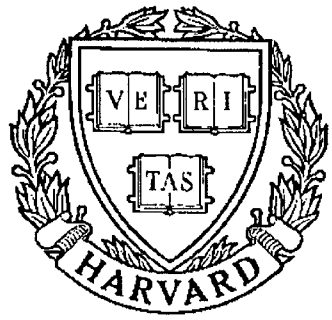
TECHNICAL RESEARCH REPORT

SYSTEMS RESEARCH CENTER



Supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), Industry and the University

Generation of Maximum Independent Sets of a Biparite Graph and Maximum Cliques of a Circular-Arc Graph

by T. Kashiwabara, S. Masuda, K. Nakajima, and T. Fujisawa

TR 89-65

Generation of Maximum Independent Sets of a Bipartite Graph and Maximum Cliques of a Circular-Arc Graph*

by

Toshinobu KASHIWABARA † and Sumio MASUDA

Department of Information and Computer Sciences
Osaka University, Toyonaka, Osaka 560, Japan

Kazuo NAKAJIMA

Electrical Engineering Department,
Institute for Advanced Computer Studies
and Systems Research Center
University of Maryland, College Park, Maryland 20742

and

Toshio FUJISAWA ‡

---

* This work was supported in part by the National Science Foundation Grants MIP-84-51510 and CDR-88-03012 (Engineering Research Centers Program) and in part by a grant from AT&T.

† T. Kashiwabara was on leave from Osaka University, with the Electrical Engineering Department and Institute for Advanced Computer Studies, The University of Maryland, College Park, Maryland 20742.

‡ T. Fujisawa was with the Department of Information and Computer Sciences, Osaka University, Toyonaka, Osaka 560, Japan. He deceased on December 15, 1988.

Generation of Maximum Independent Sets of a Bipartite Graph and Maximum Cliques of a Circular-Arc Graph

by

Toshinobu Kashiwabara, Sumio Masuda,

Kazuo Nakajima and Toshio Fujisawa

ABSTRACT

We present an efficient algorithm for generating all maximum independent sets of a bipartite graph. Its time complexity is $O(n^{2.5}+(\textit{output size}))$, where n is the number of vertices of a given graph. As its application, we develop an algorithm for generating all maximum cliques of a circular-arc graph. When the graph is given in the form of a family of n arcs on a circle, this algorithm runs in $O(n^{3.5}+(\textit{output size}))$ time.

1. Introduction

Among various types of combinatorial problems, such as decision problems to determine whether a given structure has specified properties and optimization problems to obtain a best possible structure based on some criteria, generation problems to report all structures that satisfy specified properties without duplication give rise to important applications. In particular, problems of generating all *maximal* or *maximum* sets of elements with specified properties for graphs have widely been investigated because of their practical significance [4,10-15]. (A set of elements is called maximal if the addition of a new element to the set destroys the properties the original elements satisfy. It is called maximum if its cardinality is the largest among all maximal sets.)

For general graphs, the best known algorithms for generating all maximal independent sets (and consequently, maximal cliques) [4,15] and all cutsets [14] require $O(n \cdot m \cdot \alpha)$ and $O((n+m)\cdot\alpha)$ time, respectively, where n, m and α are the numbers of vertices, edges and output sets, respectively. Recently, for several restricted classes of graphs, faster algorithms have been developed. For the generation of maximal independent sets, Leung [10] presented $O(n^2+\gamma)$ time algorithms for interval graphs and circular-arc graphs and an $O((n+m)\cdot\alpha)$ time algorithm for chordal graphs, where γ is the output size, that is, the total sum of sizes of the output sets. Later, Masuda *et al.* [12] developed $O(n \log n+\gamma)$ time algorithms for the first two classes. They also showed that all *maximum* independent sets can be found in $O(n \log n+\gamma)$ time for interval graphs and in $O(n^2+\gamma)$ time for circular-arc graphs. For maximum clique generation, Rotem and Urrutia [13] first presented an $O(n^2+\gamma)$ time algorithm for overlap graphs, and later Masuda *et al.* [11] developed an $O(n \log n+m+\gamma)$ time algorithm.

The time complexities of the generation algorithms for all but one of the above mentioned restricted classes of graphs are the sum of a polynomial function of the input size and a linear function of the output size. We call such generation algorithms *pseudo-linear time* algorithms. In this paper we consider a new problem which admits a pseudo-linear time generation algorithm, that is, the problem of generating all *closed ancestor sets* (defined in Section 2)

of an acyclic directed graph. As an application of this algorithm, we present two other pseudo-linear time generation algorithms. The first one produces all maximum independent sets of a bipartite graph in $O(n^{2.5}+\gamma)$ time. The second algorithm generates all maximum cliques of a circular-arc graph in $O(n^{3.5}+\gamma)$ time.

