JOURNAL OF COMPUTER AND SYSTEM SCIENCES 37, 292–311 (1988)

On Locating Minimum Feedback Vertex Sets*

ERROL L. LLOYD AND MARY LOU SOFFA

Department of Computer Science, University of Pittsburgh, Pittsburgh, Pennsylvania

AND

CHING-CHY WANG

IBM Watson Research Center, Yorktown Heights, New York

Received February 24, 1987; revised October 30, 1987

In this paper we study the problem of locating minimum feedback vertex sets in directed graphs. First, we introduce three new transformation-based classes of graphs for which minimum feedback vertex sets can be computed in polynomial time. Second, we delineate an inclusion hierarchy among all of the classes of graphs for which polynomial time feedback vertex set algorithms presently exist. Among the classes of graphs included in the hierarchy are: the reducible flow graphs, the cyclically reducible graphs, the three transformation-based classes that we introduce, and an infinite sequence of classes based on an algorithm of Smith and Walford for finding minimum feedback vertex sets in arbitrary graphs. It follows from our results that one of our new classes, as well as each "Smith/Walford" class, properly includes both the reducible flow graphs and the cyclically reducible graphs. The results presented here serve to unify and focus the work on locating minimum feedback vertex sets in polynomial time.

1. INTRODUCTION

Given a directed graph $G=(V, E)$, a **feedback vertex set** of G is a subset of V containing at least one node from every directed cycle in G. The problem of finding a minimum (cardinality) feedback vertex set of a directed graph is one that arises in numerous applications involving deadlock recovery. Unfortunately, it has been shown that, in general, finding minimum feedback vertex sets is NP-complete [Ka]. Thus, it is natural to consider classes of graphs for which it is possible to find such sets in polynomial time. Earlier, Hopcroft and Ullman [Ho] produced such a result for reducible flow graphs (in [Sh] a linear time algorithm was given). Until recently, this was the only well-characterized class for which such a result was

* This work was supported in part by NSF Grants MCS–8119341 and MCS–8103713 to the University of Pittsburgh.

0022-0000/88 $3.00

known. In [WLS], we characterized a second class, the **cyclically reducible** graphs, and gave a simple $O(ev^2)$ algorithm for finding minimum feedback vertex sets of such graphs (where $e = |E|$ and $v = |V|$). We also showed that there is no particular inclusion relationship between this class and the reducible flow graphs. In [SW], Smith and Walford developed an algorithm for finding minimum feedback vertex sets in arbitrary graphs. In the worst case their algorithm requires exponential time. However, for situations where arbitrary graphs are under consideration, it is the algorithm of choice and is thus of interest.

In this paper our purpose is to unify the work on delineating classes of graphs for which minimum feedback vertex sets may be found in polynomial time. For purposes of explanation, we break the classes that we consider into three groups:

1. The classes previously studied—reducible flow graphs and cyclically reducible graphs (Section 2).

2. Three new classes based on transformations—the two-way reducible, forward reducible, and backward reducible graphs (Section 3).

3. An infinite set of classes based on the Smith/Walford algorithm. We view that algorithm as consisting of a (possibly unbounded) number of stages. Corresponding to halting the algorithm after each stage, there is a well-defined class of graphs (Section 4).

Each of the classes that we study can be defined using some notion of graph reducibility. For each class, it can be shown that:

(a) a minimum feedback vertex set corresponds in a natural way to the particular notions of reducibility used to define the class, and,

(b) certain "Church–Rosser" properties hold with respect to the reducibilities used to define the class.

Together, these two properties are used to formulate natural polynomial time algorithms that serve the dual purpose of recognizing graphs in the class and finding minimum feedback vertex sets of those graphs. In this paper we establish these results for the three new transformation-based classes and for each Smith/Walford class. The results for reducible flow graphs and for cyclically reducible graphs were shown previously [Ho, Sh, WLS].

In the later portion of the paper (Section 5) we delineate a complete hierarchy of inclusion relationships with respect to the (above mentioned) classes, and establish, for each class, its precise position in the hierarchy. A major consequence of this hierarchy is that the reducible flow graphs, the cyclically reducible graphs, and the forward and backward reducible graphs are all properly included in the two-way reducible graphs, as well as in every Smith/Walford class. These results also provide a focus for future work, in that the hierarchy can be partitioned between the two-way reducible graphs and the Smith/Walford classes. Every class in the portion of the partition containing the two-way reducible graphs can be recognized (and a minimum feedback vertex set found) in time $O(e \log v)$ or better, while the best

methods at present for the Smith/Walford classes utilize time $O(v^2e)$ and up (depending on the particular class). This suggests that future work should be directed toward improved algorithms for the Smith/Walford classes and/or identifying new classes that properly include the two-way reducible graphs and are recognizable in (fast) polynomial time.

2. Previously Defined Classes

In this section we review the relevant definitions and results for the two classes that have been previously studied.

2.1. *Reducible Flow Graphs*

We begin our discussion of reducible flow graphs with a definition:

> A **flow graph** is a directed graph (without multiple edges) that has a distinguished node v^* such that there is a directed path from v^* to every other node in the graph.

The equivalence of several alternative notions of reducible flow graph is established in [HU]. The one that we find most useful is based on the following two transformations that may be applied to a flow graph G:

t_1—Given a node x with a self-loop, remove from G the edge that is the self-loop at node x.

t_2—Given a node $x \neq v^*$ with indegree 1 and no self-loop, combine x and its only predecesor y, into a single node. The name of the combined node is y.

In applying t_2, the combining of x and y into a single node means that any node z that was a predecessor of y in G, becomes a predecessor of the combined node. Also, if z was a successor of either x or y in G, then z becomes a successor of the combined node. In addition, any multiple edges are merged into a single edge.

Given a directed graph G, an **RF-sequence** of G is a sequence of transformations (and corresponding nodes of G), such that the transformations can be iteratively applied to the nodes of G in the order in which the transformations appear in the sequence. A **complete RF-sequence** is one that reduces G to a single node.

A **reducible flow graph** is a flow graph having a complete RF-sequence. In [HU], it is shown that the following Church–Rosser property holds with respect to the RF-sequences:

> If G is a reducible flow graph, and transformation t can be applied to node x in G, then there is a complete RF-sequence of G having t applied to x as its first transformation.

We will prove a similar result for each of the other classes that we study.

A linear time algorithm is given in [Sh] both for recognizing reducible flow graphs and for finding a minimum feedback vertex set in such a graph.

2.2. *Cyclically Reducible Graphs*

The cyclically reducible graphs were studied in [WLS]. Here, we begin by briefly reviewing the concepts introduced in that paper.

