An Algorithm for Finding a Minimal Equivalent Graph of a Digraph

HARRY T. HSU

Colorado State University, Fort Collins, Colorado

ABSTRACT It is found that Moyles and Thompson's algorithm contains some mistakes. An efficient algorithm for finding a minimal equivalent graph (MEG) is presented The algorithm proceeds with the following steps First, all the strongly connected (s c) components are found. Then the set of vertices is reordered such that the set of vertices in an s c component is ordered by consecutive integers The rows and columns of the adjacency matrix are permuted accordingly Then an MEG for each s c. component is found Finally, the parallel and the superfluous edges are removed

KEY WORDS AND PHRASES: digraph, algorithm, minimal equivalent graph, adjacency matrix, acyclic digraph, condensed digraph

CR CATEGORIES: 5 32

1. *Introduction*

This paper studies the problem of removing the maximum number of edges from a digraph without affecting the reachability of the digraph. An algorithm for finding a minimal equivalent graph (MEG) was first presented by Moyles and Thompson [1]. It is found that their algorithm contains some mistakes and that it is not efficient.

This paper first discusses an MEG of an acyclic digraph and then an MEG of a strongly connected (s.c.) digraph. These two results are then used to find an MEG of a digraph. In order to find an MEG of a digraph, first all the s c. components are found, then the set of vertices are reordered such that the set of vertices in an s c. component is always ordered by consecutive integers. The rows and columns of an adjacency matrix are permuted accordingly It is found that this will simplify the problem of removing the parallel edges and the superfluous edges.

The reader is advised to read Moyles and Thompson [1]. The terminology used in this paper follows that in Moyles and Thompson [1] as nearly as possible.

2. *Preliminaries*

Let $G = \langle V, E \rangle$ be a digraph, where V is the set of all the vertices where $|V| = N$, and E is the set of all the edges in G, where $|E| = M$. For all vertices v_i and v_j of G, if there is an elementary path from v_i to v_j, it is said that $v_i \mathrel{\mathrm{R}} v_j$. All the paths in this paper refer to elementary directed paths.

An s.c. digraph is one such that for all v_i and $v_j \in V$, $v_i \mathrel{\mathrm{R}} v_j$ and $v_j \mathrel{\mathrm{R}} v_i$. An acyclic digraph is one such that if $v_i \mathrel{\mathrm{R}} v_j$, then $v_j \not\mathrel{\mathrm{R}} v_i$. An acyclic digraph contains no cycles. The following defines an MEG of a digraph.

This research was supported in part by a Colorado State University faculty research grant

Author's address Department of Computer Science, Colorado State University, Fort Collins, CO 80521

Let $G^\circ = \langle V, E^\circ \rangle$ be an MEG of the digraph $G = \langle V, E \rangle$; then the following conditions must be satisfied.

(1) For all v_i and v_j in V, $v_i \mathrm{R}\, v_j$ in G° if and only if $v_i \mathrm{R}\, v_j$ in G.

(2) E° is the smallest subset of E such that condition (1) is satisfied.

It is quite obvious that there can be more than one MEG. This paper presents an algorithm to find one of the MEGs.

If the adjacency matrix X of a digraph G is given, then the path matrix P can be easily found by Warshall's algorithm [2].

If a digraph contains l s.c. components, then the set of vertices V can be partitioned into l subsets, i.e. $V = V_1 \cup V_2 \cup \cdots \cup V_l$, where $V_i \cap V_j = \varnothing$ for $i \neq j$ and $|V_i| = n_i$. Some V_i might have only one vertex. The problem of finding the set of all the s.c. components is exactly the same as partitioning the set of vertices V into l subsets.

An s.c. component is defined as $G_i = \langle V_i, V_i \times V_i \cap E \rangle$.

