1979

Exclusive tear sets for flowsheets

Rudolphe L. Motard
Carnegie Mellon University

Arthur, W. Westerberg

EXCLUSIVE TEAR SETS FOR FLOWSHEETS

by

Rodolphe L. Motard and Arthur W. Westerberg

May, 1979

DRC - 06 - 7 - 79

EXCLUSIVE TEAR SETS FOR FLOWSHEETS

by

Rodolphe L. Motard[1]
and
Arthur W. Westerberg[2]

[1]Department of Chemical Engineering
Washington University
St. Louis, Mo. 63130

[2]Department of Chemical Engineering/Design Research Center
Carnegie-Mellon University
Pittsburgh, Pa. 15213

Abstract

A new implicit enumeration algorithm locates the minimum weighted tear sets among those which belong to the nonredundant families of Upadhye and Grens [1975]. Theoretical developments support the algorithm and give a new insight relating nonredundancy to the tearing of unit loops in a flowsheet. If all loops can be torn exactly one time, the nonredundant family is unique and a member of it is trivial to find. If not possible, two or more nonredundant families exist and the aoove algorithm discerns among them, picking a tear set which minimizes the maximum number of times any unit loop is torn.

The literature abounds with algorithms to find the "tear" streams automatically for process flowsheeting calculations. Tear streams are those which are to be guessed and iterated in the course of solving a process flowsheet containing recycles, using a so-called sequential modular type of flowsheeting system such as FLOWTRAN (Seader, Seider and Pauls [1974]), CONCEPT [1973] or PACER [1971]. One criterion for selecting tear streams is to select the fewest such streams. The algorithm of Barkley and Hotard [1972] is one of many which solves this problem. Another is to select the minimum weighted tear set where each stream is assigned a weight, and the best tear set is defined as the one which gives rise to the minimum sum of weights associated with the tear streams. Christensen and Rudd [1969] present one of the many algorithms for this criterion.

The best criterion appears to be that of Upadhye and Grens [1975] wherein the tear set is required to belong to a nonredundant family of tear sets. All tear sets in this family are shown to have the same convergence behavior using successive substitution, and Upadhye and Grens give qualitative arguments, together with numerical evidence, that these tear sets are likely to be better than any others. Rosen and Pauls [1977] support this contention with further numerical evidence using the well-known Cavett problem (Cavett [1963]).

We shall show in this paper that the nonredundant family of tear sets is directly related to the unique tearing of unit loops within the flowsheet. We shall discover that the inability to find a tear set which tears all unit loops exactly one time gives rise to more than one nonredundant family in the Upadhye and Grens sense. Such a discovery allows us to distinguish among nonredundant families and argue qualitatively that some of these should have better convergence properties than others.

Finally ve give an algorithm to find the minimum weighted tear set from among those in the nonredundant families deemed best by the above arguments. If only a member of a nonredundant family is desired and that family is unique, a trivially easy algorithm is available, provided the unit loops are available.

Theory

The structure of a flowsheet can be captured in an obvious manner using graph theoretic concepts. We shall assume we are dealing with an irreducible subset of units within the flowsheet, wherein all the units in the subset must be solved together because of recycles. We shall make the following standard definitions to aid us.

A digraph G(N,E) is a directed graph comprising a set of nodes and directed edges connecting those nodes, where

$N = \{n_i \mid n_i$ is a node in G$\}$

$E = \{e_j \mid e_j$ is a directed edge in G running from a source node n_s to a destination node n_d, n_s, n_d are members of N$\}$

Clearly the nodes relate to the process units in a process flowsheet and the directed edges to the connecting process streams.

A path p(jt) is a string of nodes and edges in G of the form

$$p(jt) = n_1 e_1 n_2 e_2 \cdots n_m e_m n_{m+1}$$

where n_k is the source node and n_{k+1} is the destination node for edge e_k.

A node loop, u(jt) > is a path where $n_1 = n_{m+1}$.

A simple node loop is a node loop which does not contain two or more node loops within it.

A tear set ET(jfc) is a set of edges with the properties

1) If the edges in ET(/) are deleted, G will contain no node loops.

2) If any single edge in ET(X) is not deleted while the remaining ones are, G will contain node loops.

We shall now define a covered node loop o(n) > ^iere u(n) is a simple node loop in G. To do this we present the following algorithm,

Algorithm I

I. Select any (tear) edge e_p which is not a member of u(*0 .

II. Flag edge e_p and all edges which are in any simple node loop with it. Some of these flagged edges may be in u(n) •

III. Repeat, selecting another (tear) edge e_σ which is not a member of u(n) and is not yet flagged. Continue until the edges in all other node loops with any edges in common with \j(n) are flagged.

If no edge in u(*0 remains unflagged, we shall define u(n) as a covered node loop. Clearly then u(*0 can be a covered node loop if and only if a subset of k of the (tear) edges selected by Algorithm I have the following properties.

1) None of them is in a simple node loop with any of the rest of them.

2) Each tear edge is responsible for flagging some of the edges in u(*0 by being in simple node loops with these edges.

3) As a set they must cause all edges in u(*0 to be flagged

Consider two nodes n_1 and n_2 connected by the paths p(l) and p(2). See Figure la. Path p(l) goes from n^ to n_2 and p(2) from n^ to i^, Let u(l) = P(l) P(2) be a covered node loop. We must be able to locate a second simple node loop, u(2), which contains p(l) but not p(2) ; therefore we need a second path p(3) from n_2 to i^ with u(2) = p(l) p(3) resulting. In addition we need a simple node loop, u(3), containing p(2) and neither p(l) nor p(3). We therefore require a second path p(4) from n^ to n_2 with u(3) = p(2) p(4) , but we find we have also formed the node loop u(4) = p(3) p(4).

