now each of the subpaths of the optimum path must itself be an optimum path between the states it connects. If this were not so, one of the subpaths could be replaced by an alternative subpath with a smaller cost and this would give rise to a path between the two terminal states with a smaller total cost than the original path, in violation of the assumption that the original path was optimum.

These conditions are precisely those to which dynamic programming applies. The problem, when viewed in this manner, is thus naturally suited to use of the dynamic programming method. However, this utilization of dynamic programming is quite different from its earlier application to the problem by Sargent and Westerberg (*3*). The major difference is in the state-spaces of these two algorithms; the state-space of Sargent and Westerberg consists of permutations of nodes, whereas here it consists of combinations of cycles. In the former work, for every combination of nodes there exists an optimum permutation, whereas here for every combination of cycles opened there exists an optimum combination of torn streams. In addition, the present work, unlike that of Sargent and Westerberg, is related to the internal structure of the system through the cycle/stream matrix of the process.

Algorithm Definition

For the development of a dynamic programming algorithm it is advantageous to introduce formal notation for the sets of streams involved, and in these terms to consider the properties of the set constituting the optimum decomposition in comparison with those of sets for nonoptimum decompositions.

Let $\{I\}$ be the set of all S streams in the process. Let $\{N\}$ and $\{K\}$, of size α_N and α_K respectively, be sets of torn streams constituting valid decompositions, with $\{N\}$ an optimum set and $\{K\}$ any nonoptimum set. These two sets may or may not have common elements. Let $\{n\}$ and $\{k\}$ be subsets of $\{N\}$ and $\{K\}$ of size α_n and α_k respectively and let $\{J\}$ be any subset of $\{I\}$. The function $D(\{J\})$ is defined as the sum of the costs of the streams in the set $\{J\}$. By the definition of optimum set, it is clear that $D(\{N\}) \leq D(\{K\})$. Further, let $\{C\}$ be the set of all cycles present in the system and $\{c\}_J$ a subset of $\{C\}$ comprising the cycles opened by the set $\{J\}$ of torn streams.

In this system a state is defined by the specification of the cycles opened. It can be shown that if the number of cycles in a system is a, the total number of states possible is 2^a. A one-to-one correspondence can be established between these states and the positive integers 0, 1, 2, ..., $(2^a - 1)$, with the first and the last integer corresponding to the initial and final states, respectively.

Since a set of torn streams $\{J\}$ uniquely determines the state of the system, the function $D(\{J\})$ could also be associated with a given combination of cycles or state. However, as there are, in general, more than one set of streams

leading to the same state, such a function would not be unique. An additional restriction must be included to define a unique cost function $E(\{c\}_J)$ for any state of the system.

$$E(\{c\}_J) \equiv \min D(\{J\}_i) \tag{1}$$

for all $\{J\}_i$ such that $\{J\}_i$ opens just the set of cycles $\{c\}_J$.

If the minimization defined in Equation (1) is achieved by the set $\{J\}_\beta$, then $\{J\}_\beta$ is called the optimum set of streams for the state defined by the set of cycles $\{c\}_J$. Since a state can also be defined by a positive integer, the argument of the function E can also be an integer p where p indexes the state given by $\{c\}_J$. Thus

$$E(p) \equiv E(\{c\}_J) = D(\{J\}_\beta) \tag{2}$$

Also, if $\{M(p)\}$ is defined to be the optimum set of torn streams to reach the state p

$$E(p) = D(\{M(p)\}) \,. \tag{3}$$

With this nomenclature, a recursion relationship can be developed to lead to the optimum decomposition from the initial state with the addition of one stream to some previous set of torn streams at each step.

$$E(p) = \min_{\substack{\text{all streams } l \\ \text{all states } q \\ \text{such that } q<p}} [E(q) + B(l)] \tag{4}$$

$$\{M(p)\} = \{M(q)\} \cup \{L\}. \tag{5}$$

Here $B(l)$ is the cost associated with the stream l, and $\{L\}$ is a one-stream set consisting of the stream l that minimizes the right-hand side of Equation (4). The initial conditions for the recursion are given by

$$E(0) = 0 \tag{6}$$

$$\{M(0)\} = \{\phi\} \,. \tag{7}$$

The optimum set of torn streams is given by $\{M(u)\}$ and the corresponding total cost is given by $E(u)$ where $u \equiv 2^a - 1$.