If the set of vertices are reordered such that in each s.c. component the set of vertices are ordered by consecutive integers, and the rows and columns of adjacency matrix X are permuted accordingly, then X can be partitioned into submatrices as:

$$X = \begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1l} \\ X_{21} & X_{22} & \cdots & X_{2l} \\ \vdots & \vdots & & \vdots \\ X_{l1} & X_{l2} & \cdots & X_{ll} \end{bmatrix}.$$

X_{ii} is of order $n_i \times n_i$; X_{ij}, where $i \neq j$, is of order $n_i \times n_j$. X_{ii} is the adjacency matrix of the s.c. component G_i. If the submatrix X_{ij}, where $i \neq j$ has m nonzero entries, then the m edges shown in X_{ij} are called parallel edges from the ith to the jth s.c. components. If there is a path $(i, \cdots, k)$ of length greater than or equal to 2, and $(i, k) \in E$, then (i, k) is called a superfluous edge.

A condensed digraph (see Berztiss [3]) is defined as:

$$Y = \begin{bmatrix} 0, & y_{12}, & \cdots, & y_{1l} \\ y_{21}, & 0, & \cdots, & y_{2l} \\ \vdots & \vdots & & \\ y_{l1}, & y_{l2}, & \cdots, & 0 \end{bmatrix},$$

where $y_{ii} = 0$ for all i, and $y_{ij} = 1$ for $i \neq j$ if the submatrix $X_{ij} \neq 0$.

The digraph G' with adjacency matrix Y is called a condensed digraph. Each s.c. component in G is represented by a vertex in G'. If (i, k) is a superfluous edge in G', then all the edges shown in X_{ik} are superfluous edges in G.

3. *Presentation of Results*

This section first deals with an MEG of two special classes of digraphs: acyclic and strongly connected. The two results are then used to find an MEG of a general class of digraphs.

A. Acyclic Digraphs. Moyles and Thompson's algorithm A4 is suitable for acyclic digraphs. However, it is too complicated to implement. It requires a maximum of up to $N^3 + \sum_{i=2}^{N-1} \binom{N-1}{N-i}$ steps of computation and a very large computer storage area. The following is a very simple algorithm to find an MEG of an acyclic digraph. In order to apply the algorithm, a path matrix $P = [p_{ij}]$ is also required. The path matrix P can be found by using Warshall's algorithm [1].

Theorem 1. *Let $X = [x_{ij}]$ be the adjacency matrix of an acyclic digraph. If $p_{ij} = 1$*

and $p_{jk} = 1$, then x_{ik}, the (i, k)-th entry of X, can be set to zero without affecting the reachability of the acyclic digraph.

PROOF. If x_{ik} is initially zero, then no edge is removed if x_{ik} is set to zero.

If x_{ik} is initially 1, then there is an edge connecting i and k. If $p_{ij} = 1$ and $p_{jk} = 1$, then there exists a path from i to j and a path from j to k. If (i, k) is an edge in the path from i to j, then $(k \cdots j \cdots k)$ forms a cycle, which is a contradiction. Similarly, it can be shown that (i, k) cannot be an edge in the path from j to k.

Since $p_{ij} = 1$ and $p_{jk} = 1$, the edge (i, k) is a superfluous edge. Therefore x_{ik} is set to zero.

ALGORITHM A1. Finding an MEG of an Acyclic Digraph

```
(AC1) Initially set $X \leftarrow P$.
(AC2) Set $j \leftarrow 1$.
(AC3) Set $i \leftarrow 1$.
(AC4) (a) If $x_{ij} = 1$.
          For all $x_{jk} = 1$, set $x_{ik} = 0$, where $k = 1, 2, 3, \cdots, N$.
          Go to AC5.
      (b) If $x_{ij} = 0$, go to AC5
(AC5) (Update $i$) set $i \leftarrow i + 1$.
      If $i \leq N$, go to AC4. Otherwise, go to AC6.
(AC6) (Update $j$) set $j \leftarrow j + 1$.
      If $j \leq N$, go to AC3. Otherwise go to AC7
(AC7) Terminate the algorithm.
      $X$ is the adjacency matrix of an MEG of an acyclic digraph.
```

THEOREM 2. *If Algorithm A1 is applied to an acyclic digraph, then X is the adjacency matrix of an MEG of an acyclic digraph.*

PROOF. If $p_{ij} = 1$ and $p_{jk} = 1$, then as a result of Theorem 1, x_{ik} should be set to zero. Therefore in step AC4 of Algorithm A1, x_{ik} is set to zero.