2. Definitions

We begin by the definitions of several terms on undirected graphs. Let $G = (V, E)$ be an undirected graph. If $(u, v) \in E$, we say that u and v are *adjacent* and are *incident* on the edge (u, v). If $(u, v) \notin E$, they are said to be *independent.* A set of vertices $S \subseteq V$ is called a *clique* (resp., *independent set*) of G if its vertices are pairwise adjacent (resp., independent). A clique (resp., independent set) with the maximum cardinality is called a *maximum clique* (resp., *maximum independent set,* which may be abbreviated as MIS). A set of edges $M \subseteq E$ is called a *matching* of G if no two edges in it have a common endpoint. If a vertex is incident on some edge in M, it is said to be *covered* by M; otherwise it is said to be *exposed.* A matching is called a *perfect matching* if it covers all the vertices of G. A *path* is a sequence of vertices $[v_1, v_2, \ldots, v_k]$ such that $(v_i, v_{i+1}) \in E$ for $i = 1,2,\ldots,k-1$. An *alternating path* with respect to M is a path that contains edges in M and edges not in M alternately (the words "with respect to M" may be omitted if no confusion arises). A graph (V', E') is called a *subgraph* of G if $V' \subseteq V$ and $E' \subseteq E$. A subgraph (V', E') of G is called the *induced subgraph* defined on V' and may be denoted by $G(V')$ if $E' = \{(u, v) \in E \mid u, v \in V'\}$.

We now define some terms on directed graphs (digraphs, for short). Let $\vec{G} = (V, \vec{E})$ be a digraph. Such terms as adjacent vertices, independent set, clique, matching, and induced subgraph of a digraph are defined for its undirected version by ignoring edge directions. A directed edge from vertex u to vertex v is denoted by $(u \rightarrow v)$. This edge is an *outgoing edge* of u and an *incoming edge* of v. For a vertex $v \in V$, the number of its incoming (resp., outgoing) edges is called the *indegree* (resp., *outdegree*) and are denoted by $indeg_{\vec{G}}(v)$ (resp., $outdeg_{\vec{G}}(v)$). If a vertex has indegree 0, it is called a *source.* A *directed path* from vertex u to

vertex w is a sequence of vertices $[u = v_1, v_2, \ldots, v_k = w]$ such that $(v_i \rightarrow v_{i+1}) \in \vec{E}$ for $i = 1,2,\ldots,k-1$. If $u = w$, the path is called a *directed cycle.* A digraph is said to be *strongly connected* if for any pair of vertices u and v, there is a directed path from u to v and vice versa. A *strongly connected component* of digraph $\vec{G} = (V, \vec{E})$ is a strongly connected induced subgraph $\vec{G}(U)$ such that for any $v \in V-U$, $\vec{G}(U \cup \{v\})$ is not strongly connected.

A digraph with no directed cycles is said to be *acyclic.* It is obvious that any acyclic digraph has at least one source. In an acyclic digraph $\vec{G} = (V, \vec{E})$, a vertex u is called an *ancestor* (resp., *descendent*) of vertex v if either $u = v$ or there exists a directed path from u to v (resp., from v to u). A subset S of V is called a *closed ancestor set* (abbreviated as *CA-set*) of $\vec{G}$ if, for any $v \in S$, all ancestors of v are contained in S. An example is given in Figure 1. An edge $(u \rightarrow v) \in \vec{E}$ is called a *redundant edge* of $\vec{G}$ if after its elimination, u is still an ancestor of v. A graph $\vec{H}$ is called a *transitive reduction* of $\vec{G}$ if (i) $\vec{H}$ has the same vertex set V and the same ancestor-descendant relationships as $\vec{G}$ has, and (ii) there is no graph with the same vertex set but fewer edges than $\vec{H}$ that preserves the ancestor-descendant relationships of $\vec{G}$. It is known that, for any acyclic digraph $\vec{G}$, its transitive reduction is unique and can be constructed by removing all redundant edges from $\vec{G}$ [1]. Figure 2 depicts the transitive reduction of the acyclic digraph of Figure 1.

