Computing 18, 199—208 (1977)

Computing

The Complexity of Comparability Graph Recognition and Coloring*

M. C. Golumbic, New York, N.Y.

Received July 1, 1976

Abstract — Zusammenfassung

The Complexity of Comparability Graph Recognition and Coloring. Using the notion of G-decomposition introduced in Golumbic [8, 9], we present an implementation of an algorithm which assigns a transitive orientation to a comparability graph in $O(\delta \cdot |E|)$ time and $O(|E|)$ space where δ is the maximum degree of a vertex and $|E|$ is the number of edges. A quotient operation reducing the graph in question and preserving G-decomposition and transitive orientability is shown, and efficient solutions to a number of NP-complete problems which reduce to polynomial time for comparability graphs are discussed.

Die Komplexität der Erkennung und der Färbung von transitiv orientierbaren Graphen. Wir verwenden den in Golumbic [8, 9] eingeführten Begriff der G-Zerlegung, um eine Implementierung eines Algorithmus anzugeben, der einem transitiv orientierbaren Graphen eine transitive Orientierung in Zeit $O(\delta \cdot |E|)$ und Platz $O(|E|)$ zuordnet, wobei δ der maximale Grad eines Knoten und $|E|$ die Anzahl der Kanten ist. Wir zeigen eine Quotientenoperation, die den betrachteten Graphen reduziert und G-Zerlegung und transitive Orientierbarkeit bewahrt, und es werden effiziente Lösungen einiger NP-vollständiger Probleme diskutiert, die, für transitiv orientierbare Graphen, in Polynomzeit lösbar sind.

1. Introduction

In Golumbic [8, 9] the notion of G-decomposition was presented in order to describe a new matroid associated with any undirected graph. It was shown that the length of a G-decomposition equals the rank of the matroid and that a set of representatives, called a scheme, constitutes a basis of the matroid. A new characterization of comparability graphs was given including a method for producing a transitive orientation and counting the number of such orientations.

In this paper we move from the strict algebraic setting into an algorithmic presentation of these terms. Section 4 discusses the computational complexity of recognizing comparability graphs. An implementation is shown for an algorithm which assigns a transitive orientation in $O(\delta \cdot |E|)$ time and $O(|E|)$ space where δ is the maximum degree of a vertex and $|E|$ is the number of edges. In Section 5 we present a quotient operation which reduces the graph

* This work was supported in part by NSF grant DCR-75-09218.

under consideration at each iteration of the algorithm. Section 6 deals with efficient solutions to the clique problem, minimum coloring problem, et. al. for comparability graphs. These problems are *NP*-complete for arbitrary graphs, but polynomial solutions can be found for many special classes of graphs by exploiting their particular structure. (See Gavril [4] for a similar treatment of chordal graphs.)

2. Definitions

A *graph* (V, E) consists of an anti-reflexive binary relation E over a finite set V of *vertices*. The members of E are called *arcs* or *edges* and can be thought of as ordered pairs of distinct vertices. Thus we are assuming all graphs are loop-free and have no multiple edges. We define the relations

$$[a\,b \in E^{-1} \Leftrightarrow b\,a \in E] \quad \text{and} \quad \bar{E} = E \cup E^{-1}$$

respectively. A graph is *undirected* if $E = E^{-1}$.

An undirected graph (V, E) is called a *comparability graph* if there exists a graph (V, F) such that

$$F \cap F^{-1} = \emptyset; \quad F \cup F^{-1} = E; \quad F^2 \subseteq F$$

where $F^2 = \{a\,c \mid a\,b, b\,c \in F \text{ for some vertex } b\}$. The relation F is a partial ordering of V whose comparability relation is precisely E and F is called a *transitive orientation* of E.