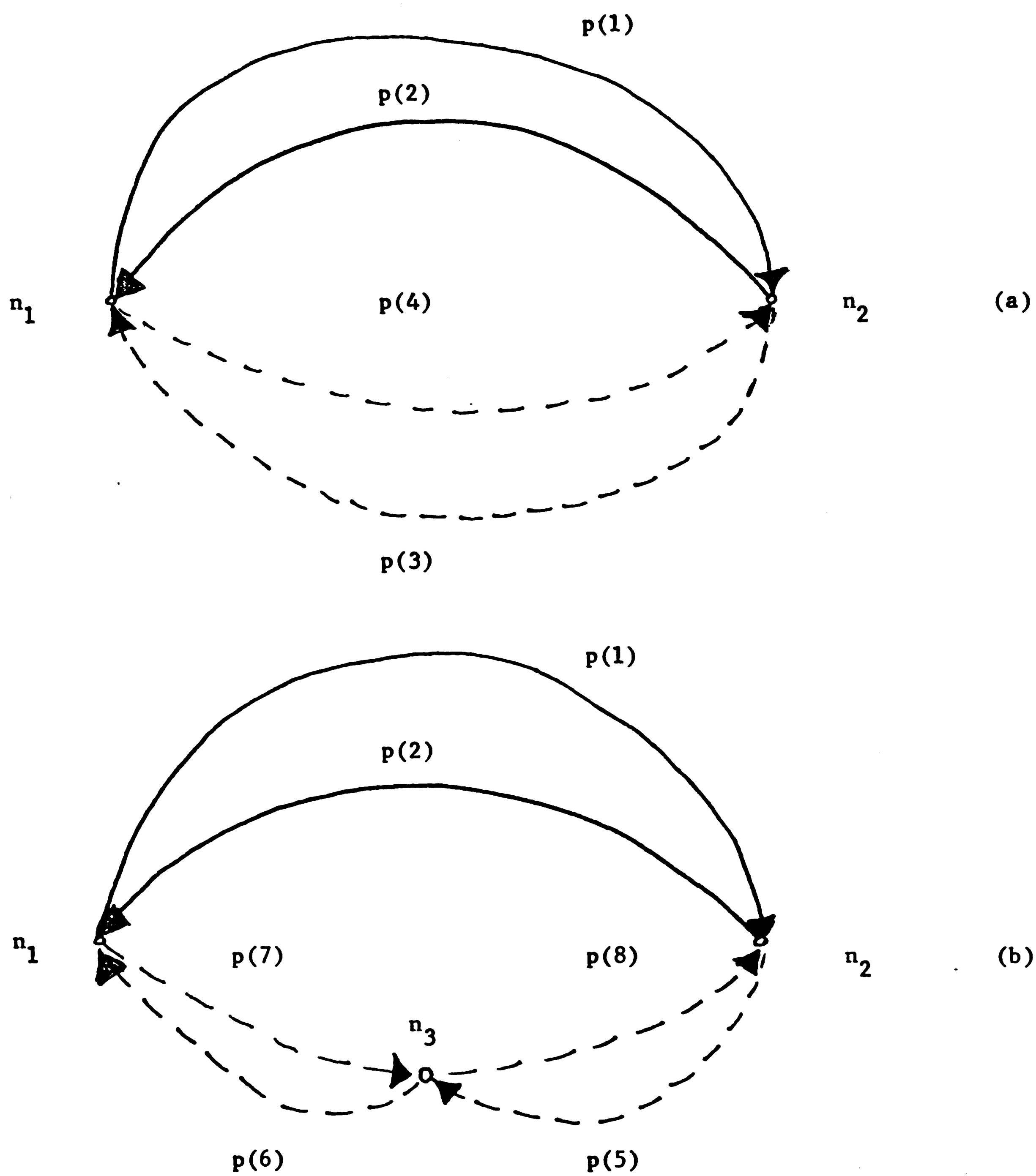

Figure 1. The Minimum Structure Required in a Digraph for a Covered Loop to Exist.

Selecting an edge from p(3) will cause us to flag all edges in p(l) but will also cause us to flag all edges in p(4). Clearly both p(3) and p(4) must be created if u(l) is to be covered. The node loop p(3) p(4) must result; it cannot be a simple node loop therefore (a simple node loop contains no other node loops). Paths p(3) and p(4) must pass through a common third node, n_3, which breaks p(3) into two paths, p(5) and p(6), and p(4) into p(7) and p(8) as illustrated in Figure lb. The node loop p(3) p(4) then contains two node loops and is no longer simple.

The simple loops for Figure lb are (p(l) p(2)), (p(6) p(7)), (P(5) p(8)), (p(l) p(5) p(6)) and (p(2) p(7) p(8)). To cover loop u(l) = p(l) p(2) we select one edge each from p(5) and p(7) (or from p(6) and p(8)). Selecting an edge from p(5) flags all edges in p(l), p(5), p(6), and p(8). Then selecting an edge from p(7) flags all edges in p(2) and p(7). Thus u(l) = p(l) p(2) is indeed covered.

Result 1: If a digraph contains no covered loops, an edge tear set can always be developed which tears each node loop exactly one time.

proof: Using Algorithm I, no untorn loop can be encountered that has no unflagged edges remaining. If one does, then the loop is covered, which contradicts our assumption. Algorithm I will therefore find the desired tear set. Q.E.D.

We shall call such a tear set when it exists an exclusive edge tear set.

We note some properties of the structure in Figure lb which are required for a covered node loop to exist. At least three nodes must exist with each having at least two input edges and two output edges.

Result 2: If a digraph G(N,E) contains fewer than three nodes, each with both multiple inputs and outputs, it cannot contain a covered node loop and an exclusive tear set can always be found.

Proof: The proof is a direct consequence of the above observation. Q.E.D.

We shall call the structure in Figure lb a cyclic cascade and note that if and only if one exists in a digraph, then so does a covered node loop. Apparently such a structure is not commonly found in the digraph corresponding to a process flowsheet so an exclusive edge tear set exists for the digraphs corresponding to most flowsheets and is readily found by Algorithm I. Anticipating the connection between exclusive tear sets and unique nonredundant families, we note that Upadhye and Grens discovered but one flowsheet with more than one nonredundant family out of several hundred tested, and this flowsheet was highly heat integrated. (Heat exchanges have two inputs and two outputs each.)

We now wish to prove the following theorem.

Theorem 1: If and only if a digraph G(N,E) has an exclusive tear set, then the nonredundant family of edge tear sets as defined by Upadhye and Grens is unique, and each member of it is an exclusive tear set with no other nonredundant family of tear sets existing.

Proof: IF PART: We shall first need to define a nonredundant family of edge tear sets which we shall do using the Upadhye and Grens replacement rule. This rule states that an edge tear set is transformed to another in the same family by first identifying a node which has all of its input edges in the tear *set.* These edges are deleted and replaced by all the output edges of that node. The family is nonredundant if, after exhaustive application of the replacement rule, no edge already in the tear set is also introduced by the replacement rule. This double listing of an edge is redundant, and the algorithm to find a nonredundant family by this approach states that one should delete all but one listing of the repeated edge, moving to a new family and then continue. The above steps are repeated until a family is found which is shown to be nonredundant.

Let us assume an exclusive tear set exists. Then each simple node loop is torn exactly one time (see Result 1 and definition of an exclusive tear set). The replacement rule replaces all of a node's input edges by its output edges. All simple node loops passing through that node are torn once by these input edges before the replacement by assumption. Clearly they are <u>all</u> torn exactly once after the replacement by the output edges. Thus if the replacement rule starts with an exclusive edge tear set, it can only generate exclusive edge tear sets.