For convenience, we may use a few additional notation. A clique of cardinality k may be called a k-clique. A set that contains an element v may be called a v-in set. Thus a clique of cardinality k that contains vertex v may be called a v-in k-clique. For a graph or digraph $G = (V, E)$ and a set of vertices $S \subseteq V$, $G-S$ denotes the induced subgraph $G(V-S)$; that is, the one obtained from G by eliminating all vertices in S and their incident edges.

3. Closed ancestor set generation

In this section, we give a pseudo-linear time generation algorithm for the CA-sets of an acyclic digraph. Let $\vec{G} = (V, \vec{E})$ with $|V| = n$ be an acyclic digraph and $\vec{G}'$ be its transitive reduction. Since $\vec{G}'$ preserves the ancestor-descendant relationships among the vertices

of V, any CA-set of $\vec{G}$ is a CA-set of $\vec{G}'$ and vise versa.

Given a subgraph $\vec{H} = (V_H, \vec{E}_H)$ of $\vec{G}'$ and a set of vertices S such that $S \subseteq V - V_H$, the procedure CAS($\vec{H}$, S) (to be given below) finds all CA-sets W of $\vec{H}$ without duplication and produces all sets $W \cup S$. Thus the execution of CAS($\vec{G}'$, $\emptyset$) yields all CA-sets of $\vec{G}'$. They are the CA-sets of $\vec{G}$ from the above observation. Our algorithm is described as follows:

Algorithm 1.

1. Let $\vec{G}' = (V, \vec{E}')$ be the transitive reduction of $\vec{G}$.
2. CAS($\vec{G}'$, $\emptyset$). □

We now explain how to construct CAS($\vec{H}$, S). If $\vec{H} = (\emptyset, \emptyset)$, that is, a null graph, the procedure ($\vec{H}$, S) simply produces S. If $\vec{H} \neq (\emptyset, \emptyset)$, it has at least one source. Let v be any source of $\vec{H}$. Among the sets that are to be produced by CAS($\vec{H}$, S), all sets $W \cup S$ that contain v in W are generated by executing CAS($\vec{H}-\{v\}$, $S \cup \{v\}$) and all sets $W \cup S$ that do not contain v are obtained by CAS($\vec{H}-\{$all descendants of $v\}$, S). Thus, we derive the following recursive procedure:

Procedure CAS($\vec{H}$, S)
begin
if $\vec{H} = (\emptyset, \emptyset)$
then output S.
else begin
$v \leftarrow$ a source of $\vec{H}$.
CAS($\vec{H}-\{v\}$, $S \cup \{v\}$).
CAS($\vec{H}-\{$all descendants of $v\}$, S).
end
end □

In the remainder of this section, we analyze the time complexity of Algorithm 1. The transitive reduction of $\vec{G}$, as well as the "transitive closure" of $\vec{G}$, can be computed by matrix

multiplication [1]. Using the algorithm in [5], we can thus obtain $\vec{G}'$ in $O(n^{2.49+})$ time. To find a source v of $\vec{H}$ when $\vec{H} \neq (\emptyset, \emptyset)$ in the procedure CAS($\vec{H}$, S), we first determine a topological order of $\vec{G}'$ before we call CAS($\vec{G}'$, $\emptyset$). This can be done in $O(n + |\vec{E}'|)$ time [2]. Then, in each execution of CAS($\cdot$, $\cdot$), a source of the graph can be found as the vertex with the least index with respect to the topological order.

In order to analyze the time complexity of the other parts of the algorithm, we need a few more definitions and lemmas. A pair of vertices is called an *incomparable pair* (abbreviates as *i-pair*) if neither one of them is an ancestor of the other. For any acyclic digraph $\vec{H} = (V_H, \vec{E}_H)$ with no redundant edges, we denote the numbers of its CA-sets, v-in CA-sets, and i-pairs, by $N_{CAS}(\vec{H})$, $N^v_{CAS}(\vec{H})$, and $N_{ip}(\vec{H})$, respectively.

Lemma 1. $|\vec{E}_H| \leq 4N_{ip}(\vec{H}) + |V_H|$.