Algorithm Implementation

The Equations (4) through (7) constitute a formal description of the present dynamic programming approach to the decomposition problem. A few modifications, however, are necessary in order to provide for efficient implementation of such a procedure.

Instead of looking back from each stage p as done in Equation (4), it is possible and more efficient to look ahead. If the number of streams is S, then from each state q it is possible to reach not more than S states, one for each of the streams not yet torn. Suppose the stream l combined with the set $\{M(q)\}$ yields a new state p. This state may or may not have been reached earlier from some state other than q. If it has not been previously encountered, then the current optimum set of streams and the cost for the state p are given respectively by

$$\{M(p)\} = \{M(q)\} \cup \{L\}$$

and

$$E(p) = E(q) + B(l)$$

If the state p has been encountered previously, then a value for $E(p)$ already exists. This value is checked against the new value $E(q) + B(l)$. If $E(p) \leq (E(q) + B(l))$, then the set $\{M(q)\} \cup \{L\}$ is no better than the previous set $\{M(p)\}$ and therefore no change need be made in $E(p)$ and $\{M(p)\}$. However, if $E(p) > (E(q) + B(l))$, then $E(p)$ is replaced by $(E(q) + B(l))$ and $\{M(p)\}$ is replaced by $\{M(q)\} \cup \{L\}$. The same procedure is repeated for the state q and the next stream, $(l + 1)$. When all available streams have been considered, the procedure is repeated for the next state $(q + 1)$. This method of looking ahead is computationally superior to the looking back method by a considerable margin.

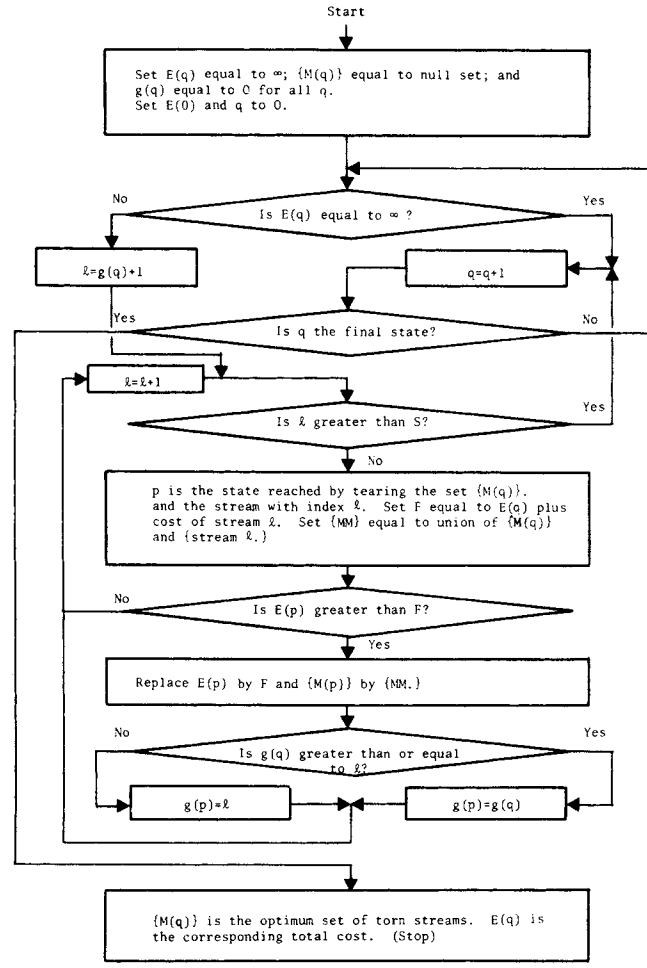

Fig. 2. Logic diagram for dynamic programming algorithm.