Given a directed graph $G = (V, E)$, a node z is **deadlocked** if there is a (possibly trivial) directed path in G from z to a node lying on a directed cycle (here, a self-loop is a directed cycle). Note that if node z is deadlocked, then there is a deadlocked node y (not necessarily distinct from z) such that there is a directed edge from z to y. The **associated graph of node x, A(G, x)**, contains node x and all nodes of G that are no longer deadlocked if x is removed from G. The edges of $A(G, x)$ are all edges of G that are incident only to nodes in $A(G, x)$. An example is given in Fig. 1.

An associated graph is **cyclic** if it contains at least one directed cycle. A **D-sequence** for G is a sequence of nodes such that if the associated graphs of the nodes in the sequence are removed from G (incrementally) in the order in which the nodes appear in the sequence, then each of those associated graphs is cyclic. A D-sequence is **complete** if the removal of the associated graphs of the nodes in the sequence leaves G with no directed cycles. Note that by the definition of an associated graph, a complete D-sequence of a graph G containing at least one cycle will leave G empty. If the original G has no directed cycles, then the empty sequence is the only complete D-sequence of G. Finally, a graph G is **cyclically reducible** if there exists a complete D-sequence for G.

In [WLS] we proved the following results:

THEOREM 1. *If $(y_1, ..., y_k)$ is a complete D-sequence for G, then $\{y_1, ..., y_k\}$ is a minimum feedback vertex set for G.*

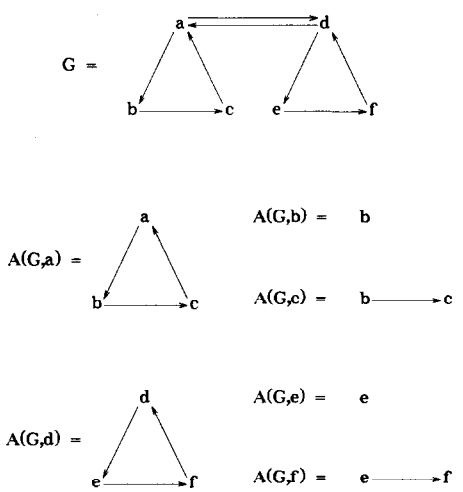

FIG. 1. Associated graphs.

THEOREM 2. *If G is a cyclically reducible graph having a node x such that $A(G, x)$ is cyclic, then there is a complete D-sequence for G having x as its first node.*

The second theorem establishes a Church–Rosser property for D-sequences that makes the definition extremely useful from a computational point of view. In particular, we have the following algorithm for (simultaneously) recognizing a cyclically reducible graph and finding a minimum feedback vertex set for such a graph:

> Given G, choose a node x such that $A(G, x)$ is cyclic; remove $A(G, x)$ from G, place x into the feedback vertex set, and repeat the process on the now reduced G. The algorithm completes when G has no deadlocked nodes or when no associated graph is cyclic.

It follows from Theorems 1 and 2 that the algorithm completes with G having no deadlocked nodes if and only if G is cyclically reducible, and in that case, the set of nodes whose associated graphs were removed, form a minimum feedback vertex set of G. Although this algorithm can be easily implemented to run in time $O(ev^2)$, the running time can be significantly improved using the results of Section 5. Based on those results there is an $O(e \log v)$ algorithm for recognizing, and finding minimum feedback vertex sets of, cyclically reducible graphs.

Before concluding this section, we note for future use, that there is a relatively simple procedure for determining the nodes in $A(G, x)$. In particular, if x is removed from G, then some node of $A(G, x)$ must have outdegree zero, unless x itself is the only node in $A(G, x)$. Using this observation, the process of locating all of the nodes in $A(G, x)$ becomes one of iteratively removing from G nodes of outdegree zero until no nodes of outdegree zero remain. $A(G, x)$ contains node x and each of the nodes removed by this process [WLS].

3. THREE TRANSFORMATION-BASED CLASSES

In this section we describe three new classes[1] of graphs based on simple and easily recognizable graph transformations.

3.1. *Definitions of the Classes*

Each class of graphs considered here is based on some subset of the transformations described below. As usual, we assume that G is a directed graph without multiple edges. The transformations are:

Transformation 1. Given a node x with a self-loop, remove x and all edges incident to x from G.

[1] Independently, Levy and Low [LL] formulated a class of graphs identical to our two-way reducible class, showed that the reducible flow graphs are a proper subset of those graphs, and gave an $O(e \log v)$ algorithm for finding minimum feedback vertex sets there.

Transformation 2. Given a node x with outdegree 0, remove x and all edges incident to x from G.

Transformation 3. Given a node x with indegree 0, remove x and all edges incident to x from G.

Transformation 4. Given a node x with outdegree 1 and no self-loop, combine x and its only successor, y, into a single node. The name of the combined node is y.

Transformation 5. Given a node x with indegree 1 and no self-loop, combine x and its only predecessor, y, into a single node The name of the combined node is y.

We omit the details of combining x and y into a single node (as done in applying transformations 4 and 5) since they are analogous to the details for t_2 in the case of reducible flow graphs.

We use $T(G, x)$ to denote the graph resulting from the application of transformation T to node x of graph G. Also, we find it convenient to refer to transformation 1 as the **loop-transformation**, to transformations 2 and 3 as **0-transformations,** and to transformations 4 and 5 as **1-transformations**.

A key concept in what follows is that of a **T-sequence**. Loosely, a T-sequence of G is a sequence of pairs of transformations and nodes such that each transformation can be applied to its corresponding node when the transformations are applied successively in the order in which they appear in the sequence. More formally, a **T-sequence** of G is a sequence of ordered pairs of transformations and nodes such that if $\langle \mathrm{T}, x \rangle$ is the jth ordered pair, then x is a node in G_{j-1}, and transformation T is applicable to node x in G_{j-1}. Here, $G_0 = G$, and $G_j = \mathrm{T}(G_{j-1}, x)$, where $\langle \mathrm{T}, x \rangle$ is the jth pair of the sequence. A T-sequence is **complete** if the graph is empty after applying the entire sequence. A graph G is **two-way reducible** if there is a complete T-sequence for G.

The other two classes of transformation-based graphs are subclasses of the two-way reducible graphs and are of interest primarily due to their relationships (as delineated in section 5) to the reducible flow graphs and the cyclically reducible graphs. We define an **F-sequence** to be a T-sequence containing (only) transformations 1, 2, and 4 (self-loop, outdegree 0, and outdegree 1). An F-sequence is **complete** if the graph is empty after applying the entire sequence. A graph G is **forward reducible** if there is a complete F-sequence for G. Similarly, a **B-sequence** is a T-sequence containing (only) transformations 1, 3, and 5 (self-loop, indegree 0, and indegree 1). As above, a graph G is **backward reducible** if there is a complete B-sequence for G.