Proof. We will assign each edge to either a vertex or an i-pair. Let $\tilde{H} = (V_H, \tilde{E})$ be a subgraph of $\vec{H}$ such that (i) $indeg_{\tilde{H}}(v) \leq 1$ and $outdeg_{\tilde{H}}(v) \leq 1$ for every $v \in V_H$, and (ii) for any $e \in \vec{E}_H - \tilde{E}$, $(V_H, \tilde{E} \cup \{e\})$ does not satisfy condition (i). Let $e = (u \rightarrow v)$ be any edge in $\vec{E}_H$. Note that $outdeg_{\tilde{H}}(u) = 1$ or $indeg_{\tilde{H}}(v) = 1$ or both equal one. If $e \in \tilde{E}$, we assign it to vertex v. Otherwise, $e \in \vec{E}_H - \tilde{E}$, and we assign it to an i-pair in the following manner:

Case 1. $outdeg_H(u) = 0$.

In this case, $indeg_{\tilde{H}}(v) = 1$. Let w be a vertex such that $(w \rightarrow v) \in \tilde{E}$. Since $\vec{H}$ has no redundant edges, $\{u, w\}$ is an i-pair. We assign e to this i-pair.

Case 2. $outdeg_H(u) = 1$.

There exists a vertex w $(\neq v)$ such that $(u \rightarrow w) \in \tilde{E}$ and $\{v, w\}$ is an i-pair. We assign e to this i-pair.

In the manner described above, every edge $e \in \vec{E}_H$ is assigned to a vertex or an i-pair. Clearly, at most one edge is assigned to every vertex. We show that at most 4 edges are assigned to every i-pair, which completes the proof. Assume that five edges e_1, e_2, e_3, e_4, and

e_5 are assigned to an i-pair $\{x, y\}$. Since they are incident on either x or y, at least two of them are incoming edges of x or y, or outgoing edges of x or y. For example, suppose that e_1 and e_2 are incoming edges of x, and let $e_1 = (u \rightarrow x)$ and $e_2 = (v \rightarrow x)$. Since $e_1 \in \vec{E}_H - \tilde{E}$ and e_1 is assigned to the i-pair $\{x, y\}$, the edge $(u \rightarrow y) \in \tilde{E}$ must exist due to Case 2. Likewise, $(v \rightarrow y) \in \tilde{E}$. Thus, $indeg_H(y) \geq 2$, which contradicts the definition of $\tilde{H}$. The other cases lead to a similar contradiction. □

Lemma 2. $N_{CAS}(\vec{H}) \geq N_{ip}(\vec{H}) + |V_H|$.

Proof. For any $v \in V_H$, $\{v$'s ancestors$\}$ is a CA-set. Similarly, for any i-pair $\{u, v\}$ of $\vec{H}$, $\{u$'s ancestors$\} \cup \{v$'s ancestors$\}$ forms a CA-set. It is clear from the definition of i-pairs that these $|V_H| + N_{ip}(\vec{H})$ sets are all distinct, and the lemma follows. □

Lemma 3. For any source v of $\vec{H}$, $|V_H| + |\vec{E}_H| \leq 4N^v_{CAS}(\vec{H})$.

Proof. If $|V_H| = 1$, the lemma trivially holds. Suppose that $|V_H| \geq 2$ and let $\vec{E}_1$ be the set of all edges of graph $\vec{H} - \{v\}$. From Lemma 1, $|\vec{E}_1| \leq 4N_{ip}(\vec{H} - \{v\}) + |V_H| - 1$. Since $|\vec{E}_H| - |\vec{E}_1| \leq |V_H| - 1$, we have $|\vec{E}_H| \leq 4N_{ip}(\vec{H} - \{v\}) + 2|V_H| - 2$. Furthermore, for any CA-set S of $\vec{H} - \{v\}$, $S \cup \{v\}$ is a CA-set of $\vec{H}$, and hence $N^v_{CAS}(\vec{H}) \geq N_{CAS}(\vec{H} - \{v\})$. Therefore, by Lemma 2, $4N^v_{CAS}(\vec{H}) \geq 4N_{ip}(\vec{H} - \{v\}) + 4|V_H| - 4 \geq |\vec{E}_H| + 2|V_H| - 2 \geq |V_H| + |\vec{E}_H|$. □

Theorem 1. All CA-sets of a given acyclic digraph $\vec{G}$ can be generated in $O(n^{2.49+} + \gamma)$ time, where γ is the output size.