A possible modification to the procedure, although simple, further reduces the amount of computation required by nearly half. Consider a set of torn streams $\{n\}$ to reach the state q, such that $\{n\} = \{M(q)\}$. Let $\sigma(\{n\})$ denote the state reached by tearing the set $\{n\}$, so that $q = \sigma(\{n\})$. Now this state q can be reached by a single arc from at least α_n previous states, namely $\sigma(\{n\} - \{L\})$, where $\{L\}$ is a one-stream set consisting of the lth stream in the set $\{n\}$, for $l = 1, 2, \ldots, \alpha_n$. For all these α_n different paths, $E(q)$ and $\{M(q)\}$ are exactly the same. The duplication of effort associated with examination of all these paths can be avoided by definition of a function g over the states, such that $g(q)$ is the largest stream index number of the set $\{M(q)\}$. Then, at any state q only those streams whose index numbers are greater than $g(q)$ need be considered. If $g(q) = S$ for any state q, that state need not be considered for the purpose of looking ahead.

These considerations lead to the specification of an algorithm for selection of the optimum decomposition.* The logic flow diagram of the algorithm is shown in Figure 2. Briefly the procedure is as follows.

* Since any valid set of torn streams must include all the input streams to at least one of the units, a further modification would be to start with tearing the sets of input streams to each of the units and then to proceed from the earliest state thus reached. However, with this procedure, the g function cannot be employed and optimality still guaranteed; thus in most cases it is not advantageous.

First $E(q)$ is set to a large positive number, $g(q)$ is set to 0, and $\{M(q)\}$ is set to $\{\phi\}$ for all states q, and $E(0)$ is set to 0. Then starting from the initial state, the new states reached by tearing each stream, and their E, M and g values are found in turn. The next state, in terms of its index, is then considered. If it has not been encountered previously, it is skipped and the following state is considered until a state reached previously is found. Now, starting from this state, available streams are torn to reach all possible states, and their E, M and g values are set. This procedure is repeated until the final state is attained. The values of M and E for the final state will give the optimum set of torn streams and the total cost for this set respectively.

Example of Dynamic Programming Algorithm Application

The operation of this algorithm can be easily seen in its application to the simple example of Figure 1. Here the possible number of states is 2^4 or 16. These 16 states and their integer representations are shown in Table 2. The procedure can be visualized through Figures 3a and 3b, which represent the states by nodes and the streams torn by arcs connecting the nodes. In the diagrams states are numbered according to Table 2. To the right of each node there appear two sets of numbers. The upper number is the current minimum cost for the state. The lower numbers are the current best set of torn streams to reach the state from the initial state. The arcs are labeled by the numbers of the corresponding torn streams. Figure 3a shows all the states reached in one step from the initial state. Note that the optimum solution *at this stage* is given by stream {2}, with a cost of 9. Figure 3b illustrates the continued application of the algorithm. In both figures, the optimum paths to all nodes (states) are shown by solid lines and nonoptimum paths are shown by dotted lines; these latter in practice need not be recorded. In case there exists more than one optimum path to a node, all paths found after the first such path are considered nonoptimum. The optimum set of tears is thus found to be streams {1, 4, 7} with a cost of 7.