We shall next prove that, if an exclusive tear set exists, repeated application of the replacement rule will transform it into <u>all</u> others. Assume that exclusive edge tear set ET(1) exists. Generate any other by use of Algorithm I and call it ET(2). We shall show that one can transform ET(1) into ET(2) by systematic application of the replacement rule, and, since ET(1) and ET(2) are arbitrary exclusive tear sets, we prove the "if Part" of our theorem. Order the simple node loops, and then, for each such loop, identify the edge which tears it in ET(1) and the edge which tears it in ET(2). We apply the following algorithm.

1. Select a simple node loop and call it u(l)•

2. Move the tear for u(l) forward around u(l) via the replacement rule until either (a) the tear for the loop reaches the desired location for it in ET(2) or (b) a node n(l) is encountered with multiple inputs.

   a. If (a) is true, select a node loop whose tear is not yet in the position desired in ET(2) and repeat from 2. Call this node loop $u^f(1)$•

   b. If (b) is true, continue with step (3).

3. Discover a loop u(2) with the properties (1) the loop passes through n(l), (2) the loop is not torn by the tear for u(l) and (3), if u(l) and u(2) have conmon portions, these common portions are connected. (If a node loop exists which satisfies properties (1) and (2) but not (3), then another node loop exists which satisfies property (3), and it is to be the one selected. Figure 2 presents an example. Figure 2a shows our digraph. Figure 2b identifies node loop u(l) and Figure 2c, node loop u(2). Note that u(2) has two disconnected portions in common with u(l) • For this example we replace the parallel path in u(2) which connects the two common portions by the part of \j(l) existing between these portions and find a loop 0(2) which satisfies property 3.)

4. Merge portions of u(l) and u(2) which are in common, as illustrated in Figure 3, into one supernode. Clearly the tear for u(2) cannot be in the common portions for otherwise u(l) would be doubly torn.

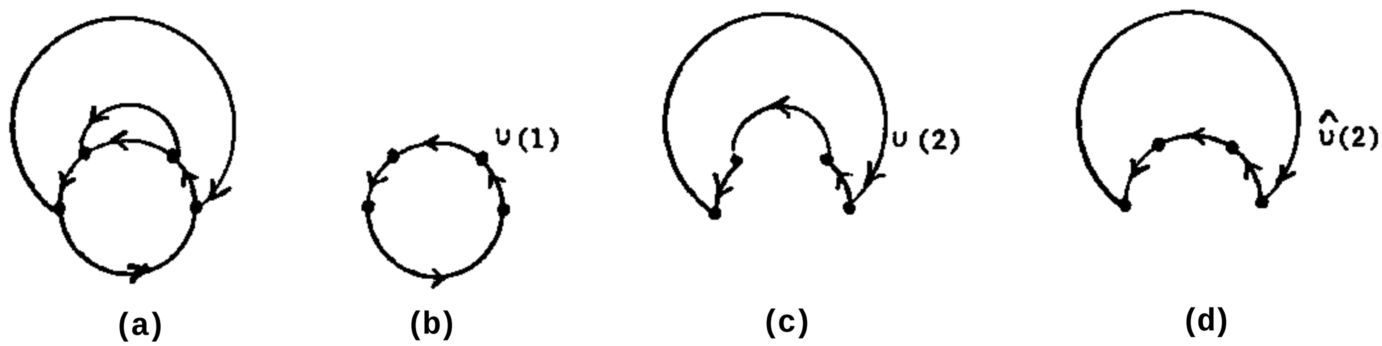

Figure 2. Finding a Loop Satisfying Property 3.

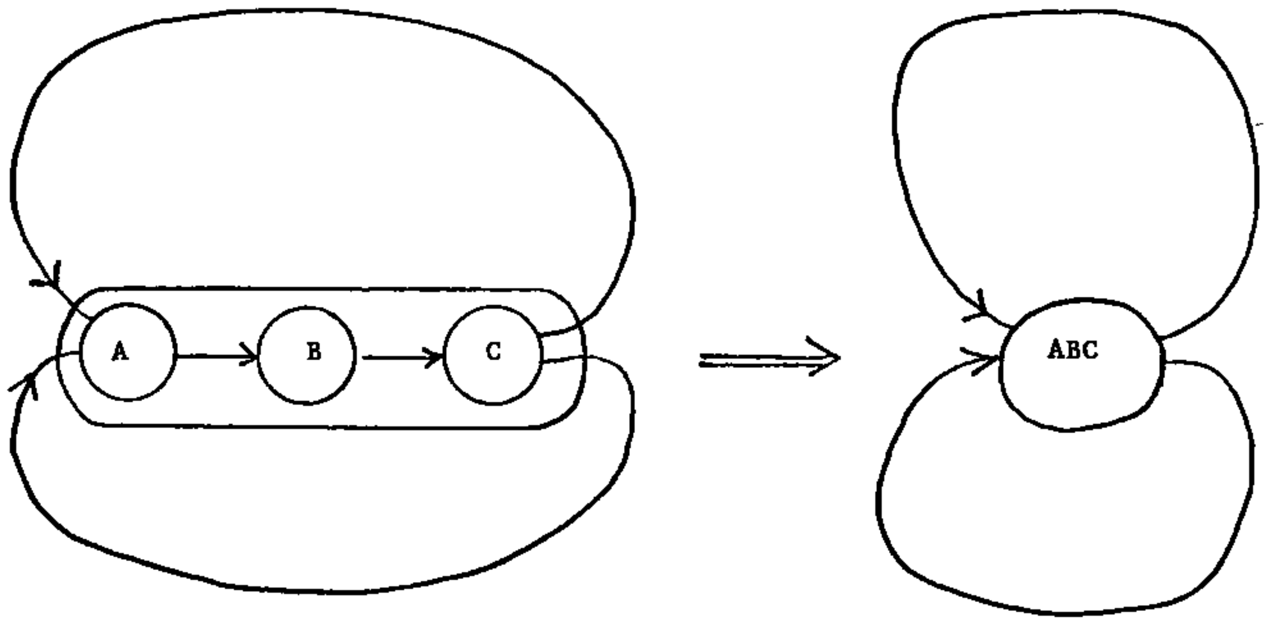

Figure 3. Merging Common Portions of Two Loops into a Supernode.