Proof. As mentioned earlier, the construction of $\vec{G}'$ and the determination of a topological order can be done in $O(n^{2.49+})$ and $O(n + |\vec{E}|)$ time, respectively. For a subgraph $\vec{H} = (V_H, \vec{E}_H)$ of $\vec{G}'$ and $S \subseteq V - V_H$, the execution of CAS($\vec{H}$, S) i) finds a source of $\vec{H}$, say $s(\vec{H})$, ii) adds $s(\vec{H})$ to S and computes $\vec{H} - \{s(\vec{H})\}$, and iii) computes $\vec{H} - \{$descendants of $s(\vec{H})\}$. All of these can clearly be performed in $O(1 + |V_H| + |\vec{E}_H|)$ time, which may be bounded by $O(N^{s(\vec{H})}_{CAS}(\vec{H}))$ time due to Lemma 3. Therefore, the overall time complexity of the

algorithm is $O(n^{2.49+}+\sum_{\vec{H},S} N_{CAS}^{s(\vec{H})}(\vec{H}))$, where the summation is taken for all $\vec{H}$ and S computed in the algorithm. The output family of CAS($\vec{H}$, S), $\{W \cup S \mid W$ is a CA-set of $\vec{H}\}$, has $N_{CAS}^{s(\vec{H})}(\vec{H})$ sets that contain $s(\vec{H})$. This implies that $\sum_{\vec{H},S} N_{CAS}^{s(\vec{H})}(\vec{H}) = O(\gamma)$, which completes the proof. □

Remark. If $\vec{G}$ has no redundant edges, its CA-sets can be generated in $O(\gamma)$ time since $n + |\vec{E}| = O(\gamma)$ by Lemma 3. □

4. Generation of maximum independent sets of a bipartite graph

A graph (V, E) is said to be *bipartite* if V can be partitioned into two disjoint independent sets X and Y. Such a partition (X, Y) is called a *bipartition.* To specify the bipartition, the bipartite graph may be denoted by (X, Y, E).

In this section, we consider the problem of generating all MIS's of a bipartite graph $B = (X, Y, E)$. We first reduce this problem to that of finding all MIS's of a bipartite graph that has a perfect matching. We then show that the latter problem can further be reduced to the generation problem of all CA-sets of an acyclic digraph. These reductions will lead to a pseudo-linear time MIS generation algorithm for B.

4.1 First Reduction

Let M be a maximum matching of B. Let X_0 (resp., Y_0) be the set of exposed vertices in X (resp., Y). Let L_X (resp., L_Y) be the union of X_0 (resp., Y_0) and the set of all vertices of B that lie on some alternating path from a vertex in X_0 (resp., Y_0). Due to the maximality of M, $L_X \cap L_Y = \emptyset$. Finally, we define $X' = X - L_X - L_Y$ and $Y' = Y - L_X - L_Y$. An example is shown in Figure 3. Recall that $B(U)$ is the induced subgraph of B defined on a set of vertices U. For simplicity, we denote graph $B(X' \cup Y')$ by B'. Let M' be the intersection of M and the edge set of B'. It is not difficult to see that M' is a perfect matching of B'. The

following theorem enables us to reduce the problem of generating all MIS's of B to that of generating all MIS's of B' .

Theorem 2. The family of MIS's of B is equal to the family $\{(L_X \cap X) \cup (L_Y \cap Y) \cup Z \mid Z$ is an MIS of $B'\}$. □

In order to prove this theorem, we need two lemmas. The proof of Lemma 4 is elementary and is left out.

Lemma 4. B has no edge between i) a vertex in $L_X \cap X$ and a vertex in $(L_Y \cap Y) \cup Y'$ nor ii) a vertex in $L_Y \cap Y$ and a vertex in $(L_X \cap X) \cup X'$. □

Theorem 1 in Desler and Hakimi [6] shows that both $(L_X \cap X) \cup \{v \in Y \mid v$ is not adjacent to any vertex in $L_X\}$ and $(L_Y \cap Y) \cup \{v \in X \mid v$ is not adjacent to any vertex in $L_Y\}$ are MIS's of B. Thus, by Lemma 4, we have Lemma 5.

Lemma 5. $(L_X \cap X) \cup (L_Y \cap Y) \cup X'$ and $(L_X \cap X) \cup (L_Y \cap Y) \cup Y'$ are MIS's of B. □

We are now ready to prove Theorem 2.

Proof of Theorem 2. ("$\supseteq$" part) Let Z be an MIS of B' . Since B' has a perfect matching M' , $|Z| = |X'| = |Y'|$. Furthermore, $(L_X \cap X) \cup (L_Y \cap Y) \cup Z$ is an independent set due to Lemma 4, and Z is disjoint from $(L_X \cap X) \cup (L_Y \cap Y)$. Therefore, $(L_X \cap X) \cup (L_Y \cap Y) \cup Z$ is an MIS of B by Lemma 5.