Let (V, E) be an undirected graph. Define the binary relation Γ on E as follows:

$$a\,b\ \Gamma\ a'\,b' \quad \text{iff} \quad \begin{array}{ll} \text{either} & a = a',\ b\,b' \notin E \\ \text{or} & b = b',\ a\,a' \notin E. \end{array}$$

The relation Γ represents a type of local forcing. Since E is anti-reflexive $a\,b\ \Gamma\ a\,b$, *however*, $a\,b\ \not\Gamma\ b\,a$. The reader should not continue until he is convinced of this fact. The reflexive, transitive closure Γ^* of Γ is an equivalence relation on E and hence partitions E into what we shall call the *implication classes* of E.

Thus edges $a\,b$ and $c\,d$ are in the same implication class if and only if there exists a *Γ-chain* of edges

$$a\,b = a_0\,b_0\ \Gamma\ a_1\,b_1\ \Gamma\ \ldots\ \Gamma\ a_k\,b_k = c\,d, \quad \text{with} \quad k \geq 0.$$

If one considers the graph (E, Γ) then the implication classes of (V, E) correspond to the connected components of (E, Γ).

3. Decomposition Algorithm

Let (V, E) be an undirected graph.

Initially let $i = 1$ and $E_1 = E$.

Step I: Pick an edge e_i from E_i.

Step II: Enumerate the implication class B_i of E_i containing e_i.

Step III: Define $E_{i+1} = E_i - \bar{B}_i$.

Step IV: If $E_{i+1} = \emptyset$, let $k = i$ and STOP;
otherwise increase i by one and go back to Step I.

We call $E = \bar{B}_1 + \bar{B}_2 + \ldots + \bar{B}_k$ the *G-decomposition* of E corresponding to the *scheme* $[e_1, e_2, \ldots, e_k]$. The symbol $+$ is used for the union of disjoint sets.

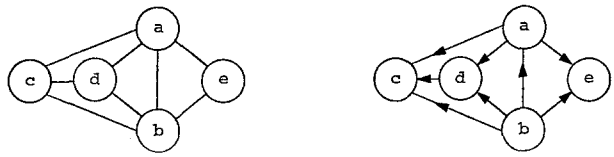

Fig. 1. An undirected graph and the transitive orientation generated by the scheme [*a c, b c, d c*]

Example: The graph in Fig. 1 has implication classes:

$A_1 = \{a\,b\}$ $\quad A_1^{-1} = \{b\,a\}$
$A_2 = \{c\,d\}$ $\quad A_2^{-1} = \{d\,c\}$
$A_3 = \{a\,c, a\,d, a\,e\}$ $\quad A_3^{-1} = \{c\,a, d\,a, e\,a\}$
$A_4 = \{b\,c, b\,d, b\,e\}$ $\quad A_4^{-1} = \{c\,b, d\,b, e\,b\}$

The scheme $[a\,c, b\,c, d\,c]$ yields a G-decomposition in which $B_1 = A_3$, $B_2 = A_4 + A_1^{-1}$ and $B_3 = A_2^{-1}$.

In Golumbic [9] we prove the following:

Theorem 1 (*TRO Theorem*): *Let* (V, E) *be an undirected graph with G-decomposition* $E = \bar{B}_1 + \bar{B}_2 + \ldots + \bar{B}_k$.

The following statements are equivalent:

i) (V, E) *is a comparability graph,*

ii) $A \cap A^{-1} = \emptyset$ *for all implication classes* A *of* E.

iii) $B_i \cap B_i^{-1} = \emptyset$ *for* $i = 1, 2, \ldots, k$.

Furthermore, when these conditions hold, $B_1 + B_2 + \ldots + B_k$ *is a transitive orientation of* E.

Our algorithm is a modification of that found in [10]. The major improvement comes from the observation that under our definition of Γ, anti-symmetry implies transitivity for implication classes and unions thereof. (See Golumbic [9], Theorem 2.) This will allow us to reduce the time complexity when $|E| \ll |V|^2$. The reader may verify that an implication class A is either disjoint from or identically equal to its reversal A^{-1}.

4. Complexity of the Decomposition Algorithm

A version of the decomposition algorithm in a pseudo-computer language is presented here to suggest how we may enumerate an implication class. This allows us to show that one can find a G-decomposition and test for transitive orientability in $O(\delta \cdot |E|)$ time and $O(|E|)$ space where δ is the maximum degree of a vertex.