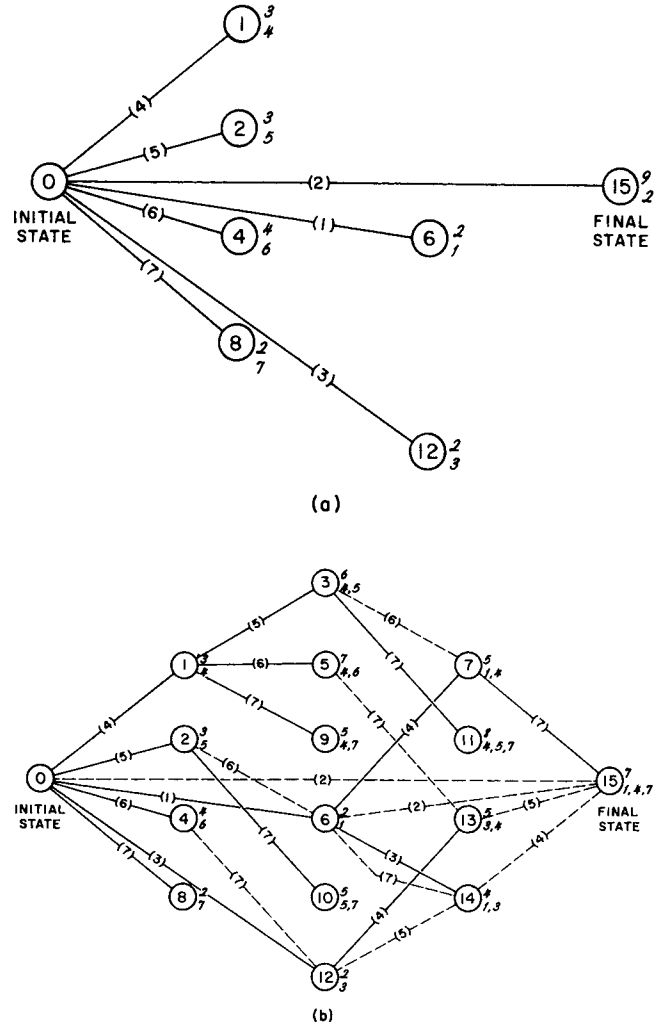

Fig. 3. State diagram illustrating application of dynamic programming algorithm. (a) First step from initial state; (b) Complete decomposition.

COMPARISON OF DECOMPOSITION ALGORITHMS

The present algorithm can be compared with other exact algorithms such as those of Sargent and Westerberg (*3*) and Lee and Rudd (*1*), which have certain fundamental similarities in their characterization of the systems. Examination of the dynamic programming algorithm of Sargent and Westerberg reveals that even when the number of cycles is somewhat larger than the number of units, their procedure requires more total operations than the one presented here. This results from the fact that within any given group of nodes there exist many permutations, each of which must be examined to find optimum ordering.

Table 2. Possible States for the Decomposition of the Process in Figure 1 Using the Dynamic Programming Algorithm

State index	Cycles open	State index	Cycles open
0	None	8	4
1	1	9	1,4
2	2	10	2,4
3	1,2	11	1,2,4
4	3	12	3,4
5	1,3	13	1,3,4
6	2,3	14	2,3,4
7	1,2,3	15	All

Even though dynamic programming helps reduce the number of subgroups to be considered, the number remaining is still very great for large problems. However, if the number of nodes is much smaller than the number of cycles, as in the case of a complete graph, the Sargent-Westerberg algorithm would require fewer total logical and arithmetic operations. The concept of the cycle/stream matrix is not used by Sargent and Westerberg. If this matrix is not otherwise required, the operations involved in its determination must be taken into account in any comparison. However, the matrix need be determined only once for a process while repeated applications of the decomposition algorithm are required if weighting factors are to be evaluated. Thus a truly comprehensive comparison is not possible.

Comparison of the present algorithm with that of Lee and Rudd (*1*) indicates that the present procedure requires far fewer operations for its implementation. A major reason for this is that repetitious considerations of sets of streams are avoided. With the Lee-Rudd algorithm, the only a priori limit to the size of sets considered is the number of cycles, and it is necessary to impose an arbitrary limit on the size of these sets. If this limit is placed too low, the algorithm may fail to give correct results. If it is too high, the calculations would be impractical. In the present algorithm, however, no such arbitrary limit is necessary; there exists an upper bound on the number of

operations necessary for its implementation. In practice, the number of operations required may be significantly smaller than this maximum.