5. Starting now with u(2), move the tear forward via the replacement rule until either (a) node n(l) (which may now be in a supernode) is encountered or (b) a node n(2) is encountered which has multiple inputs.

   a. If (a) is true then

      1. Repeat from 3 if n(l) has an as yet untorn input edge. Call the new loop found $u^f(2)$.

      2. Otherwise all edges to n(l) are now in the tear set so we can move the tear set through n(l) via the replacement rule. If n(l) is in a supernode, unmerge the supernode and move the tear set through n(l) via the replacement rule. Continue with step 2.

   b. If (b) is true continue with step 6.

6. Find a loop u(3) with the properties (1) the loop passes through n(2), (2) the loop is not torn by the tears for u(l) nor u(2) and (3), if u(l), u(2) and u(3) have common portions, these portions are connected.

7. Merge portions of u(3), u(2) and/or u(l) which are in comnon into one supernode. (If the supernode includes parts of u(l), then in fact u(3) is not separated from u(l) by u(2) . It should be relabeled as loop $u^f(2)$, a "second (not third) level" node loop relative to u(l).)

8. Start now with u(3) and move its tear forward via the replacement rule until either (a) node n(2) is encountered or (b) a node n(3) is encountered which has multiple inputs.

t Etc.

It should be clear with the above how to continue. The loops u(l)> u(2), • • • • can be discovered only to a finite depth because each must not be torn by the tears of the earlier ones and only a finite number of simple loops occurs in a finite digraph.

The loops u(l), u(2), ... form a "tree" of loops with 0(1) being the root. Figure 4 illustrates. The structure must be a tree for, if it is not, then it will contain a covered loop and an exclusive tear set does not exist. Loops u'(k) and u(k) may contain common edges and nodes, but, since they are not dealt with at the same time, no merging need occur among them unless they both join loop O(k-l) at the same node (or supernode) .

We now argue that, since the loops form a tree structure, we can move the tear for loop u(l) to its position in ET(2) , and then we can return the tears for all loops above u(l) in the three to their original positions unless the tear for u(l) in ET(2) is in the portion of u(l) and u(2) which is in conmon. The tear for u(2) in ET(2) must be that for u(l) in this case, and it need not be moved.

Assume the tear is not in the common portion. Clearly the tear for u(l) can be moved forward via the replacement rule through n(l) and then n^f(l) if necessary to get it to the position desired in ET(2). After moving through n(l) the tear for u(2) just follows n(l) . It can be moved forward anywhere around u(2) without encountering n(l) again, and thus the tear for u(2) can be put back to its original position without moving the tear for u(l) . proceeding up the tree to level 3 the tear for u(3) cannot be in the common section for loops u(l), u(2) and 0(3) for if it were then o(3) would have become a loop at level 2 in our tree. Thus this tear can always be returned to its original position, and it can be done without disturbing u(l) or u(2). Etc.

While we can see that all tears for the loops in the tree structure can be returned to their original places, except perhaps for u(2) vhere it is not necessary if it cannot be done, tears for other loops not in the tree may have been moved because of the moving of the tear for o(l) .

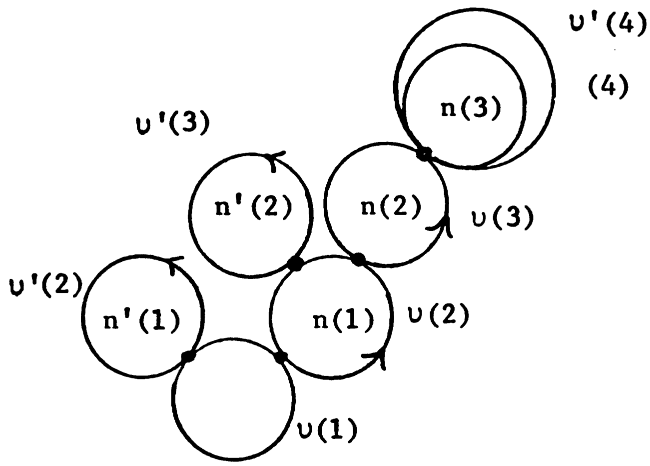

Figure 4. The Tree Structure to the Loops Discovered When Moving Tears for υ(i).

Such tears must come because the tear for u(l) passes through a "diverging" node n(l) which has an edge leaving it which is not a part of any of the loops appearing in the tree of loops. Since all paths leaving u(l) must return to u(l) eventually (because G(N,E) is irreducible), such a path must belong to one (or more) loops which are torn by the tear for u(l) in ET(1) but not in ET(2) . Otherwise the edge would appear in the tree. In other words the edge starts one of more paths which leave u(l) at node n(l) to return at a one of more nodes, all of which follow the desired tear for u(l) in ET(2). Figure 5 illustrates, the original tear for u(l) is shown in Figure 5a, the final desired tear in Figure 5b.

We note that in passing the tear through node n(l), a tear is created along the parallel path at its beginning. Clearly the path must have a tear in ET(2) since the path p(l) is covered by the tear for u(l) and the loop p(l)p(3) must therefore be torn along p(3). Having the tear at the beginning of p(3) guarantees us that the tear in p(3) can be moved anywhere along p(3) and thus to its desired position in ET(2) without affecting the tear for loop u(l)«

The approach is therefore to move the tear for each loop in turn using the above algorithm. Tears are created for parallel paths at their beginning so they may then be moved forward to exactly where needed. Tears in loops created above the loop of interest in the tree structure can always be returned to their original position so previously moved tears can be moved back to their target position if they appear later in the upper levels of a tree for a different loop.

Thus we can take each loop in turn and place the tear for it from ET(1) to its position in ET(2) without moving previously moved tears, and this position may be anywhere in the loop. The digraph has a finite number

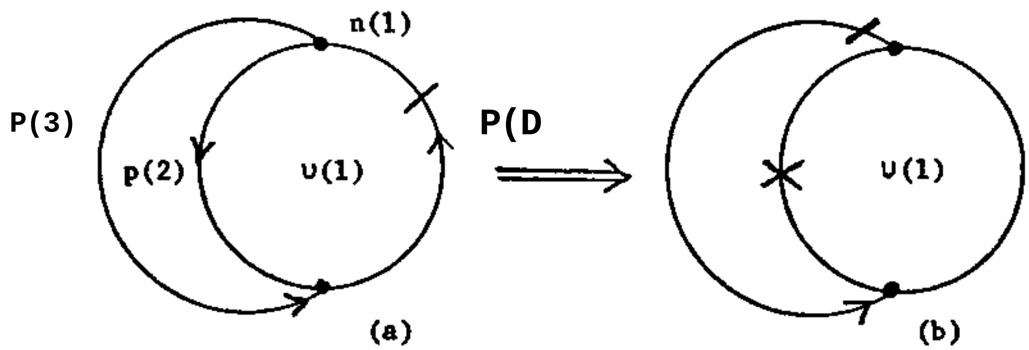

Figure 5. Moving a Tear in u(l) through a Diverging Node.

of simple loops so the process is finite. Thus the replacement rule will generate all the exclusive tear sets starting from any given one.