Let (V, E) be an undirected graph with vertices $v_1, v_2, \ldots, v_n$. In the algorithm below we use the function

$$\text{CLASS}(i,j) = \begin{cases} 0 & \text{if } v_i v_j \notin E \\ k & \text{if } v_i v_j \text{ has been assigned to } B_k \\ -k & \text{if } v_i v_j \text{ has been assigned to } B_k^{-1} \\ \text{undefined} & \text{if } v_i v_j \in E \text{ has not yet been assigned} \end{cases}$$

and $|\text{CLASS}(i,j)|$ denotes the absolute value of CLASS (i,j).

The set E is always assumed to be a collection of ordered pairs and the *degree* d_i of vertex v_i is taken to mean the number of edges with v_i as first coordinate (often called the *out-degree*). We freely use the identity

$$|E| = \sum_{i=1}^{n} d_i$$

in our analysis.

Decomposition Algorithm (alternate version)

Input: Adjacency sets where $j \in \text{ADJ}(i)$ if and only if $v_i v_j \in E$.

Output: A G-decomposition of the graph given by the final values of CLASS and a variable FLAG which is zero if the graph is a comparability graph and one otherwise.

Method: In the k-th-iteration an unexplored edge is placed in B_k, (its CLASS is changed to k). Every edge placed into B_k is explored by adding to B_k those edges Γ-related to it in the graph E_k. (Notice that $v_i v_j \in E_k$ if and only if either $|\text{CLASS}(i,j)|$ equals k or is undefined throughout the k-th-iteration.)

The variable FLAG is changed from 0 to 1 the first time a B_k is found such that $B_k \cap B_k^{-1} \neq \emptyset$. At that point it is known that (V, E) is not a comparability graph by Theorem 1.

The algorithm is as follows:

```
begin
    initialize: k←0; FLAG←0;
    for each edge v_i v_j in E do
        if CLASS (i, j) is undefined then
            begin
                k←k+1;
                CLASS (i, j)←k; CLASS (j, i)← −k;
                EXPLORE (i, j);
            end;
```

```
    end
end
procedure EXPLORE (i, j):
    for each m ∈ ADJ (i) such that [m ∉ ADJ (j) or |CLASS (j, m)| < k] do
        if CLASS (i, m) is undefined then
            begin
                CLASS (i, m) ← k; CLASS (m, i) ← −k;
                EXPLORE (i, m);
            end
        else
            if CLASS (i, m) = −k then
                begin
                    CLASS (i, m) ← k; FLAG ← 1;
                    EXPLORE (i, m);
                end;
    end
    for each m ∈ ADJ (j) such that [m ∉ ADJ (i) or |CLASS (i, m)| < k] do
        if CLASS (m, j) is undefined then
            begin
                CLASS (m, j) ← k; CLASS (j, m) ← −k;
                EXPLORE (m, j);
            end
        else
            if CLASS (m, j) = −k then
                begin
                    CLASS (m, j) ← k; FLAG ← 1;
                    EXPLORE (m, j);
                end;
    end
return
```

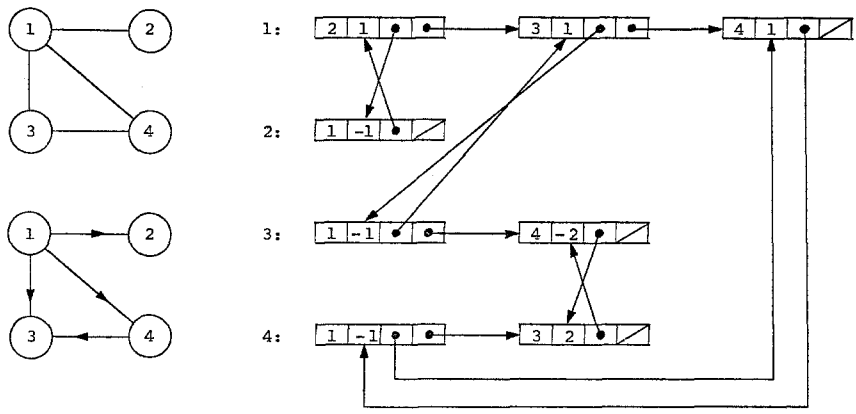

Fig. 2. An undirected graph, the transitive orientation generated by the scheme [(1, 2), (4, 3)] and its data structure after running the algorithm