Since X is initially set to P, after Algorithm A2 is applied, X might not be the adjacency matrix of an MEG. If it is so, then the resulting digraph G must contain at least a superfluous edge. Let H be a subgraph of G which contains a superfluous edge. H is shown in Figure 1. Clearly the edge (i, k) is superfluous. Let $i_j = \min\{i_1, i_2, \cdots, i_l\}$. Initially $x_{ii_j} = 1$ and $x_{i_jk} = 1$. Then as a result of Algorithm A1, x_{ik} is set to zero. Therefore there is no edge (i, k) after Algorithm A1 is applied, which is clearly a contradiction.

Therefore X must be the adjacency matrix of an MEG after Algorithm A1 is applied.

Example 1, shown in Figure 2, illustrates Algorithm A1.

$$P = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}.$$

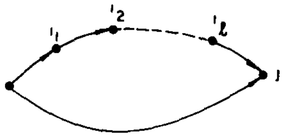

FIG. 1. Subgraph H

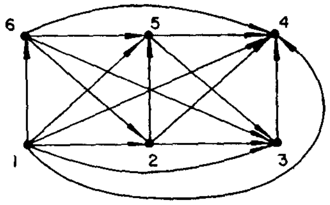

FIG. 2 Digraph G

Initially set $X \leftarrow P$. After Algorithm A1 is applied, X becomes

$$X = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

An MEG of the digraph is given in Figure 3.

B. STRONGLY CONNECTED DIGRAPHS. Before presenting an algorithm for finding an MEG of an s.c. digraph, it is first shown that Moyles and Thompson's Algorithm A2 for finding an MEG of an s.c. digraph contains some mistakes. Example 2, shown in Figure 4, illustrates a case where Moyles and Thompson's Algorithm A2 cannot find an MEG of an s.c. digraph.

A tree is constructed in Figure 5, and according to Moyles and Thompson's Algorithm A2, it is not certain which one of the following two sequences should be taken: {1-3-4-5-1, 1-2-3-6-1} or {1-3-4-5-1, 1-3-6-1-2}. In either case, all the vertices are included. But in both cases, they are not MEGs.

An MEG of the s.c. graph is given in Figure 6.

The following steps present an algorithm for finding an MEG of an s.c. digraph. An acyclic subgraph $G1$ of G is first defined as:

(1) Pick up any vertex of G which has an outgoing degree greater than or equal to the incoming degree and order it vertex 1.

(2) Order all those vertices v which are adjacent from vertex 1, i.e. $(1, v) \in G$ as vertices 2, 3, 4, ...

(3) Apply the same procedure as given in step (2) to every other vertex which has been ordered so far. If $(i, j) \in G$ and $(j, i) \notin G$ where $i > j$, then the two orders i and j can be interchanged. The idea is to make $G1$ contain as many edges of G as possible.

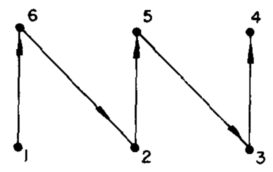

FIG. 3 MEG of digraph G

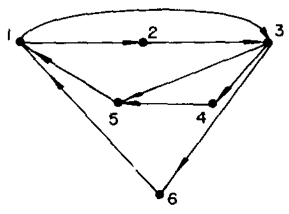

FIG. 4. Case where Moyles and Thompson's algorithm A2 cannot find an MEG of a digraph