In Section 5, we will show that every reducible flow graph is backward reducible, and that a graph is cyclically reducible if and only if it is forward reducible. As a consequence, the class of two-way reducible graphs includes both the reducible flow graphs and the cyclically reducible graphs.

3.2. *Minimum Feedback Vertex Sets and Church–Rosser Properties*

We begin this section by showing how minimum feedback vertex sets are related to complete T-sequences:

THEOREM 3. *If G is a two-way reducible graph and τ is a complete T-sequence for G, then $\{y_1, ..., y_k\}$ is a minimum feedback vertex set of G, where $y_1, ..., y_k$ are the nodes of G to which the loop-transformation is applied in τ.*

Proof. Inductively, we assume that the theorem holds for all graphs with fewer nodes than G. We begin by considering $\langle T, x \rangle$, the first element of τ. We let τ' be identical to τ with that first element omitted, and let graph $G' = T(G, x)$. Obviously, τ' is a complete T-sequence for G', and by the induction hypothesis, $Y = \{y_1, ..., y_k\}$ is a minimum feedback vertex set of G', where $y_1, ..., y_k$ are the nodes of G' to which the loop-transformation is applied in τ'. Note that in G, the loop-transformation is also applied to each of $y_1, ..., y_k$. There are three possibilities for T (the first transformation in τ).

Case 1. T is a 0-transformation. Y is also a minimum feedback vertex set of G, since x does not lie on any cycle in G.

Case 2. T is the loop-transformation. Since removing x from G breaks no cycles in G', and the removal of Y from G cannot break the self-loop at x, it follows that $Y \cup \{x\}$ is a minimum feedback vertex set of G.

Case 3. T is a 1-transformation. We claim that Y is also a minimum feedback vertex set of G. Note that establishing that Y is a feedback vertex set of G is enough, since the minimality will follow from the minimality of Y with respect to G'. Thus, consider any cycle $C = (z_1, ..., z_s)$ in G. If x is not on C, then C is also present in G', and is broken by the removal of Y. If x is on C, say $z_i = x$, then, it follows from the definitions of the 1-transformations, that $(z_1, ..., z_{i-1}, z_{i+1}, ..., z_s)$ is a cycle in G'. But then some node z_j from that cycle is in Y. The removal of Y (in particular, z_j) from G breaks cycle C. ∎

Next we show that a Church–Rosser property holds with respect to complete T-sequences:

THEOREM 4. *If G is a two-way reducible graph and transformation T_i is applicable to x in G, then there is a complete T-sequence of G having $\langle T_i, x \rangle$ as its first element.*

Proof. Inductively, we assume that the theorem holds for all graphs with fewer nodes than G. Since G is a two-way reducible graph, there exists a complete T-sequence τ for G. Let $\langle T_s, y \rangle$ be the first element of τ. It follows from the definitions of the transformations that if $y = x$, then $T_s = T_i$, hence the theorem holds. Thus, we assume that $x \neq y$ and consider three cases.

Case 1. No tansformation is applicable to x in $T_s(G, y)$. Since T_i is not applicable to x in $G' = T_s(G, y)$, T_i must be a 1-transformation. We will assume, without loss of generality, that the outdegree of x in G is 1, and that T_i is Transformation 4. Since no transformation is applicable to x in G', the outdegree of x in G' must be at least 2. It follows from the definitions of the transformations that the only way for x to have a larger outdegree in G' than in G, is if T_s is Transformation 5 (the indegree of y in G is 1) and for x to be that sole predecessor of y in G. Then, in G, y is the only successor of x. It follows that $T_s(G, y) = T_i(G, x)$, except that node x in $T_s(G, y)$ is node y in $T_i(G, x)$. Thus, consider a T-sequence τ', which is identical to τ, except that $\langle T_i, x\rangle$ is substituted for $\langle T_s, y\rangle$ as the first element of the sequence and y is substituted for x in the element of τ involving x. Since τ is a complete T-sequence for G, so is τ'.

Case 2. No transformation is applicable to y in $T_i(G, x)$. In a manner analogous to that of Case 1, we again conclude that $T_s(G, y) = T_i(G, x)$ and that the T-sequence τ' specified in Case 1 is a complete T-sequence for G.

Case 3. T_k is applicable to x in $T_s(G, y)$ and T_m is applicable to y in $T_i(G, x)$. By way of induction, we assume that $\langle T_k, x\rangle$ is the second element of τ. Now consider a T-sequence τ' that is identical to τ, except that $\langle T_i, x\rangle$ and $\langle T_m, y\rangle$ are substituted for $\langle T_s, y\rangle$ and $\langle T_k, x\rangle$ as the first two elements of τ. In Lemma 1 (below) we show that $T_k(T_s(G, y), x) = T_m(T_i(G, x), y)$. Since τ is a complete T-sequence for G. it follows that τ' is also a complete T-sequence for G. ∎

LEMMA 1. $T_k(T_s(G, y), x) = T_m(T_i(G, x), y)$.

Proof. We begin by noting that the two graphs in question have the same node set, so without loss of generality, assume that edge (z, w) is present in $T_k(T_s(G, y), x)$ and is not present in $T_m(T_i(G, x), y)$. It follows from the definitions of the transformations that (z, w) is not present in G and is created in the process of transforming G into $T_k(T_s(G, y), x)$. Since neither the loop-transformation nor the 0-transformations create edges, a 1-transformation must be applied in creating (z, w). There are two cases to consider.

Case 1. (z, y) and (y, w) are in G and T_s is a 1-transformation. Since edges (z, y) and (y, w) are present in G, they must also be present in $T_i(G, x)$. Then, since edge (z, w) is not present in $T_m(T_i(G, x), y)$, it must be that T_m is the loop-transformation. This means that the application of T_i to x in G creates the self-loop on y. It follows that edges (z, y), (y, w), (y, x), and (x, y) are all present in G. Thus, in G, both the indegree and outdegree of y are at least two–contradicting the fact that T_s is a 1-transformation.