Complexity analysis: The adjacency sets are stored as linked lists sorted into increasing order. The element of the list ADJ (i) represents edge $v_i v_j$ will have 4 fields containing respectively, j, CLASS (i,j), pointer to CLASS (j,i) and pointer to next element on ADJ (i) [see Fig. 2]. The storage requirement for this data structure is $O(|E|)$, and if sorting the lists is done in

$$O\left(\sum_{i=1}^{n} d_i \log d_i\right),$$

then the entire initialization of the data structure can be accomplished in $O(|E| \cdot \log \delta)$ time.

The crucial factor in the analysis of our algorithm is the time required to access or assign the CLASS function. Ordinarily finding CLASS (i, m) could take $O(d_i)$ steps by scanning ADJ (i), but if a temporary pointer happened to be in the "neighborhood", then a reference to CLASS (i, m) or CLASS (m, i) would take a fixed number of steps. Consider the first loop of EXPLORE (i, j). Two temporary pointers simultaneously scan ADJ (i) and ADJ (j) looking for values of m which satisfy the condition in the *for* statement. Since the lists are sorted and thanks to these neighborly pointers, this loop can be executed in $O(d_i + d_j)$ steps. The second loop is done similarly, hence the time complexity of EXPLORE (i,j) is $O(d_i + d_j)$.

In the main program, a temporary pointer scans each adjacency list successively in the *for* loop implying a time complexity of $O(|E|)$. Finally, the algorithm calls EXPLORE once for each edge or its reversal (both if their implication classes are not disjoint). Therefore, since

$$\sum_{v_i v_j \in E} (d_i + d_j) = 2 \sum_{i=1}^{n} d_i^2 \leq 2\delta \sum_{i=1}^{n} d_i = 2\delta |E|$$

it follows that the time complexity for the entire algorithm (including preprocessing the input) is at most $O(\delta \cdot |E|)$. Thus we have proved the following:

Theorem 2: *Comparability graph recognition and finding a transitive orientation can be done in $O(\delta \cdot |E|)$ time and $O(|E|)$ space where δ is the maximum degree of a vertex.*

The algorithm as presented in this section explores the edges in a depth first search. Replacing each recursive call EXPLORE (x, y) by placing $x\,y$ in a queue of edges to be explored would change the algorithm to breadth first search. Some future application may lead us to prefer one over the other.

5. A Quotient Theorem for G-Decompositions of a Graph

In this section we will present a quotient operation reducing the graph under consideration at each iteration of the decomposition algorithm by merging certain vertices and their adjacent edges. This quotient operation preserves schemes, G-decompositions, transitive orientability and the basic structure of the comparability matroid.

Define the following relation on the vertices of a graph (V, E):

$$a \sim a' \text{ if and only if for all } b \in V,$$

$$[a\,b \in E \Leftrightarrow a'\,b \in E] \quad \text{and} \quad [b\,a \in E \Leftrightarrow b\,a' \in E].$$

By the anti-reflexivity of E, we have in addition

$$a \sim a' \text{ implies } a\,a', a'\,a \notin E.$$

If $a \sim a'$, then we say that a and a' *can be merged.* In terms of the adjacency matrix of E, two vertices are $\sim$-related iff their corresponding columns are equal as well as their corresponding rows. The relation $\sim$ is easily seen to be an equivalence on V. Thus non-adjacent vertices having identical adjacencies with the other vertices of V are equivalent under $\sim$ and can be merged.