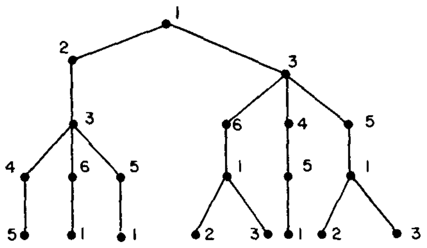

FIG. 5. Tree

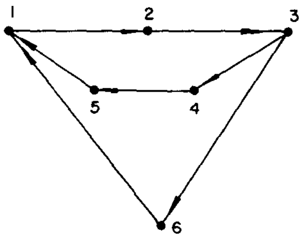

FIG. 6. MEG of G

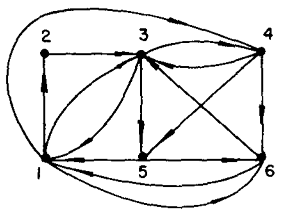

FIG 7 G

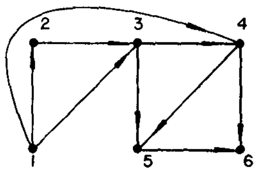

FIG. 8 $G1$

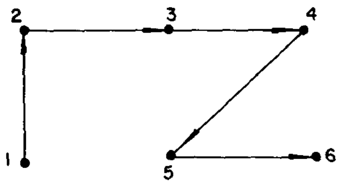

FIG 9 $G2$

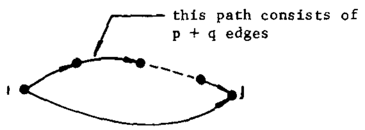

FIG. 10. Superfluous edge (i, j) and path between i and j

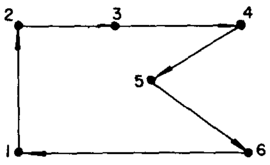

FIG 11. MEG of $G3$

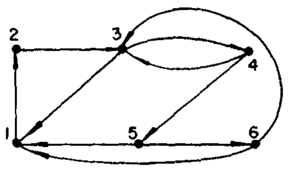

FIG. 12. $G3 = G2 \cup G - G1$

It can easily be seen that $G1$ contains all the vertices of G and that $G1$ is a subgraph of G such that for all $(i, j) \in G1$, $i < j$. It can easily be seen that $G1$ is acyclic.

Figure 8 shows an acyclic digraph $G1$ which is obtained from an s.c. digraph G given in Figure 7. Define $G2$ to be an MEG of $G1$ which is obtained by removing all the superfluous edges of $G1$. $G2$ can be obtained by applying Algorithm A1. Figure 9 gives an example of $G2$ which is an MEG of the digraph $G1$ given in Figure 8.

In order to apply Algorithm A2, it is assumed that (i, j) is a superfluous edge and that between vertices i and j there is a path which consists of p number of edges from $G - G1$ and q number of edges from $G2$. Figure 10 shows the superfluous edge (i, j) as well as the path between i and j.

ALGORITHM A2. Finding an MEG of an S C. Graph

```
G3 is defined to be $G2 \cup G - G1$
(SC1) Initially set $p = 1$ and $q = 1$, where $p$ and $q$ are defined as shown in Figure 10
(SC2) Use only $p$ number of edges from $G - G1$ and $q$ number of edges from $G2$; try to remove as
      many superfluous edges of $G3$ as possible.
After SC2 is exhausted, go to SC3.
(SC3) (Update $p$) set $p = p + 1$
      (a) If $p \leq |G - G1|$, where $|G - G1|$ is the number of edges in $G - G1$, go to SC2.
      (b) If $p > |G - G1|$, go to SC4
(SC4) (Update $q$) set $q = q + 1$.
      (a) If $q \leq |G2|$, set $p = 1$ and go to SC2.
      (b) Otherwise go to SC5.
(SC5) Terminate the algorithm The resulting digraph $G4$ is an MEG of $G$.
```

Figure 11 shows an MEG of the s.c. digraph which is given in Figure 7. $G4$ is obtained by applying algorithm A2 to the digraph $G3 = G2 \cup G - G1$. $G3$ is shown in Figure 12.

C. AN MEG OF A DIGRAPH. Now Algorithms A1 and A2 shall be used to find an MEG of a digraph. In the first place, all the s.c. components shall be found.