Case 2. (z, x) and (x, w) are in $T_s(G, y)$ and T_k is a 1-transformation. We begin by considering T_i. Since (z, x) and (x, w) are present in $T_s(G, y)$, it follows that T_i is not a 0-transformation. If T_i is the loop-transformation, then T_k would also have to be the loop-transformation, since the self-loop on x could not be effected by any

transformation applied to y. Thus, T_i is a 1-transformation. It follows that edges (z, x) and (x, w) are not both present in G, since if they were, then edge (z, w) would be present in $T_i(G, x)$ and hence, in $T_m(T_i(G, x), y)$. Thus, we assume without loss of generality that (z, x) is not present in G. Since (z, x) is present in $T_s(G, y)$, it must be that T_s is a 1-transformation and that edges (z, y) and (y, x) are present in G. Moreover, either edge (x, w) or edges (x, y) and (y, w) are present in G. The latter situation is handled by case 1 (since (z, y) and (y, w) are in G and T_s is a 1-transformation), so suppose that edge (x, w) is present in G. Since T_i is a 1-transformation, edge (y, w) is present in $T_i(G, x)$. Since (z, y) is present in G, it is also present in $T_i(G, x)$, and since (z, w) is not in $T_m(T_i(G, x), y)$, T_m cannot be a 1-transformation. Thus, T_m is the loop-transformation. Since T_s is a 1-transformation, it must be that the application of T_i to x in G creates the self-loop on y. It follows that edges (y, x) and (x, y) are present in G. Finally, since T_s is a 1-transformation, there is a self-loop on x in $T_s(G, y)$. Thus, T_k is the loop-transformation—a contradiction. ∎

Based on Theorems 3 and 4, we have the following algorithm for (simultaneously) recognizing two-way reducible graphs and finding minimum feedback vertex sets for such graphs:

> Given G, choose a transformation T and a node x, such that T can be applied to x; update G by applying that transformation; if T is the loop-transformation, then place x into the feedback vertex set; repeat the process on the now transformed G. The algorithm completes when G is empty or when no transformation can be applied.

It follows from Therems 3 and 4, that the algorithm completes with G empty if and only if G is a two-way reducible graph, and in this case, the feedback vertex set is of minimum cardinality. An implementation of the above algorithm that requires time $O(e \log v)$ is given in [LL].

To conclude this section we note that it follows from the proofs of Theorems 3 and 4 that analogous results hold for both forward reducible graphs and backward reducible graphs. It also follows that there are $O(e \log v)$ algorithms for recognizing, and finding minimum feedback vertex sets in, these graphs.

4. The Smith/Walford Classes

The best known algorithm for finding minimum feedback vertex sets in arbitrary graphs is given by Smith and Walford [SW]. As expected, given the NP-completeness of the general problem, that algorithm requires exponential time in the worst case. On the other hand, it is likely to be considerably faster on most graphs than the brute force algorithm, simply because of the manner in which it examines and prunes the search space.

4.1. *Definitions and the Smith/Walford Algorithm*

Before discussing the Smith/Walford algorithm, we require several definitions that relate a graph $G = (V, E)$ and a set of vertices V'. First, we let $V' - G$ denote $V' - V$, and let $V' \cap G$ denote $V' \cap V$. Similarly, we let $G - V'$ be the subgraph of G with node set $V - V'$, and containing all of the edges of G incident only to nodes in $V - V'$. Also, if $G' = (V', E')$ is a subgraph of G, then $G - G'$ is taken to be identical to $G - V'$.

Now we proceed to several definitions specific to the Smith/Walford algorithm. Given $G = (V, E)$, and a set of nodes H, the **SW-associated graph of H,** $A_{sw}(G, H)$, consists of the nodes in H and all nodes of G that are no longer on a directed cycle if the nodes in H are removed from G.[2] The edges of $A_{sw}(G, H)$ are all edges of G that are incident only to nodes in $A_{sw}(G, H)$. Whenever the meaning is clear and there is no conflict with our earlier use, we will use $A(G, H)$ in place of $A_{sw}(G, H)$. A set of nodes H is **essential** in G, if for every set D of $|H| - 1$ nodes in $A(G, H)$, there is at least one directed cycle in $A(G, H) - D$. If H is essential and $|H| \leqslant i$, then H is a W_i**-set** in G.

Using the above terminology, the Smith/Walford algorithm is as follows:

```
FVS ← ∅;        (the feedback vertex set)
i ← 0;          (i indicates the stage)

while G contains at least one directed cycle do
    begin
        i ← i + 1;
        while there is a W_i-set in G do
            begin
                let S be a W_i-set of minimum cardinality in G;
                FVS ← FVS ∪ S;
                G ← G − A(G, S):

            end;
    end;
```

Informally, this algorithm iteratively removes the smallest possible essential sets from G until no directed cycles remain. At the end of the algorithm the graph will be empty, unless the original graph had no directed cycles. Note that the cardinality of the essential sets that are selected by the algorithm in stage i need not all be of cardinality i—once G is modified by removing the associated graph of an essential set, then essential sets of smaller cardinality may exist (hence, the minimum cardinality requirement).

The class **Smith/Walford i-reducible** (SW_i-reducible), consists of all graphs for which the Smith/Walford algorithm can complete in i or fewer stages. If $(S_1, ..., S_k)$

[2] In [SW]. G_F is essentially our $A_{sw}(G, H)$.

is a sequence of W_i-sets removed by the algorithm when recognizing G in i or fewer stages, then $(S_1, ..., S_k)$ is a **complete** W_i-sequence for G.

4.2. *Minimum Feedback Vertex Sets and Church–Rosser Properties*

Smith and Walford [SW] note the following relationship between complete W_i-sequences and minimum feedback vertex sets:

> If $(S_1, ..., S_k)$ is a complete W_i-sequence for G, then $S_1 \cup \cdots \cup S_k$ is a minimum feedback vertex set for G.

It follows from the definition of SW_i, that for G in SW_i, the algorithm **can** make a sequence of choices of essential sets such that it completes in i or fewer stages. This does not guarantee that the algorithm will in fact make those choices—the possibility exists that in some instances the algorithm will make choices forcing it to use more than i stages to recognize the graph. The next (Church–Rosser) result shows this is not the case.