Next we define $(\tilde{V}, \tilde{E})$, the *quotient graph* of (V, E). Let $\tilde{V}$ be the set of all equivalence classes under $\sim$ and $\tilde{a}$ represent the $\sim$-class containing $a \in V$. For any subset $A \subset E$ we define

$$\tilde{A} = \{\tilde{a}\,\tilde{b} \mid a\,b \in A\}. \tag{1}$$

Notice that $\tilde{c}\,\tilde{d} \in \tilde{A}$ does *not* in general imply that $c\,d \in A$, however, it is true that

$$\tilde{a}\,\tilde{b} \in \tilde{E} \Leftrightarrow a\,b \in E.$$

The reader may easily show the following:

Proposition 3: *Let $\tilde{a}\,\tilde{b}, \tilde{c}\,\tilde{d} \in \tilde{E}$. Conditions i), ii), and iii) are equivalent and imply iv), but not conversely.*

i) $\tilde{a}\,\tilde{b} = \tilde{c}\,\tilde{d}$

ii) $a \sim c$ *and* $b \sim d$

iii) $\tilde{a} = \tilde{c}$ *and* $\tilde{b} = \tilde{d}$

iv) $a\,b\,\Gamma^*\,c\,d$.

The next theorem and corollary show that in passing from a graph to its quotient, we retain a one-to-one correspondence between their respective implication classes, G-decompositions, and transitive orientations.

Theorem 4: *Let (V, E) be an undirected graph and $(\tilde{V}, \tilde{E})$ its quotient graph.*

i) If A is an implication class of E, then $\tilde{A}$ is an implication class of $\tilde{E}$.

ii) If $[e_1, \ldots, e_k]$ is a scheme for E with corresponding G-decomposition $E = \bar{B}_1 + \bar{B}_2 + \ldots + \bar{B}_k$, then $[\tilde{e}_1, \ldots, \tilde{e}_k]$ is a scheme for $\tilde{E}$ with corresponding G-decomposition $\tilde{E} = \bar{\tilde{B}}_1 + \bar{\tilde{B}}_2 + \ldots + \bar{\tilde{B}}_k$.

iii) If (V, F) is a transitive orientation of (V, E), then $(\tilde{V}, \tilde{F})$ is a transitive orientation of $(\tilde{V}, \tilde{E})$.

Furthermore, every implication class, scheme, G-decomposition and transitive orientation of $\tilde{E}$ is of this form.

Corollary 5: *The number of implication classes, the length of a scheme or G-decomposition and the number of transitive orientations are all invariant under the quotient operation.*

Proof: Statement i) follows from ii) by choosing a G-decomposition of E beginning with $B_1 = A$.

(ii) Let $\bar{B}_1 + \bar{B}_2 + \ldots + \bar{B}_k$ be a G-decomposition of E. We claim that

$$a\,b \in B_i \Leftrightarrow \tilde{a}\,\tilde{b} \in \tilde{B}_i. \tag{2}$$

If $\tilde{a}\,\tilde{b} \in \tilde{B}_i$, then $\tilde{a}\,\tilde{b} = \tilde{c}\,\tilde{d}$ for some $c\,d \in B_i$. Hence, $a\,b\ \Gamma^*\ c\,d$ by Proposition 3 and thus $a\,b \in B_i$. The opposite implication follows from (1) proving (2). Therefore, $\tilde{E} = \bar{\tilde{B}}_1 + \bar{\tilde{B}}_2 + \ldots + \bar{\tilde{B}}_k$.

It remains to be shown that $\tilde{B}_i$ is an implication class of $G_i = \bar{\tilde{B}}_i + \ldots + \bar{\tilde{B}}_k$. This follows from the claim that

$$\tilde{a}\,\tilde{b}\ \Gamma^*_{G_i}\ \tilde{c}\,\tilde{d} \Leftrightarrow a\,b\ \Gamma^*_{E_i}\ c\,d \tag{3}$$

where $E_i = \bar{B}_i + \ldots + \bar{B}_k$.