<u>ONLY IF PART</u>: The proof depended on the nonexistence of a covered loop for otherwise the loop tree in Figure 4 would cease to be a tree. A loop at a higher level would find itself connected to a loop at a lower level as illustrated in Figure 6.

The tear set indicated by the single strokes cannot be transformed into the one indicated by the x's by systematically using the replacement rule. QED.

An example will illustrate. Consider the graph in Figure 7a. Select u(l) (Figure 7b) as the paths {1,2,7} and move the tear from path 2 to just before node C, a node with two inputs. We discover u(2) an untorn loop passing through node C, It has common edges 1 and 7 with u(l) so Figure 7d is formed by merging these edges, forming a single supernode (comprising nodes C,A,B) which joins the two loops u(l) an(* u(2) .

The tear for loop o(2) is moved to just before supernode C,A,B. Note it creates a tear at the start of path 5 because diverging node D has two output edges. Figure 7e shows the result of expanding the supernode and then applying the replacement rule across node C, creating a tear on paths 4 and 7. We find node A on u(l) has two inputs so we must locate a loop u'(2) (Figure 7f) which is not torn and which passes through C. Figure 7g shows the tree of loops with common edges 1 and 2 merged. Node E has two inputs, so we find loop u(3) but it has edges 8 and 1 in common with both u(2) and u(l). Merging these edges yields the tree in Figure 7i. We see that our supernode is nodes EABC combined and that u(3) really is a second loop tied to loop u(l) at the supernode; u(3) is relabeled $u^{n}(2)$ since it is at the second level only of the tree.

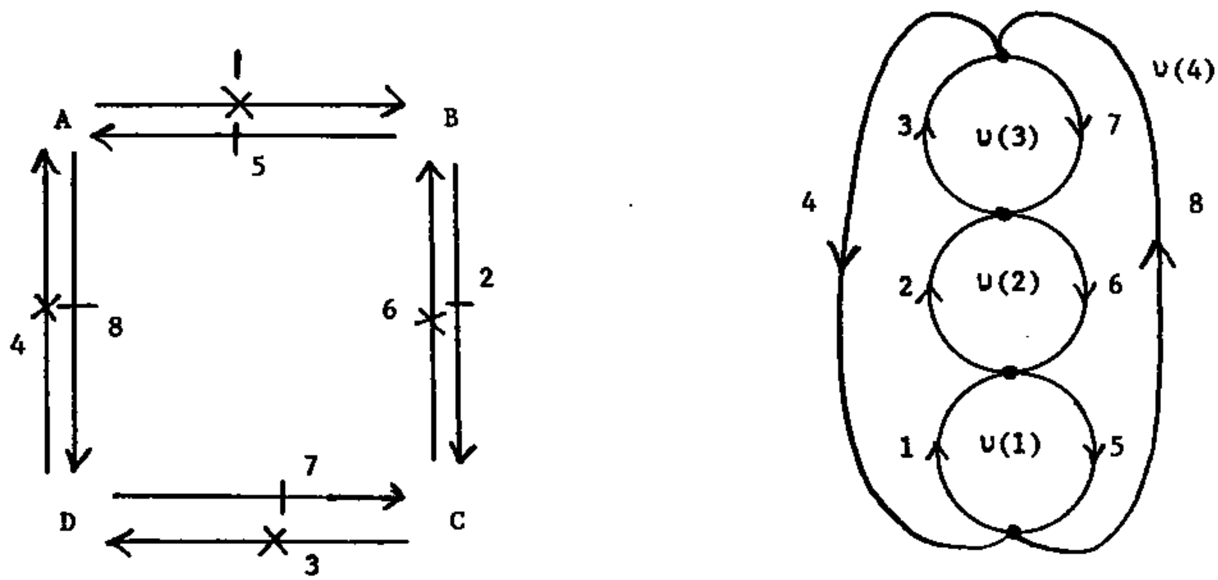

Figure 6. A Digraph Leading to a Nontree Loop Structure.