In general, for a system with v nodes, S streams, and a cycles, the upper bounds on the number of basic operations needed by the present dynamic programming algorithm are

$$\text{number of logical operations} = (S+2)(2^a-1)$$
$$\text{number of arithmetic operations} = (S/2)(2^a-1).$$

For the same system, the Lee-Rudd algorithm requires a total number of logical and arithmetic operations roughly equal to

$$3\binom{S}{2} + S'\left[2\sum_{i=2}^{a} i\binom{S'-1}{i}\right] + 2\sum_{i=2}^{a}\sum_{j=1}^{S''-1}\left[(i+j-1)\binom{S''}{i}\binom{S''-i}{j}\right] + \ldots.$$

Here S', S'', ... denote the number of remaining streams at the end of each elimination stage. The Sargent-Westerberg algorithm requires about

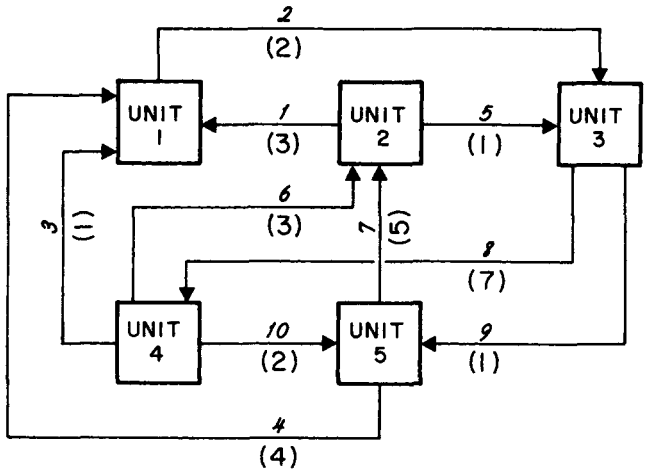

Fig. 4. Example Process I.

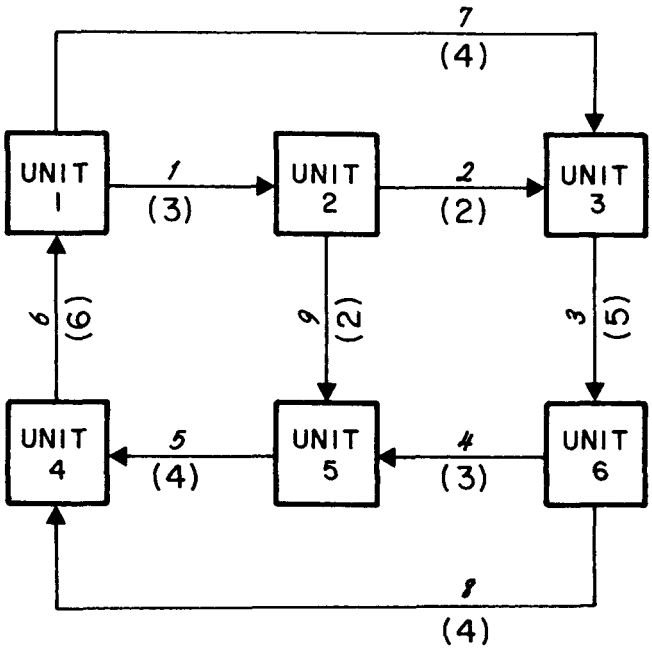

Fig. 5. Example Process II.

$$\sum_{i=2}^{v}\left[\left\{3+\frac{1}{2}\binom{v}{i}\right\}\left\{i\binom{v}{i}\right\}\right]$$

logical operations and

$$\sum_{i=2}^{v}\left[\left\{\frac{S}{v}(i+1)+2\right\}\left\{i\binom{v}{i}\right\}\right]$$

arithmetic operations, but does not require the once only determination of a cycle/stream matrix.