We have the implications

$$\begin{aligned}\tilde{a}\,\tilde{b}\ \Gamma_{G_i}\ \tilde{a}'\,\tilde{b}' &\Rightarrow \tilde{a}\,\tilde{b}\ \Gamma_{G_i}\ \tilde{a}'\,\tilde{b}\ \Gamma_{G_i}\ \tilde{a}'\,\tilde{b}' \text{ and } \tilde{a}\,\tilde{a}', \tilde{b}\,\tilde{b}' \notin G_i \\ &\Rightarrow \text{by (2)}, a\,a', b\,b' \notin E_i \\ &\Rightarrow a\,b\ \Gamma_{E_i}\ a'\,b\ \Gamma_{E_i}\ a'\,b'.\end{aligned}$$

Conversely, $a\,b\ \Gamma_{E_i}\ a'\,b'$ implies $\tilde{a}\,\tilde{b}\ \Gamma_{G_i}\ \tilde{a}'\,\tilde{b}'$ using (2). Therefore, claim (3) is proved.

Finally, if $\bar{A}_1 + \ldots + \bar{A}_k$ is another G-decomposition of E such that $\tilde{A}_i = \tilde{B}_i$ for each i, then by (2) $A_i = B_i$. On the other hand, if $\bar{D}_1 + \ldots + \bar{D}_k$ is a G-decomposition of $\tilde{E}$, then it is easily shown that $\bar{C}_1 + \ldots + \bar{C}_k$ is a G-decomposition of E where $C_i = \{a\,b \in E \mid \tilde{a}\,\tilde{b} \in D_i\}$ and that $\tilde{C}_i = D_i$. Thus the correspondence between G-decompositions of E and $\tilde{E}$ is one-to-one.

Obviously, $[e_1, \ldots, e_k]$ is a scheme for E if and only if $[\tilde{e}_1, \ldots, \tilde{e}_k]$ is a scheme for $\tilde{E}$.

(iii) Every transitive orientation of a graph must arise from a G-decomposition of the graph. Hence, $a\,b \in F \Leftrightarrow \tilde{a}\,\tilde{b} \in \tilde{F}$ (by (2)). From this the following implications hold:

$$F \cap F^{-1} = \emptyset \Rightarrow \tilde{F} \cap \tilde{F}^{-1} = \emptyset$$

$$F + F^{-1} = E \Rightarrow \tilde{F} + \tilde{F}^{-1} = \tilde{E},$$

and

$$\tilde{a}\,\tilde{b}, \tilde{b}\,\tilde{c} \in \tilde{F} \Rightarrow a\,b, b\,c \in F \Rightarrow a\,c \in F \Rightarrow \tilde{a}\,\tilde{c} \in \tilde{F}.$$

This concludes the proof of the theorem.

Theorem 4 allows one to replace E_i by its quotient at every iteration of the decomposition algorithm without affecting the number of transitive orientations or the basic structure of the comparability matroid.

It was discovered in Golumbic [9] that the length of any G-decomposition or scheme of a graph (V, E) is an invariant which we denoted by $g(E)$. Theorem 4 leads us to the following:

Corollary 6: *Let K_n be the complete graph on n vertices. Then $g(K_n)=n-1$.*

Proof: Every edge of K_n is its own implication class. Choose an edge $a\,b$ and let $B_1=\{a\,b\}$. In $K_n-\bar{B}_1$ we have $a\sim b$, and the quotient of $K_n-\bar{B}_1$ is precisely K_{n-1}. By induction, $g(K_{n-1})=n-1$ so $g(K_n)=n$. The case $n=1$ is trivial.

The proof of Corollary 6 reveals an interesting property. For distinct vertices x and y, if $\{x\}\cup \mathrm{ADJ}(x)=\{y\}\cup \mathrm{ADJ}(y)$ in E, then $A=\{x\,y\}$ is an implication class of E and $x\sim y$ in $E-\bar{A}$.