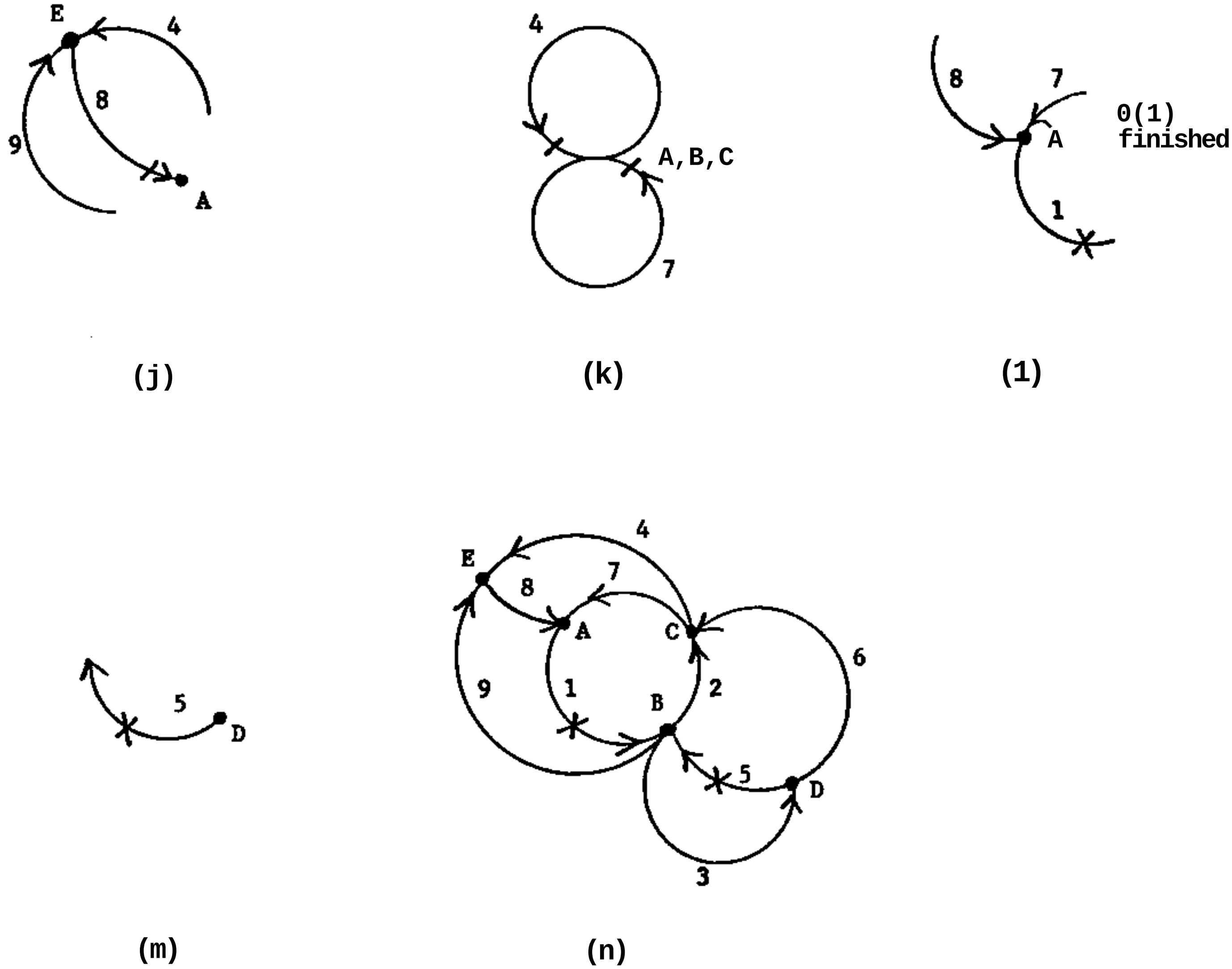

Figure 7. Moving Tears from T(l) = 2,3,9 to T(2) = 1,5

We can now move the tears on 9 and 4 through node E to path 8. At this point both paths into node A are torn, and we can move our tears through A onto path 1 (Figure *7JL)* , our desired goal for the tear for loop u(l). Since the tear on path 1 is in the common part of u(l) with u(2), u'(2) and u"(2), these loops are torn, and we cannot put the tears along these loops back to their original positions — nor should we.

We finally find loop u^f(l) comprising paths 3 and 5 but the tear is already on path 5 from when we moved the tear for loop u(2) through node D. Thus this tear is already in place. Figure 7n shows the final tear set, ET(2), imposed on our network.

Nev Tear Algorithm

We extend the argument of Upadhye and Grens by favoring those tear schemes which tear unit loops the fewest number of times. We would prefer to select tear schemes which minimize the maximum number of times any loop is torn. Thus if an exclusive tear set exists, we want our best tear set to be an exclusive tear set.

We examine the digraph in Figure 6. This digraph contains a looping cascade and thus has no exclusive tear set. We display the simple node loops for this digraph by using a loop/edge incidence matrix. We list each loop along the left border of the matrix and each edge across the top. If an edge appears in the loop, we put a nonblank character (e.g. 'x') in the row for the loop. For the digraph in Figure 6, the loop incidence matrix is

Loop \ Edge	1	2	3	4	5	6	7	8
1	x				x			
2		x				x		
3			x				x	
4				x				x
5	x	x	x	x				
6					x	x	x	x

We can first apply Algorithm I and discover a covered loop. We first remove edge 1 which covers (flags) edges e_1, e_2, e_3, e_4 and e_5 and tears loops u(1) and u(5). We then remove edge e_6 which covers edges e_7 and e_8 and tears loops u(2) and u(6). Clearly loops u(3) and u(4) are covered but not torn. The Upadhye and Grens replacement rule would find three different nonredundant families:

1) based on one tear of loop u(5) and three of loop u(6)

2) based on two tears each of loops u(5) and loop o(6)

3) based on three tears of loop $_o$(5) and one of loop $_u$(6).

We see that families 1 and 3 triple tear a loop whereas family 2 double tears two loops.

The algorithm we wish to propose will select the second family as best because no node loop is torn more than twice for it.

We now devise an algorithm based on implicit enumeration (branch and bound) to locate effectively an edge tear set for an irreducible graph. We shall allow an edge e_j to have a weight W_j assigned to it. We define the multiplicity m(X) of an edge tear set ET(jt) as the maximum number of times any of the node loops are torn. The weight of an edge tear set is the sum of the weights assigned to the edges in the edge tear set ET(/). Our algorithm will locate an edge tear set such that no other edge tear set has a lower multiplicity, and, of all those with the same multiplicity, the selected set has the minimum weight. The algorithm, with explanation, is as follows.

I. For the irreducible digraph, locate all node loops (an algorithm which extends in an obvious and minor way the LOOPFINDER algorithm in Forder and Hutchison [1969] is recommended) and display them in a node incidence matrix, 11.

II. Assign to each loop i a multiplicity $\mu_i = 0$, set level count LEV = 0, set $E_{best} = W_{best} = \infty$, set $\hat{n}$ = number of rows in $\underline{\underline{M}}$, set WTSUM = 0.

III. Assign to each edge e_j the following three numbers:

A. An untorn loop count λ_j, where λ_j is the number of loops which are as yet untorn and which include edge e_j.

B. An edge efficiency η_j, where $\eta_j = \lambda_j / W_j$. η_j equals the number of loops which would be torn per unit of assigned weight for the edge e_j.

C. An edge multiplicity ϵ_j, where

$$\epsilon_j = \max_k \{\mu_k | e_j \text{ appears in loop } \ell_k\}$$

With these numbers we can assess the value of adding edge e_j next to the edge tear set partially completed. If edge e_j is added to the edge tear set, λ_j more loops will be torn with an efficiency per unit of edge weight η_j. At least one loop in the set of all loops will be torn with multiplicity $\epsilon_j + 1$.

IV. Increment LEV by one. Reorder onto a list, List I of level LEV, the indices of all edges in increasing order of multiplicity, reordering edges with equal multiplicity in order of decreasing efficiency.

V. For each edge e_i on the ordered List I, develop a lower bound on multiplicity. The lower bound assumes all edges before e_i on the ordered list are not in the edge tear set and that edge e_i is and all following can be in the edge tear set. To establish the multiplicity bound use e_i and, in order, each of the edges following. If $\hat{n}$ loops reamin to be torn, then the fewest edges needed to tear all the remaining loops, using edges $e_i, \cdots e_p$ would be such that

$$\lambda_i + \lambda_{i+1} + \cdots + \lambda_{p-1} + \theta_1 \lambda_p = \hat{n} \qquad (1)$$

where $0 < \theta_1 \leq 1$. Since this number assumes loops cut by each edge are different from those cut by the other edges, we clearly have a lower bound on the number of edges needed from the sequence used. The multiplicity e_p establishes the multiplicity bound b_i^* for edge i. If an insufficient number of edges exist for (1) to be established for an edge, no multiplicity bound exists for that edge.