For a process with 15 units, 20 cycles and 50 streams, the Lee-Rudd algorithm would require up to 2×10^9 logical and 5×10^8 arithmetic operations. Use of the Sargent-Westerberg algorithm would require roughly 5.8×10^8 logical and 7.8×10^6 arithmetic operations. The same problem would require, *at the most*, 5.4×10^7 logical and 2.5×10^7 arithmetic operations with the present algorithm; in most cases the requirements would be much smaller. This represents a computational advantage of at least one order of magnitude over the other two algorithms for such large systems.

Examples

Because of the different structure of these three algorithms, it is not possible to compare quantitatively the number of operations they respectively require for a truly general case. Meaningful comparisons are better carried out for specific cases, of which two, designated as Process I and Process II and shown in Figures 4 and 5 respectively, are considered here. Process I has been previously used by Lee and Rudd to illustrate their algorithm. The number of operations required by these three algorithms for these two cases are shown in Tables 3 and 4 respectively. In addition computation of the cycle/stream matrix by the method of Tiernan (*10*) requires 858 total operations for Process I and 1038 for Process II. In the case of Process I, with the Lee-Rudd algorithm, no sets containing more than 3 streams were considered; for Process II the limit was 2 streams. In the absence of such limits, the required number of operations would have been much greater for the Lee-Rudd algorithm.

It can be seen from these tables that in both cases the present algorithm is significantly more efficient than the

Table 3. Comparison of Efficiency of Decomposition of Algorithms Applied to Process I

Type of operation	Number of operations required: Lee-Rudd algorithm (*1*)	Sargent-Westerberg algorithm (*3*)	Dynamic programming algorithm
Logical	2200	677	1197
Arithmetic	650	820	240
Total	2850	1497	1437

Table 4. Comparison of Efficiency of Decomposition of Algorithms Applied to Process II

Type of operation	Number of operations required: Lee-Rudd algorithm (*1*)	Sargent-Westerberg algorithm (3)	Dynamic programming algorithm
Logical	1044	1926	182
Arithmetic	301	1650	63
Total	1345	3576	245

Lee-Rudd algorithm; the number of operations required is smaller by about 40% and 80% for Process I and Process II respectively. For Process I the Sargent-Westerberg procedure requires about the same number of operations as the present algorithm if the determination of the cycle stream matrix is not included. This can be attributed to the fact that in Process I, the number of units is much smaller than the number of cycles. In the case of Process II, where the number of cycles is slightly smaller than the number of units, the total number of operations required by the Sargent-Westerberg procedure is even greater than that required by the Lee-Rudd algorithm. Thus, even for the very small process systems considered in these examples, the present dynamic programming algorithm demonstrates considerable advantage in efficiency over earlier exact procedures. This advantage would be far greater for large systems, except for the relatively uncommon case where the number of units in an irreducible block is significantly smaller than the number of cycles, where the Sargent-Westerberg algorithm might be superior. Also, if the decomposition procedure is to be applied only once to a process the effort in determination of the cycle/stream matrix can, in some cases, tip the balance in favor of the Sargent-Westerberg algorithm.

LIMITATIONS

The algorithm developed here has, of course, certain potential disadvantages of its own. The storage requirements can tend to be rather large; for a system with a cycles, roughly $3(2^a - 1)$ locations are required. A simple modification in the procedure can reduce this number to

$$3\left[\binom{a}{a/2} + 2\sum_{i=1}^{f/2}\binom{a}{a/2-i}\right]$$

if a and f are even, with similar results for either a or f or both being odd. Here f is the maximum cycle frequency (that is, the number of cycles opened by a stream when torn). This reduction in the number of storage locations will be significant only if f is significantly smaller than a. The number of operations required to implement this modified procedure will, of course, be greater than without the modification. With increasingly larger capacities becoming available in computers, the storage requirement of the present dynamic programming algorithm does not appear to be a serious drawback. Another consideration is that this algorithm, as well as that of Sargent and Westerberg, does not appear to be advantageous for hand calculations; the efficient pattern recognition mechanisms of humans tend to favor the type of operations involved in the Lee-Rudd algorithm.