THEOREM 5. *If G is a SW_i-reducible graph and H is a W_i-set in G of minimum cardinality, then there is a complete W_i-sequence for G having H as its first set.*

We prove Theorem 5 by first establishing a modified version of that result. In particular, we define a **pseudo** W_i-**sequence** for G to be a sequence $(S_1, ..., S_k)$ of sets of nodes of G, where each S_j is either the empty set or is a W_i-set in G_{j-1}. Here, $G_0 = G$ and $G_j = G_{j-1} - A(G_{j-1}, S_j)$. A pseudo W_i-sequence is **complete** if there are no directed cycles in G_k. A graph G is **pseudo** SW_i-reducible if there is a complete pseudo W_i-sequence for G. The critical difference between complete pseudo W_i-sequences and complete W_i-sequences is that the W_i-sets in the pseudo W_i-sequences need not be of minimum cardinality. Now we state a lemma very similar to Theorem 5:

LEMMA 2. *If G is a pseudo SW_i-reducible graph and H is a W_i-set in G, then there is a complete pseudo W_i-sequence for G having H as its first set.*

To prove this lemma, we assume that $(S_1, ..., S_k)$ is a complete pseudo W_i-sequence for G. Corresponding to this sequence, we let $G_0 = G$ and $G_j = G_{j-1} - A(G_{j-1}, S_j)$. Now consider the sequence $(H, R_1, ..., R_k)$, where for $1 \leqslant j \leqslant k$, $R_j = S_j - A(G, H)$. Corresponding to this sequence, we let $G'_0 = G - A(G, H)$ and $G'_j = G'_{j-1} - A(G'_{j-1}, R_j)$. For future reference we note that

$$S_j - R_j = S_j \cap A(G, H) \qquad (*)$$

We will prove Lemma 2 by showing that $(H, R_1 ..., R_k)$ is also a complete pseudo W_i-sequence for G. This we do by proving a series of three lemmas in which we establish certain relationships among the sets H, R_j, and S_j. We begin with:

CLAIM 1. *If C is a cycle in $A(G_{j-1}, S_j)$ and $C \cap H = \varnothing$, then C is a cycle in $A(G'_{j-1}, R_j)$.*

Proof. Consider any cycle C in $A(G_{j-1}, S_j)$ such that $C \cap H = \emptyset$. Then C is a cycle in G'_{j-1}, since no node in $H \cup R_1 \cup \cdots \cup R_{j-1}$ lies on C. Since $S_1 \cup \cdots \cup S_j \subseteq H \cup R_1 \cup \cdots \cup R_j$, it follows that C is not a cycle in G'_j. In fact, for similar reasons, no node from C is in G'_j. Thus, C is a cycle in $A(G'_{j-1}, R_j)$. ∎

Before proceeding to the next claim, we note that if C is a cycle in $A(G_{j-1}, S_j)$ and $C \cap H \neq \emptyset$, then C and the set $H \cap A(G_{j-1}, S_j)$ have a common node.

Next we have a result that shows that the inclusion of H as the first set of the sequence causes a certain "balanced" trade-off for each S_j.

CLAIM 2. *For* $1 \leqslant j \leqslant k$, $|H \cap A(G_{j-1}, S_j)| = |A(G, H) \cap S_j|$.

Proof. By way of contradiction, let j be minimal such that

$$|H \cap A(G_{j-1}, S_j)| \neq |A(G, H) \cap S_j|.$$

Case 1. $|H \cap A(G_{j-1}, S_j)| < |A(G, H) \cap S_j|$. We show that this condition contradicts the essentiality of S_j in G_{j-1}. We consider any cycle C in $A(G_{j-1}, S_j)$. If $C \cap H = \emptyset$, then it follows from Claim 1, that some node in R_j lies on C. Thus, the removal of $R_j \cup [H \cap A(G_{j-1}, S_j)]$ from $A(G_{j-1}, S_j)$ breaks all of the cycles that occur there. However, $|R_j \cup [H \cap A(G_{j-1}, S_j)]| = |R_j| + |H \cap A(G_{j-1}, S_j)| < |R_j| + |A(G, H) \cap S_j| = |S_j|$, from (*). This contradicts the essentiality of S_j.

Case 2. $|H \cap A(G_{j-1}, S_j)| > |A(G, H) \cap S_j|$ In what follows, we derive a contradiction to the essentiality of H in G. To do so, we consider three sets of nodes:

$$D_1 = H \cap [A(G_0, S_1) \cup \cdots \cup A(G_{j-1}, S_j)]$$
$$D_2 = H \cap [A(G_j, S_{j+1}) \cup \cdots \cup A(G_{k-1}, S_k)]$$
$$D_3 = A(G, H) \cap [S_1 \cup \cdots \cup S_j].$$

It should be clear that D_1 and D_2 partition H. Also, from the minimality assumption with respect to j and the criterion for Case 2, we have that $|D_1| > |D_3|$ and thus, $|H| > |D_2 \cup D_3|$.

We contradict the essentiality of H in G by showing that the removal of $D_2 \cup D_3$ from $A(G, H)$ breaks all of the cycles that occur there. Thus, let C be any cycle in $A(G, H)$ and assume that C does not include a node from D_2. It follows that some node x from C is in D_1. That is, for some f, $1 \leqslant f \leqslant j$, node x is in $A(G_{f-1}, S_f)$. For x to be in this set, there must be a node y (not necessarily distinct from x) of C in $S_1 \cup \cdots \cup S_f$. But since y is on C, y is also in $A(G, H)$, and hence y is in D_3. Thus, the removal of $D_2 \cup D_3$ from $A(G, H)$ breaks all of the cycles that occur there. ∎

Now note that by assumption, H is a W_i-set in G. Likewise, for the R_j's we can show

CLAIM 3. For $1 \leqslant j \leqslant k$, R_j is a W_i-set in G'_{j-1}.

Proof. Since each S_j is a W_i-set, we know that $|R_j| \leqslant i$. All that remains is to show that R_j is essential in G'_{j-1}. By way of contradiction, consider j such that R_j is not essential in G'_{j-1}. Then there exists D', a subset of $A(G'_{j-1}, R_j)$, such that $|D'| < |R_j|$ and the removal of D' from $A(G'_{j-1}, R_j)$ breaks all of the cycles in $A(G'_{j-1}, R_j)$. It follows from Claim 1 that the removal of $D = D' \cup [H \cap A(G_{j-1}, S_j)]$ from $A(G_{j-1}, S_j)$ breaks all of the cycles in $A(G_{j-1}, S_j)$. Thus, $|D| = |D'| + |H \cap A(G_{j-1}, S_j)| = |D'| + |S_j \cap A(G, H)| < |R_j| + |S_j \cap A(G, H)| = |S_j|$, by (*). This contradicts the essentiality of S_j in G_{j-1}. ∎

That Lemma 2 holds follows from Claim 3 and noting that $\bigcup_{i=1}^{k} S_i \subseteq H \cup (\bigcup_{i=1}^{k} R_i)$. That is, $(H, R_1, ..., R_k)$ is indeed a complete pseudo W_i-sequence for G. Finally, we use Lemma 2 to prove Theorem 5.