VI. Establish, for each edge e_i on the ordered List I for level LEV and for which a multiplicity bound exists, a lower bound on the required sum of weights to complete the edge tear set. For edge e_i, consider all edges following e_i up to e_q where q is the last edge with multiplicity $\mu_q = b_i^*$. Reorder these edges e_{i+1} to e_q in order of decreasing edge efficiency onto a temporary list, List II. Let this list contain the indices $S_1, S_2, \cdots S_{q-i}$. Select e_i and just enough edges in order from List II such that

$$\lambda_i + \lambda_{S_1} + \cdots + \lambda_{S_{t-1}} + \theta_2 \lambda_{S_t} = \hat{n}$$

where $0 < \theta_2 \leq 1$.

The lower bound on the sum of weights for edge e_i is then

$$b_i^W = W_i + W_{S_1} + W_{S_2} + \cdots + W_{S_{t-1}} + \theta_2 W_{S_t} + \text{WTSUM}$$

Note, the bounds b_i^* for edges on List I for level LEV are in increasing order but the bounds b_i^W may not be.

VII. Set NXT(LEV) = 1.

VIII. Set k = NXT(LEV) and increment NXT(LEV) by one. Set £ = k-th index on List I for level LEV.

IX. A. If b^*_ℓ > $_{best}$ or if bf_k does not exist, go to Step XII.

B. If BY_ℓ ^ W_{best}, return to Step VIII.

X. Add edge e£ to the edge tear set as follows. It will be the LEV-th edge in the set.

A. Increment VTTSUM by W£.

B. For each loop X_i in which e£ appears, increment p_i by one.

C. If jj_i just becomes one

1. decrement $\hat{n}$ by one

2. for each edge e_u in loop JL_i, decrement the untorn loop counter by one.

XI. If all loops jfc_i are not yet torn (at least one y_i » 0), return to Step IIIB. Otherwise

A. Set «. = b, and W, = b- for current tear set.

ft W

Desc K Desu K

B. Save current edge tear set as best.

C. Go to Step XIII.

XII. All edges not yet considered on List I at level LEV need not be considered further as they cannot lead to a better tear set than the best found so far.

A. Decrement LEV by one.

B. If LEV = 0, exit algorithm.

C. Set k » NXT(LEV) - 1, and set £ - k-th index on List I for level LEV.

XIII. Remove edge e_ℓ from current edge tear set by

A. Decrement WTSUM by W_ℓ.

B. For each loop u(i) in which e_ℓ appears, decrement p_i by one.

C. If p_i just becomes zero

1. increment $\hat{n}$ by one

2. then for each edge e_n in loop u(i), increment the untorn loop counter λ_n by one.

XIV. Return to Step VIII.

We can illustrate the algorithm with an example, the digraph of Figure 7a. We arbitrarily assign weights to the edges and use the algorithm to find a best edge tear set.

Step I. The five node loops for the digraph in Figure 7a are illustrated in the incidence matrix shown in Figure 8. Loop u(1) contains edges e_1, e_4 and e_7. Weights W_j are assigned across the top to each edge, e.g. edge e_1 has a weight of 5.

Step II. Again looking at Figure 8, we see $p_i = 0$ assigned to each loop along the left border, $\hat{n}$ is 5 here.

Step III. Figure 8 also has λ_j, η_j and α_j values assigned for each edge. $\lambda_1 = 3$ because edge e_1 appears in three as yet untorn loops. $\eta_1 = 3/5 = 0.6$ and $\alpha_1 = \max(p_1, p_4, p_5) = 0$. Since all loops are as yet untorn, all loops torn in the first step will be torn once, i.e. with a multiplicity of $\alpha_j + 1 = 1$.

H	NEdges	X	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	
			2	3	1	5	3	1	1	1	Weights, W_j
0	u(l)	x	x					x			
0	u(2)		x	x			x				
0	u(3)			x		x					
0	u(4)	x	x		x				x		
0	u(5)	x							x	x	

LEV=0	X_j	.	3	3	2	1	1	1	1	2	1
	7)$_j$.	0.6	1.5	0.67	1	0.2	0.33	1	2	1
	ϵ_j	0	0	0	0	0	0	0		0	0

Figure 8. Loop/Edge Incidence Matrix for Digraph in Figure 7a.

Edge	8	2	4	7	9	3	1	6	5
λ_j	2	3	1	1	1	2	3	1	1
W_j	1	2	1	1	1	3	5	3	5
b_j^{ϵ}	0	0	0	0	0	0	0	-	-
b_j^W	3	4	6	6-2/3	H	8	13	-	-

Figure 9. Multiplicity and Weight Bounds for Reorder Edges for Figure 8.

Step IV. Increment LEV to one. List I will be the indices (8,2,4,7,9,3,1, 6,5)- Edge e_8 has the highest efficiency (7)^ 2) and edge e_5 the lowest 0i5- o-2)-

Step V. Figure 9 shows the multiplicity bounds for each edge with edges reordered as done in Step IV. For edge 8 we need to tear at least eg and e_2 to tear n - 5 loops so bg - max(«g,«2) = 0. For edge e_2 we need to tear it and at least edges e_4 and e_7 to tear 5 loops so t>2 = ^ ^ ' ^ ' V *" o# bf = max(e_4, c_7, « ,0 = 0, and so forth. Edge e^ has no bound since e^ and all edges following (e_5) can tear only 2 loops. Similarly e_5 has no bound•

Step VI. In a manner similar to establishing the multiplicity bound for an edge, we establish a weight bound. Since all multiplicity bounds are equal to zero, list II is the same as list I. For the first edge, e_o, we need edge e_6 and e_z at least to tear 5 loops so b^ = W_o+ W_o+ WTSUM = 1 + 2 + 0 = 3. b_2^W= W_2+ W_4+ W_y+ WTSUM = 2 + 1 + 1 + 0 = 4 . For edge $e_?$ we need e_7* e_9* e_3 and only l^3 of e_1 to get 5 loop tears so b^ = W_7 + W_9 + W_3 + W_1^3 + WTSUM = 6-2/3.

Step VII. Set NXT(l) = 1.

Step VIII. Set k = 1, NXT(l) = 2 and £ = 8, the first index on List I for level 1.

Step IX. b_8^* = 0 < E_{best} = 09 and b_8^W = 3 < W_{best} = » so continue to step X.

Step X. We shall add edge e_g as the first edge in the tear set. WTSUM will be incremented from zero to zero + W_g= 3. $y_{,,4}$ and $p_{,5}$ are incremented to unity because e_o appears in loops u(^) an^ u(5) • $\hat{n}$ will be decremented to 5 - 2 s 3 since all loops which are torn are torn for the first time here. The untorn loop counter for e^1 is decremented by 2 since two loops in which it occurs have just been torn, e^2 by 1, e^3 by zero and so forth.