("$\subseteq$" part) Let D be an MIS of B. Applying Lemma 5 to $B(L_X)$ and $B(L_Y)$, we see that $L_X \cap X$ and $L_Y \cap Y$ are MIS's of $B(L_X)$ and $B(L_Y)$, respectively. Therefore, $|D \cap L_X| \leq |L_X \cap X|$ and $|D \cap L_Y| \leq |L_Y \cap Y|$. Since B' has a perfect matching, X' is an MIS of B' and $|D \cap (X' \cup Y')| \leq |X'|$. By Lemma 5, $|D| = |(L_X \cap X) \cup (L_Y \cap Y) \cup X'| = |L_X \cap X| + |L_Y \cap Y| + |X'|$. Thus, $|D \cap L_X| = |L_X \cap X|$, $|D \cap L_Y| = |L_Y \cap Y|$ and $|D \cap (X' \cup Y')| = |X'|$, and hence $D \cap L_X$, $D \cap L_Y$ and $D \cap (X' \cup Y')$ are MIS's of $B(L_X)$, $B(L_Y)$ and B' , respectively.

The proof will be completed by showing that $L_X \cap X$ (resp., $L_Y \cap Y$) is the only MIS of $B(L_X)$ (resp., $B(L_Y)$). We prove the case for L_X (the other case is similarly proved). $B(L_X - X_0)$ has a perfect matching that consists of edges in M. We denote it by M_X. L_X consists of $2 \cdot |L_X \cap Y|$ vertices covered by M_X and $|X_0|$ exposed vertices. Let D_X be any MIS of $B(L_X)$. Since $|D_X| = |L_X \cap X| = |X_0| + |L_X \cap Y|$, D_X contains all vertices in X_0 and exactly one endpoint of every edge in M_X. Let y be any vertex in $L_X \cap Y$ and $[x_0, y_1, x_1, y_2, x_2, \ldots, y_k, x_k, y]$ be an alternating path, where $x_0 \in X_0$, $y_1, \ldots, y_k \in L_X \cap Y$, $x_1, \ldots, x_k \in L_X \cap X$, and $(y_i, x_i) \in M_X$ for $i = 1,2,\ldots,k$. Since $x_0 \in D_X$, $y_1 \notin D_X$. Thus, $x_1 \in D_X$ and $y_2 \notin D_X$. Repeating this argument $k-1$ more times, we know that $y \notin D_X$. Therefore, D_X has no vertex in $L_X \cap Y$, and hence $D_X = L_X \cap X$. □

4.2 Second Reduction

We now show that the generation of all MIS's of B' can be reduced to that of all CA-sets of an acyclic digraph. Recall that B' has a perfect matching M'. For each edge $e = (u, v)$ such that $u \in X'$ and $v \in Y'$, we give a direction from v to u if $e \in M'$, and from u to v if $e \notin M'$. We denote the resultant digraph by $\vec{B}' = (X', Y', \vec{E}')$.

Theorem 3. Let K be the vertex set of a strongly connected component of $\vec{B}'$. For any MIS D of $\vec{B}'$, either i) $K \cap X' \subseteq D$ and $(K \cap Y') \cap D = \emptyset$, or ii) $K \cap Y' \subseteq D$ and $(K \cap X') \cap D = \emptyset$.

Proof. If $|K| = 1$, the theorem trivially holds. If $|K| > 1$, it is clear that $|K| \geq 4$ and that $|K \cap X'| \geq 2$ and $|K \cap Y'| \geq 2$. Let u be any vertex in $K \cap X'$. For any $v \in (K \cap X') - \{u\}$, there are a directed path from u to v and a directed path from v to u. When the edge directions are ignored, the path from u to v is an alternating path with respect to M' and begins with an edge not in M'. Since M' is a perfect matching, any MIS D of $\vec{B}'$ contains exactly one endpoint of each edge in M'. Therefore, by the same reason-

ing as the one used in the proof of Theorem 2, we can show that if $u \in D$, then $v \in D$ and $K \cap Y' \not\subseteq D$. Applying the same argument to the directed path from v to u, we can also show that if $u \notin D$, then $v \notin D$.