Proof of Theorem 5. Assume that G is a SW_i-reducible graph, and that H is a W_i-set in G of minimum cardinality. Let S be a complete W_i-sequence for G. Since S is also a complete pseudo W_i-sequence for G, it follows from Lemma 2 that there is a complete pseudo W_i-sequence S' for G having H as its first W_i-set. Now consider $G' = G - A(G, H)$ and a sequence S'' that is identical to S', except that the first set (namely, H) is omitted. Clearly S'' is a complete pseudo W_i-sequence for G'. Consider Q, a W_i-set in G' of minimum cardinality. Again by Lemma 2, there exists a complete pseudo W_i-sequence for G' having Q as its first set. Note that by appending H to the beginning of this sequence, we have a complete pseudo W_i-sequence for G in which the first two sets, namely H and Q, satisfy the minimality requirement. Proceeding inductively, we produce a complete W_i-sequence for G having H as its first set. ∎

We conclude this section by considering the problem of simultaneously recognizing graphs in SW_i and producing a minimum feedback vertex set for such a graph. Unfortunately, it appears to be quite difficult to implement such an algorithm efficiently—the best algorithm that we know of is the obvious one (using a linear time algorithm to find the strongly connected components when checking for essential sets) and requires time $O(iv^{2i}e)$. In particular, the time for SW_1 is $O(v^2e)$, and the time for SW_2 is $O(v^4e)$. Obviously, these times are substantially more than the $O(e \log v)$ time required for the two-way reducible class.

5. A Complete Hierarchy of Classes

In this section we delineate a complete set of inclusion relationships among the classes we have studied. The major results are:

1. If G is a reducible flow graph, then G is backward reducible.
2. If G is two-way reducible, then G is Smith/Walford one-reducible.
3. G is cyclically reducible if and only if G is forward reducible.

When these results are combined with the trivial inclusion relationships, we have a complete hierarchical classification as summarized in Fig. 2. In addition, each of the inclusion relationships is proper. Examples showing these proper inclusions are relatively easy to produce and, with one exception, are left to the reader. We begin by showing

THEOREM 6. *If G is a reducible flow graph, then G is backward reducible.*

We prove this theorem by showing how to transform a complete RF-sequence (see Section 2.1) into a complete B-sequence. Before giving the formal proof, we note that the transformations in these two types of sequences are quite similar. In fact, reducible flow transformation t_2 and our Transformation 5 are identical. Moreover, both reducible flow transformation t_1 and our loop-transformation operate on self-loops. The difference of course, is that our loop-transformation removes the node and all incident edges, while t_1 removes only the edge that is the self-loop.

In what follows, we let B(G, S) be the graph resulting from the application of a B-sequence S to a graph G. Similarly, RF(G, S) is the graph resulting from the application of a RF-sequence S to G. Now, we have

LEMMA 3. *Given S, a RF-sequence for G, there exists S', a B-sequence for G, such that* B(G, S') *is a subgraph of* RF(G, S).

Note that we do not insist that S be a complete RF-sequence or that S' be a complete B-sequence.

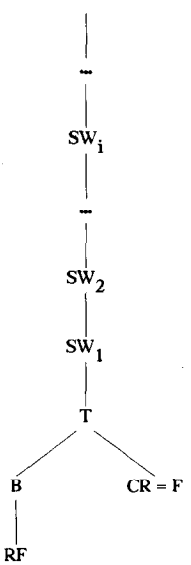

FIG. 2. The hierarchy of class inclusions: B = backward reducible; F = forward reducible; T = two-way reducible; RF = reducible flow; CR = cyclically reducible; SW$_i$ = Smith/Walford i-reducible.

Proof. The proof is by induction on the length of S, the RF-sequence. We assume that the length of S is at least one, and let $\langle T, x \rangle$ be the final transformation of S. Inductively it follows that there exists a B-sequence S'', such that $B(G, S'')$ is a subgraph of $RF(G, S - \langle T, x \rangle)$, where $S - \langle T, x \rangle$ is the RF-sequence S with the final transformation omitted. We consider three cases.

Case 1. $B(G, S'')$ does not contain x. Since x is not present in $B(G, S'')$, we let $S' = S''$. It follows immediately from the induction hypothesis that $B(G, S') \subseteq RF(G, S)$.

Case 2. $B(G, S'')$ contains x and T is t_1—that is, T removes a self-loop from x. If there is a self-loop at node x in $B(G, S'')$ then form S' from S'' by appending the loop-transformation applied to x as the final transformation. If there is no self-loop at x in $B(G, S'')$, then let $S' = S''$. In either case, it follows easily from the induction hypothesis that $B(G, S') \subseteq RF(G, S)$.

Case 3. $B(G, S'')$ contains x and T is t_2—that is, x is merged with y, its only predecessor. In $RF(G, S - \langle T, x \rangle)$, y is the unique predecessor of x. If there is an edge from y to x in $B(G, S'')$, then form S' from S'' by appending Transformation 5 applied to x as the final transformation. If no such edge exists in $B(G, S'')$, then form S' from S'' by appending Transformation 3 applied to x as the final transformation. In the first case, it follows easily from the induction hypothesis that $B(G, S') \subseteq RF(G, S)$. In the second case, it follows from the definition of t_2 that the only edges that are in $RF(G, S - \langle T, x \rangle)$ but are not in $RF(G, S)$, are edges involving node x. But since node x is not in $B(G, S')$, there are also not any edges involving x in $B(G, S')$. Thus, in the second case also, $B(G, S') \subseteq RF(G, S)$. ∎

Now we complete the proof of the theorem.

Proof of Theorem 6. Assume that G is a reducible flow graph and that S is a complete RF-sequence for G. From Lemma 3, there exists S', a B-sequence for G, such that $B(G, S') \subseteq RF(G, S)$. Since there are no edges in $RF(G, S)$, it follows that there are also no edges in $B(G, S')$. Thus, S' can be transformed into a complete B-sequence for G by appending to the end of S', a Transformation 3 for the node in $B(G, S')$. It follows that G is backward reducible. ∎

Next we establish

THEOREM 7. *If G is two-way reducible, then G is SW_1-reducible.*

Proof. The proof is by induction on the number of nodes in G. Thus, we consider G, a two-way reducible graph and assume that the theorem holds for all two-way reducible graphs having fewer nodes than G. In particular, if transformation T is applicable to x in G, then graph $G' = T(G, x)$ is two-way reducible, and hence is SW_1-reducible. Thus, let $(y_1, ..., y_k)$ be a complete W_1-sequence for G'. We will show that $(y_1, ..., y_k)$ is also a complete W_1-sequence for G provided T is not the

loop-transformation. If T is the loop-transformation, then we show that $(x, y_1, ..., y_k)$ is a complete W_1-sequence for G.