6. Cliques and Colorings of Comparability Graphs

Any *acyclic* orientation (V, F) of an undirected graph gives a natural partial ordering of V where $x>y$ if there exists a path in F from x to y. A *height* function is then induced on the vertices: $h(v)=0$ if v is a sink; otherwise, $h(v)=1+\mathrm{MAX}\{h(w)\mid v\,w\in F\}$. It is an easy exercise to show that the height can be assigned in $O(|F|)$ by a recursive depth first search. The function h is a valid vertex coloring (adjacent vertices have different colors), but is not necessarily a minimum coloring. The situation is much better if F is also transitive.

It is well known that for *comparability graphs* the size of the largest clique and the chromatic number are equal (see [1]). Furthermore, the height function of a transitive orientation F is a minimum vertex coloring, and the maximal paths of F correspond precisely to the maximal cliques. Therefore, we can summarize their computational complexity.

Theorem 7: *Given a transitive orientation of a comparability graph (V, E), finding a minimum coloring of the vertices and a maximum clique can be done in $O(|E|)$ time.*

Corollary 8: *The clique problem and minimum vertex coloring problem can be solved in $O(\delta\cdot|E|)$ time and $O(|E|)$ space for comparability graphs.*

The more general weighted clique problem, where each vertex is assigned a positive integer weight and a clique whose vertices have largest total weight is to be found, is similarly solved for comparability graphs as are many other normally *NP*-complete problems.

We conclude with an interesting polynomial-time method for finding the size of the largest independent set of vertices of a comparability graph. A set of vertices is *independent* if no pair of them is connected by an edge. We transform a transitive orientation (V, F) into a transportation network by adding two new vertices s and t and edges $s\,x$ and $y\,t$ for each source x and sink y of F. Assigning a lower capacity of one to each vertex, we initialize a compatible integer-valued flow and call a minimum-flow algorithm. The value of the minimum flow will equal the size of the smallest covering of the vertices by cliques which in turn will equal the size of the largest independent set in the case of comparability graphs (see [1]). Such a minimum flow algorithm can certainly run in $O(|V|\cdot|E|)$ steps although this is not optimal in general.

References

[1] Berge, C.: Graphs and Hypergraphs, Chapter 16. Amsterdam: North-Holland 1973.
[2] Even, S., Pnueli, A., Lempel, A.: Permutation graphs and transitive graphs. J. ACM *19*, 400—410 (1972).
[3] Gallai, T.: Transitiv orientierbare Graphen. Acta Math. Acad. Sci. Hung. *18*, 25—66 (1967).
[4] Gavril, F.: Algorithms for minimum coloring, maximum clique, minimum covering by cliques and maximum independent set of a chordal graph. SIAM J. Comp. *1*, 180—187 (1972).
[5] Ghouila-Houri, A.: Caractérisation des graphes nonorientés dont on peut orienter les arêtes de manière à obtenir le graphe d'une relation d'ordre. C. R. Acad. Sci. Paris *254*, 1370—1371 (1962).
[6] Gilmore, P. C., Hoffman, A. J.: A characterization of comparability graphs and of interval graphs. Can. J. Math. *16*, 539—548 (1964).
[7] Golumbic, M. C.: An infinite class of superperfect noncomparability graphs. IBM Research RC 5064 (October 1974).
[8] Golumbic, M. C.: Comparability graphs and a new matroid (extended abstract). Proc. Alg. Aspects of Comb., Univ. of Toronto, January 1975.
[9] Golumbic, M. C.: Comparability graphs and a new matroid. J. Comb. Th., Series *B 22*, 68—90 (1977).
[10] Pnueli, A., Lempel, A., Even, S.: Transitive orientation of graphs and identification of permutation graphs. Can. J. Math. *23*, 160—175 (1971).
[11] Shevrin, L. N., Filippov, N. D.: Partially ordered sets and their comparability graphs. Siberian Math. J. *11*, 497—509 (1970).

Martin C. Golumbic
Assistant Professor of Computer Science
Courant Institute of Mathematics
251 Mercer Street
New York, NY 10012, U.S.A.