Consequently, if $u \notin D$, then $(K \cap X') \cap D = \emptyset$, and if $u \in D$, then $K \cap X' \subseteq D$ and $K \cap Y' \not\subseteq D$. A similar result can be derived for any vertex in $K \cap Y'$. The theorem immediately follows from these results. □

Theorem 3 shows that, for the vertex set K of every strongly connected component of $\vec{B}'$, $K \cap X'$ and $K \cap Y'$ each may be treated as a group. This motivates us to contract each of them into a single vertex and create an edge directed from the vertex for $K \cap Y'$ to the one for $K \cap X'$. Let $\hat{B} = (\hat{X}, \hat{Y}, \hat{E})$ be the graph obtained from $\vec{B}'$ by performing such contractions for every strongly connected component (multiple edges are replaced by single edges). Thus, the generation of all MIS's of $\vec{B}'$ is reduced to that of all MIS's of $\hat{B}$. Clearly $\hat{B}$ is acyclic. Furthermore, it has a perfect matching $\hat{M}$ that corresponds to M' and that contains all the new edges. Suppose that $\hat{X} = \{x_1, x_2, ..., x_k\}$, $\hat{Y} = \{y_1, y_2, ..., y_k\}$, and $(y_i \rightarrow x_i) \in \hat{M}$ for $i = 1,2,...,k$. Note that each edge in $\hat{E} - \hat{M}$ is directed from a vertex in $\hat{X}$ to a vertex in $\hat{Y}$.

The second reduction is completed by contracting two endpoints of every edge $(y_i \rightarrow x_i) \in \hat{M}$ into a single vertex v_i (self-loops are eliminated). Let $\tilde{G} = (\tilde{V}, \tilde{E})$ be the resultant digraph. Since $\hat{B}$ is acyclic, $\tilde{G}$ is also acyclic. An example is shown in Figure 4. We define a function from the family of all the subsets of $\tilde{V}$ to the family of sets of vertices in $\hat{B}$ by $F(S) = \{x_i \in \hat{X} \mid v_i \notin S\} \cup \{y_i \in \hat{Y} \mid v_i \in S\}$ for every $S \subseteq \tilde{V}$. It is obvious that $F(S)$ is a one-to-one function.

Theorem 4. The family of MIS's of $\hat{B}$ is equal to $\{F(S) \mid S$ is a CA-set of $\tilde{G}\}$.

Proof. ("$\supseteq$" part) Assume that there exists a CA-set S of $\tilde{G}$ such that $F(S)$ contains adjacent vertices x_i and y_j of $\hat{B}$. The edge connecting them is not a member of $\hat{M}$, and hence it is

directed from x_i to y_i. By the definition of $F(S)$, $v_i \notin S$ and $v_j \in S$. Since $(x_i \rightarrow y_j) \in \hat{E}-\hat{M}$, $(v_i \rightarrow v_j) \in \tilde{E}$. This contradicts the assumption that S is a CA-set. Therefore, $F(S)$ is an independent set of $\hat{B}$. It is clearly maximum since $|F(S)| = |\hat{X}| = |\hat{Y}|$.

("$\subseteq$" part) Let D be an MIS of $\hat{B}$ and let $S = \{v_i \in \tilde{V} \mid y_i \in D\}$. For each $(y_i \rightarrow x_i) \in \hat{M}$, D contains either x_i or y_i. Therefore, $D = F(S)$. Let $(v_j \rightarrow v_i)$ be any edge in $\tilde{E}$ such that $v_i \in S$. Since $y_i \in D$ and $(x_j \rightarrow y_i) \in \hat{E}$, the independence of D implies that $x_j \notin D$, and hence $v_j \in S$. Thus, S is a CA-set of $\tilde{G}$. □

Theorems 2, 3 and 4 lead to the following algorithm for generating all MIS's of a given bipartite graph B.

Algorithm 2.

1. Find a maximum matching M of B. Determine sets L_X, L_Y, X', and Y' with respect to M and determine graph B' with its perfect matching M'.
2. Determine graph $\vec{B}'$ and find all of its strongly connected components. Construct graph $\hat{B}$ and its perfect matching $\hat{M}$.
3. By contracting every edge in $\hat{M}$, construct graph $\tilde{G}$ from $\hat{B}$.
4. Generate all CA-sets of $\tilde{G}$ and find $\{F(S) \mid S$ is a CA-set of $\tilde{G}\}$.
5. For each set $F(S)$, replace each vertex by the original vertices of B and add to them the vertices in $(L_X \cap X) \cup (L_Y \cap Y)$. Generate the resultant vertex sets. □