CONCLUSIONS

Several alternative algorithms can be applied to the determination of optimum decomposition or ordering of recycle processes, defined as the minimization of summed costs or weightings for torn streams. Heuristic algorithms as by Christensen and Rudd (2) can be very efficient when they achieve decomposition, but they offer no certainty that an optimum decomposition will be found by the basic algorithms described. Where calculations are to be performed by hand, the procedure of Lee and Rudd (1) has certain advantages, at least for the small systems where such hand calculations may be envisioned. For cases where a few units are involved in many recycle loops, the method of Sargent and Westerberg (3) is advantageous. However, when relatively large systems are considered, or for that matter, whenever computer implementation of the decomposition procedure is contemplated, the dynamic programming algorithm developed here possesses attractive efficiency, defined in terms of the number of operations required for machine implementation. For cases where perhaps 20 or more streams and up to perhaps 15 cycles are involved, the number of operations required by the present algorithm can be at least an order of magnitude less than would be required by the other exact methods. The dynamic programming algorithm is also very well suited to computer implementation, which should also enhance its utility.

ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation.

NOTATION

a = number of cycles in system
$B(l)$ = cost of stream l
$\{c\}_J$ = set of cycles opened by the set of torn streams $\{J\}$
$\{C\}$ = set of all cycles in system
$D(\{J\})$ = sum of costs of streams in $\{J\}$
$E(p)$ = optimum cost associated with state p
f = maximum cycle frequency of streams in the process
$g(p)$ = largest index of torn stream at state p
$\{I\}$ = set of all streams in system
$\{J\}$ = any subset of $\{I\}$
$\{J\}_i$ = subsets of $\{I\}$ opening same cycles as $\{J\}$
$\{k\}$ = any subset of the set $\{K\}$
$\{K\}$ = any valid set of torn streams
l = stream index number
$\{L\}$ = one-stream set consisting of stream l
$\{M(p)\}$ = optimum set of streams associated with state p
$\{n\}$ = any subset of the set $\{N\}$
$\{N\}$ = optimum set of torn streams
p, q = state index numbers
S = number of streams in system
v = number of units in system

Greek Letters

α_n = number of streams in set $\{n\}$
β = value of i for minimum in Equation (1)
$\{\phi\}$ = null set
$\sigma(\{J\})$ = index of state reached by tearing the set $\{J\}$

LITERATURE CITED

1. Lee, W., and D. F. Rudd, *AIChE J.*, **12**, 1184 (1966).
2. Christensen, J. H., and D. F. Rudd, *AIChE J.*, **15**, 94 (1969).
3. Sargent, R. W. H., and A. W. Westerberg, *Trans. Inst. Chem. Eng.*, **42**, 190 (1964).
4. Ramji, R., M.S. thesis, Univ. California, Berkeley (1967).
5. Christensen, J. H., *AIChE J.*, **16**, 177 (1970).
6. Bellman, R., and S. E. Dreyfus, "Applied Dynamic Programming," Princeton Univ. Press, N. J. (1962).
7. Berge, C., "Theory of Graphs and its Applications," Wiley, N. Y. (1962).
8. Norman, R. L., *AIChE J.*, **11**, 450 (1965).
9. Steward, D. V., *Soc. Ind. Applied Math. J.*, **2**, Ser. B., 345 (1965).
10. Tiernan, J. C., *Comm. ACM.*, **13**, 722 (1970).

Manuscript received October 12, 1971; revision received December 29, 1971; paper accepted December 30, 1971.