By way of preparation, we consider graph G'' that is obtained from G' by removing all nodes of G' that do not lie on any directed cycle in G'. It should be clear that $(y_1, ..., y_k)$ is also a complete W_1-sequence for G''. Note that all of the nodes removed from G' in this fashion are in $A(G', y_1)$.[3] We will often find it convenient to use graph G'' rather than G'. We proceed by cases with respect to T.

Case 1. T is the loop-transformation. As noted above, we claim that $(x, y_1, ..., y_k)$ is a complete W_1-sequence for G. This is clearly the case, provided that $G - A(G, x) = G''$. We consider z, an arbitrary node in G.

If z is not in G'', then z is not on a directed cycle in G'. That is, in G, if z is on a directed cycle, that cycle must also include x (since $G' = T(G, x)$). Thus, z is in $A(G, x)$, and hence z is not in $G - A(G, x)$.

If z is in G'', then there exists a cycle in G'' that includes z. Clearly, this cycle must also be present in G. Moreover, no node of this cycle can be present in $A(G, x)$, since the cycle does not include x. Thus, the cycle (hence, z) is present in $G - A(G, x)$.

Case 2. T is a 0-transformation. In this instance, if all nodes are removed from G that do not lie on a directed cycle in G, the resulting graph is exactly G''. Since $(y_1, ..., y_k)$ is a complete W_1-sequence for G'', it is also a complete W_1-sequence for G.

Case 3. T is a 1-transformation. This case is somewhat more difficult than the previous two, since the effect of applying a 1-transformation is not "local" to G, G', and G'' as in the other cases. In particular, such a transformation may affect some y_i and the associated graph $A(G, y_i)$. Accordingly, we return to the definition of complete W_1-sequences to establish that $(y_1, ..., y_k)$ is a complete W_1-sequence of G. Without loss of generality, we assume that T is Transformation 4. We let z be the only successor of x, and let $P = \{w: w \text{ is a predecessor of } x \text{ in } G\}$.

Corresponding to the sequence $(y_1, ..., y_k)$, we define the following graphs: $G_0 = G$ and $G_i = G_{i-1} - A(G_{i-1}, y_i)$, for $1 \leqslant i \leqslant k$. Similarly, $G'_0 = G'$ and $G'_i = G'_{i-1} - A(G'_{i-1}, y_i)$. To prove that $(y_1, ..., y_k)$ is a complete W_1-sequence, we need to show that each $A(G_{i-1}, y_i)$ contains a directed cycle and that the sequence is complete. The key to showing both of these is to establish a relationship between cycles in G'_i and cycles in G_i. The following two claims establish that relationship.

CLAIM 4. If $C' = (w_1, ..., w_s)$ is a cycle in G'_i, then C is a cycle in G_i, where

$$C = \begin{cases} (w_1, ..., w_j, x, w_{j+1}, ..., w_s) & \text{if } w_j \in P \text{ and } w_{j+1} = z \\ (w_1, .., w_s) & \text{otherwise.} \end{cases}$$

[3] Unless G' is acyclic, in which case the result follows trivially.

Proof. We proceed by induction on i. The basis, for $i=0$, follows in a straightforward fashion from the definition of Transformation 4 and the specification of C, given C'. Thus, assume that the claim holds for G'_{i-1} and G_{i-1}, and let $C'=(w_1, ..., w_s)$ be a cycle in G'_i. Since C' is in G'_i, it follows inductively that the specified C is in G_{i-1}. If C is not in G_i, then some node from C must be in $A(G_{i-1}, y_i)$. This can occur only if y_i itself is in C. But, since $y_i \neq x$, it follows that y_i is in C'. Thus, y_i is in G'_i—a contradiction, since y_i is in $A(G'_{i-1}, y_i)$. ∎

In the above claim, we established that for each cycle in G'_i, there is a related cycle in G_i. In the next claim we show the converse of that result—for each cycle in G_i, there is a related cycle in G'_i, The proof of Claim 5 is analogous to the proof of Claim 4 and is left to the reader.

CLAIM 5. *If $C=(w_1, ..., w_s)$ is a cycle in G_i, then C' is a cycle in G'_i, where*

$$C' = \begin{cases} C & \text{if node } x \text{ does not lie on } C \\ (w_1, ..., w_{j-1}, w_{j+1}, ..., w_s) & \text{if } w_j = x. \end{cases}$$

Having established the above relationships between cycles in G_i and G'_i, we now complete the proof of Theorem 7.

Completion of the proof of Case 3 *of Theorem* 7. As noted prior to the two claims, we show that $(y_1, ..., y_k)$ is a complete W_1-sequence for G, by showing for each i, $1 \leqslant i \leqslant k$, that $A(G_{i-1}, y_i)$ contains a directed cycle and that G_k contains no directed cycle, and hence the sequence is complete. The latter follows immediately from Claim 5 and the fact that G'_k contains no directed cycle. To show the former, we consider (by way of contradiction), the least i such that $A(G_{i-1}, y_i)$ contains no directed cycle. Let C' be an arbitrary cycle in $A(G'_{i-1}, y_i)$. It follows that C' is a cycle in G'_{i-1}, and from Claim 4, that the cycle C as defined there is a cycle in G_{i-1}. Since by assumption, C is not a cycle in $A(G_{i-1}, y_i)$, some node q of C must lie on a cycle D in G_i. Moreover, we can assume that $q \neq x$, since if q is x, then z, the only successor of x, must also lie on both C and D. Thus, we let q be z. Note also that since $q \neq x$, q lies on C'. Now, by Claim 5, cycle D' (as defined in Claim 5), is present in G'_i. In particular, node q lies on G'_i. But this is a contradiction, since q is in C', which means that q is in $A(G'_{i-1}, y_{i-1})$. Thus, $(y_1, ..., y_k)$ is a complete W_i-sequence of G. ∎

As noted earlier, the two-way reducible class is *properly* included in the Smith/Walford one-reducible class. Figure 3 shows a SW_1-reducible graph that is not two-way reducible.