Step XI. For our example y • tu ~ M3 ~ o yet so ^ return to step IIIB.

Steps IIIB and C, IV, V and VI lead to the results in Figures 10 and 11. LEV is reset to 2 in Step IV.

Step VII. NXT(2) = 1

Step VIII.k = 1, NXT(2) =2. k = 7.

Step IX. Continue to Step X.

Step X. Add edge e_7 to the edge tear set.

Step IX will return us to IIIB where Steps IIIB and C, IV, V and V lead to the results in Figures 12 and 13.

Step VII. NXT(3) = 1

Step VIII.k = 1, NXT(3) - 2, k - 3.

Step IX. Continue to Step X.

Step X. Add edge e_3 to the edge tear set. We now find all loops torn (no ^= 0) . So we set e_{best}= b_3^* = 0 and W_{best}= b_3^* - 5.

μ_i		e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
		5	2	3	1	5	3	1		1
0	u_x	X	X					X		
0	u_2		X	X			X			
0	u_3			X		X				
1	u_4	X	X		X				X	
1	u_5	X							X	X
	λ_j	1	2	2	0	1	1	1	0	0
	ν_j	0.2	1	0.67	0	0.2	0.33	1	0	0
	$._j$,	1	1	0	1	0	0	0	1	1

Figure 10. Loop/Edge Incidence Matrix After Tearing Edge 8.

Edge	7	3	6	5	2	1	4	8	9
λ_j	1	2	1	1	2	1	0	0	0
W_j	1	3	3	5	2	5	1	1	1
e_j	0	0	0	0	1	1	1	1	1
b_j^e	0	0	1	1	1	1	1	1	1
b_j^W	5	7	6	8	8	-	-	-	-

Figure 11. Multiplicity and Weight Bounds for Reordered Edges for Figure 10.

μ_i		e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
	w_j	5	2	3	1	5	3	1	1	1
1	u_1	x	x					x		
0	u_2		x	x			x			
0	u_3			x		x				
1	u_4	x	x		x				x	
1	u_5	x							x	x
	λ_j	0	1	2	0	1	1	0	0	0
	η_j	0	0.5	0.67	0	0.2	0.33	0	0	0
	ϵ_j	1	1	0	1	0	0	1	1	1

Figure 12. Loop/Edge Incidence Matrix After Tearing Edge 8 and then Edge 7.

Edge	e_3	e_6	e_5	e_2
λ_j	2	1	1	1
W_j	3	3	5	2
ϵ_j	0	0	0	1
b_j^{ϵ}	0	0	1	-
b_j^W	5	10	9	-

Figure 13. Multiplicity and Weight Bounds for Reordered Edges for Figure 12.

Step XII. No further edges at level 3 need be considered. Decrement LEV to 2 and set k = NXT(2) - 1 - 1 . Set £ = 7.

Step XIII.Remove edge e_7 from the tear set. Decrement ^-1 by one, set $\hat{n}$ - 3, and increment $\backslash_1$, $\backslash_2$ and X_7 by one. Note, we have simply recovered Figure 10 by these steps.

Step XIV. Go to Step VIII.

Step VIII.Set k - NXT(2) = 2, NXT(2) - 3 and £ - 3.

Step IX. See Figure 11. We are now looking at replacing e_7 by e_3 as the second tear, $b_3^{®}$ - 0 - e_{best}" 0. b^ - 7 :≥ W_{best} - 5 so we go to Step VIII. (It would require a tear weight of at least 7 so skip.)

Step VIII. k - NXT(2) - 3, NXT(2) - 4 and £ - 6.

Step IX. bg - 1 > «$_{best}$" 0 (see Figure 11). Putting edge 6 or any following in as a tear would raise the multiplicity of the solution to 2 (a loop would become doubly torn) so we can forget looking at level 2 options. Go to Step XII.

Step XI. Set LEV to l(we should now return to Figures 8 and 9) , k - NXT(l) - 1 - 1, £ - 8.

Step XIII. Delete edge e_g from the tear set.

Step VIII. Set k - NXT(l) - 2, NXT(LEV) - 3 and £ = 2.

Continuing (see Figure 9) we find replacing e_o by e_o will give
o Z
a weight bound of 4 which is less than 5, the best so far. We develop the loop incidence matrix and bounds in Figures 14 and 15 and find b_Q^W at level 2 is already up to a minimum tear weight of 8 so we stop looking with e_2 as the first level tear. The next first level tear option is e_4 (Figure 9), but it has a weight bound of 6 so we can stop altogether. The best tear set is e_{ft}, e_7 and e_1 with a multiplicity of 1 (an exclusive tear set) and a tear weight of 5. Note we examined only alternatives e_3 and e_{fi} (no effort required) at level 2, returned to level 1 and followed one more false trail based on e_2, e_9. Very few options had to be explored to find the best.

		e_1	2	3	4	5	6	7	8	9
1	v_1	x	x					x		
1	U_2		x	x			x			
0	v_3			x		x				
1	v_4	x	x		x				x	
0	v_5	x							x	x
	X	1	0	1	0	1	0	0	1	1
	T|	0.2	0	0.33	0	0.2	0	0	1	1
			1	1	1	0	1	1	1	0

Figure 14. Loop/Edge Incidence Matrix After Tearing Edge 2.

	e_9	e_5
χ	1	1
W	1	5
b_j^e	0	1
b_j^W	8	-

Figure 15. Partially Developed Multiplicity and Weight Bounds for Reordered Edges for Figure 14.

References

Barkley, R.W. and R.L. Motard, Decomposition of Nets, Chem. Eng. J., 3, 265 (1972).

Cavett, R.H., Application of Numerical Methods to the Convergence of Simulated Processes Involving Recycle Loops, Am. Pet. Inst. Repr. No. 04-63 (1963).

Christensen, J.H. and D.F. Rudd, Structuring Design Computations, AIChE J., 15, 94-100 (1969).

CONCEPT Mark III User Manual, CAD Centre, Cambridge, England (1973).

Forder, G.J. and H.P. Hutchison, The Analysis of Chemical Plant Flowsheets, Chem. Eng. Sci., 24, 771-85 (1969).

PACER 245 User Manual, Digital Systems Corp., Hanover, New Hampshire (1971).

Rosen, E.M. and A.C. Pauls, Computer Aided Chemical Process Design, Comput. Chem. Eng., 1, 11 (1977).

Seader, J.D., W.D. Seider and A.C. Pauls, FLOWTRAN Simulation, An Introduction, CACHE Committee, Ulrich's Books, Ann Arbor, Michigan (1974).

Upadhye, R.S. and E.A. Grens, Selection of Decompositions for Process Simulation, AIChE J., 21, 136 (1975).