Algorithm A3 presents an algorithm to partition V into l subsets, i.e. V_i, $1 \leq i \leq l$. V_i is the set of all the vertices in the ith s.c. component. Once the partition has been accomplished, the set of vertices in V are going to be reordered such that all the vertices in each V_i are to be ordered by consecutive integers. As a result, the columns and rows of the adjacent matrix X are also to be permuted accordingly. The adjacency matrix X is therefore represented as:

$$X = \begin{bmatrix} X_{11}, & X_{12}, & \cdots, & X_{1l} \\ X_{21}, & X_{22}, & \cdots, & X_{2l} \\ \vdots & \vdots & & \\ X_{l1}, & X_{l2}, & \cdots, & X_{ll} \end{bmatrix},$$

where each X_{ij} is a submatrix of order $n_i \times n_j$.

ALGORITHM A3 Finding All the S.C. Components of a Digraph

For convenience of representation, we let $X(i, j)$ represent x_{ij} of matrix X and $P(i, j)$ represent p_{ij} of matrix P $S(1)$ represents the smallest element in the set S

```
(PT1) Initially $S = \{1, 2, 3, \cdots, N\}$
      $I \leftarrow 1$, ROOT $\leftarrow 1$, $PR \leftarrow 1$, $J \leftarrow$ ROOT $+ 1$, $V_I \leftarrow$ ROOT.
(PT2) (a) If $J \in S$, $X(PR, J) = 1$, and $P(J, \text{ROOT}) = 1$, then $V_I \leftarrow \{J\} \cup V_I$ and $S \leftarrow S - \{J\}$.
          Set $PR \leftarrow J$, go to PT3
      (b) Otherwise go to PT3
(PT3) (Update $J$) $J \leftarrow J + 1$
      If $J \leq N$, go to PT2
      If $J > N$, go to PT4
(PT4) (a) If $S \neq \varnothing$, then ROOT $\leftarrow S(1)$.
          $I \leftarrow I + 1$, $J \leftarrow$ ROOT $+ 1$, $V_I \leftarrow$ ROOT.
          $PR \leftarrow$ ROOT
          go to PT2.
      (b) If $S = \varnothing$, go to PT5
(PT5) Print out the contents of $V_I$, $1 \leq I \leq l$
```

After the set of vertices in each s.c. component has been found, the set of vertices V is to be reordered such that in each s.c. component, the set of vertices V_i, $1 \leq i \leq l$, is to be ordered by consecutive integers. The columns and rows of X are to be permuted accordingly.

Algorithm A2 is then applied to each s.c. component with adjacency matrix X_{ii}, where $1 \leq i \leq l$. An MEG for each s.c. component with new X_{ii} is therefore obtained.

The final step to be carried out is to remove all the superfluous edges and all the parallel edges except one in X_{ij}, where $i \neq j$. This can be easily implemented by using Algorithm A4.

ALGORITHM A4 Removing the Superfluous and the Parallel Edges

```
(1) Construct the condensed digraph with adjacency matrix $Y$. Apply Algorithm A1 to the condensed
    digraph to remove all the superfluous edges
(2) Set $X_{ij}$ to 0, where 0 is a zero matrix of order $n_i \times n_j$ if the corresponding $y_{ij}$ is zero.
(3) In each $X_{ij}$, where $y_{ij} = 1$ and $i \neq j$, keep only one nonzero entry and reduce all the rest to zeros.
    Step (3) is to remove all the parallel edges.
(4) Reconstruct the adjacency matrix $X$, where $X_{ii}$, $1 \leq i \leq l$, is the new $X_{ii}$ which is obtained by
    applying Algorithm A2. The resulting $X$ is the adjacency matrix of an MEG of the digraph.
```

REFERENCES

1 MOYLES, D M., AND THOMPSON, G. L An algorithm for finding a minimal equivalent graph of a digraph *J ACM 16*, 3 (July 1969), 455-460.

2. WARSHALL, S. A theorem on Boolean matrices *J. ACM 9* (Jan. 1962), 11-12.

3 BERZTISS, A T *Data Structures*. Academic Press, New York, 1971.

RECEIVED APRIL 1973; REVISED JULY 1974