We conclude this section by proving

THEOREM 8. *G is cyclically reducible if and only if G is forward reducible.*

Proof. We begin by assuming that G is forward reducible and prove that G is cyclically reducible. The proof is by induction on the number of nodes in G. Thus,

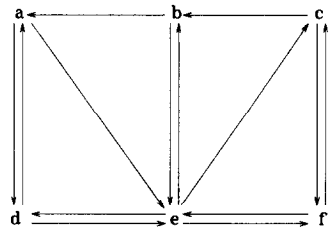

FIG. 3. A graph that is in $SW_1 - T$: G is not two-way reducible since the indegree and outdegree of every node is at least two; G is Smith/Walford 1-reducible since $(\{e\}, \{d\}, \{f\})$ is a complete W_1-sequence for G (since $A_{sw}(G, \{e\})$ contains node b).

we consider G, a forward reducible graph, and assume that the theorem holds for all forward reducible graphs having fewer nodes than G. In particular, if transformation T is applicable to x in G, then graph $G' = T(G, x)$ is forward reducible, and hence is cyclically reducible. Thus, let $(y_1, ..., y_k)$ be a complete D-sequence for G'. We claim that $(y_1, ..., y_k)$ is aso a complete D-sequence for G provided T is not the loop-transformation. If T is the loop-transformation, then we claim that $(x, y_1, ..., y_k)$ is a complete D-sequence for G. Note that this development is quite similar to that in the proof of the previous theorem. In fact, the remainder of the proof is analogous to the proof of Theorem 7 (including results analogous to Claims 4 and 5). This being the case, we leave the details to the reader.

In the remainder of the proof, we show that if G is cyclically reducible, then G is forward reducible. Thus, inductively, we assume that the result holds for all cyclically reducible graphs with fewer nodes than G. Let S be a complete D-sequence for G and let x be the first node in S. Consider the associated graph $A(G, x)$. If that graph consists only of x itself, then x has a self-loop in G. Moreover, it follows that $G - A(G, x) = T_1(G, x)$. We let G' denote $T_1(G, x)$. Then G' is a cyclically reducible graph, that has fewer nodes than G, and by the induction hypothesis, G' is forward reducible. We let S' be a complete F-sequence for G'. Then, by appending $\langle T_1, x \rangle$ as the first transformation of that sequence, we have a complete F-sequence for G, and therefore G is forward reducible.

Next we assume that $A(G, x)$ contains at least one node other than x. Let z be such a node that has outdegree zero in $G - \{x\}$. If z also has outdegree zero in G, then the result inductively holds in a fashion similar to that above, except that Transformation 2 is substituted for the loop-transformation. Thus, we assume that z has outdegree one in G, and consider $G' = T(G, z)$, where T is Transformation 4. We claim that S is also a complete D-sequence for G'. Hence G' is cyclically reducible, and the result follows as above.

To conclude that S is a complete D-sequence for G', it is enough to show that $G - A(G, x) = G' - A(G', x)$ and that $A(G', x)$ is cyclic. Note however, that if we establish the former, then it follows that $A(G, x)$ and $A(G', x)$ are identical, except for node z, which appears in $A(G, x)$. Hence, $A(G', x)$ is cyclic since $A(G, x)$ is

cyclic. Thus we concentrate on showing that $G - A(G, x) = G' - A(G', x)$. We begin by noting that $G - A(G, x)$ consists of all nodes that are deadlocked in $G - \{x\}$ and that $G' - A(G', x)$ consists of all nodes that are deadlocked in $G' - \{x\}$.

Case 1. w is deadlocked in $G - \{x\}$. Thus, there exists a path from w to some directed cycle in $G - \{x\}$. Since z is in $A(G, x)$, z cannot be on that path or cycle. It follows that the path and cycle are also present in $G' - \{x\}$ (note that all of the edges modified in forming G' from G involve z or x, and the edges in the path and cycle involve neither). Thus, w is also deadlocked in G'.

Case 2. w is deadlocked in $G' - \{x\}$. Thus, there exists a path from w to some directed cycle in $G' - \{x\}$. Note that neither x nor z can be on this cycle, since neither is present in $G' - \{x\}$. Again, each edge in the path and cycle must be present in $G - \{x\}$, and therefore w is deadlocked there. ∎

Note that the above result provides an $O(e \log v)$ algorithm for the recognition of cyclically reducible graphs (and the computation of minimum feedback vertex sets for such graphs) by using exactly the algorithm developed for forward reducible graphs.

6. Conclusions

In this paper we have unified the previous work on identifying major classes of graphs for which minimum feedback vertex sets can be found in polynomial time. A major result of this work is the establishment of a hierarchy of proper inclusions with respect to the several classes involved. In addition we have established: (1) certain Church–Rosser properties for the various classes and (2) relationships between the definitions of the classes and minimum feedback vertex sets. For each class there is a polynomial time algorithm for recognizing graphs in the class and finding minimum feedback vertex sets of graphs in the class.

There are three major open questions arising from this work. First, is there a linear time algorithm for recognizing two-way reducible graphs? Second, is there any reasonably fast algorithm for recognizing Smith/Walford i-reducible graphs, and in particular, for the recognition of Smith/Walford one-reducible graphs? Third, is there any class of graphs that properly includes the two-way reducible graphs and for which minimum feedback vertex sets can be found in (small) polynomial time?

Acknowledgments

R. E. Tarjan provided a number of useful suggestions and helpful comments, including our first pointer to the work of Smith and Walford. He also suggested the names for the three new transformation-based classes.

REFERENCES

[HU] M. S. HECHT AND J. D. ULLMAN, Flow graph reducibility, *SIAM J. Comput.* **1**, No. 2 (1972), 188–202.

[Ho] J. E. HOPCROFT AND J. D. ULLMAN, An n log n algorithm for detecting reducible graphs. *in* "Proceedings, Sixth Annual Princeton Conference on Information Science and Systems, 1972."

[Ka] R. M. KARP, Reducibility among combinatorial problems, *in* "Complexity of Computer Computations," (R. E. MILLER AND J. W. THATCHER, Eds.), pp. 85–103, Plenum, New York, 1974.

[LL] H. LEVY AND D. W. LOW, "A New Algorithm for Finding Small Cycle Cutsets," IBM Los Angeles Scientific Center Report G320–2721, 1983.

[Sh] A. SHAMIR, A linear time algorithm for finding minimum cutsets in reducible graphs, *SIAM J. Comput.* **8**, No. 4 (1979), 645–655.

[SW] G. W. SMITH AND R. B. WALFORD, The identification of a minimal feedback vertex set of a directed graph, *IEEE Trans. Circuits and Systems* **CAS-22**, No. 1 (1975), 9–14.

[WLS] C. C. WANG, E. L. LLOYD, AND M. L. SOFFA, Feedback vertex sets and cyclically reducible graphs, *J. Assoc. Comput. Mach.* **32**, No. 2 (1985), 296–313.