Step 1 can be done in $O(n^{2.5})$ time by using the Hopcroft-Karp matching algorithm [9], where n is the number of vertices in B. Step 4 can be performed in $O(n^{2.49+}+\gamma)$ time due to Theorem 1, where γ is the output size. Steps 2, 3 and 5 use $O(n^2)$, $O(n^2)$, and $O(\gamma)$ time, respectively. Thus, we have the following theorem:

Theorem 5. All MIS's of a bipartite graph can be generated in $O(n^{2.5}+\gamma)$ time, where n is the number of vertices of the graph and γ the output size. □

5. Generation of maximum cliques of a circular-arc graph

For a family A of arcs on a circle, a graph (V, E) is called the *circular-arc graph* for A if there exists a one-to-one correspondence between V and A such that two vertices in V are adjacent if and only if the corresponding arcs in A intersect. For convenience, we use two terms "vertex" and "arc" interchangeably. Moreover, we may call a clique of the circular-arc graph for A simply a clique of A.

In this section, we present a pseudo-linear time generation algorithm for all maximum cliques of a circular-arc graph. We assume that the graph is given in the form of a family of arcs $A = \{a_1, a_2, \ldots, a_n\}$ such that their endpoints are all distinct and no single arc covers the whole circle. We assume that arc a_i does not contain any arc in $\{a_{i+1}, a_{i+2}, \ldots, a_n\}$ for $i = 1,2,\ldots,n-1$ (if necessary, the renumbering can be done by sorting the arcs in ascending order of their lengths). Let μ denote the cardinality of a maximum clique of A. For $i = 1,2,\ldots,n$, let $A_i = \{a_i, a_{i+1}, \ldots, a_n\}$ and let M_i denote the family of all a_i-in μ-cliques of A_i. Note that M_i may be empty.

It is easy to show that $M_i \cap M_j = \emptyset$ for $i \neq j$ and that the family of maximum cliques of A is $\bigcup_{i=1}^{n} M_i$. Therefore, our goal is to find $M_1, M_2, \ldots, M_n$. In the following, we show that this problem can be reduced to the generation of all MIS's of a bipartite graph. Let p_i^l be one endpoint of arc a_i and p_i^r the other endpoint. We denote by A_i^l (resp., A_i^r) the set of all arcs in A_i that contain p_i^l but not p_i^r (resp., p_i^r but not p_i^l) and by A_i^{lr} the set of all arcs in A_i that contain both p_i^l and p_i^r. Let $B_i = (A_i^l \cup A_i^r, E_i)$ with $E_i = \{(u, v) \mid u, v \in A_i^l \cup A_i^r$, arcs u and v do not intersect$\}$. An example is given in Figure 5. B_i is a bipartite graph with bipartition (A_i^l, A_i^r). Such a bipartite graph was used in [7] to obtain a maximum clique of a circular-arc graph. Let τ_i denote the cardinality of an MIS of B_i. If $\tau_i + |A_i^{lr}| + 1 < \mu$, we know that $M_i = \emptyset$.

Theorem 6. If $\tau_i + |A_i^{lr}| + 1 = \mu$, then $M_i = \{A_i^{lr} \cup \{a_i\} \cup I \mid I$ is an MIS of $B_i\}$.

Proof. ("$\supseteq$" part) By the definition of B_i, any MIS I of B_i is a clique of A_i. Since every arc in $A_i^{lr} \cup \{a_i\}$ intersects all arcs in I, $A_i^{lr} \cup \{a_i\} \cup I$ is an a_i-in μ-clique of A_i.

("$\subseteq$" part) Let C be an a_i-in μ-clique of A_i. $C - \{a_i\} - A_i^{lr}$ is a clique of $A_i^l \cup A_i^r$, and thus it is an independent set of B_i. Since $|C| = \mu$, $|C - \{a_i\} - A_i^{lr}| \geq \mu - 1 - |A_i^{lr}| = \tau_i$. Therefore, $C - \{a_i\} - A_i^{lr}$ is an MIS of B_i. □

For $i = 1,2,...,n$, we first compute τ_i by finding a maximum clique of A_i. Using the algorithm by Apostolico and Hambrusch [3], this can be done in $O(n^2 \log \log n)$ time for each i. μ can be determined as $Max\{\tau_i + 1 + |A_i^{lr}| \mid i = 1,2,...,n\}$. If $\tau_i < \mu - 1 - |A_i^{lr}|$ for some i, we do not proceed further for i. On the other hand, if $\tau_i = \mu - 1 - |A_i^{lr}|$, we find M_i in the following manner: (1) construct graph B_i, (2) generate all MIS's of B_i, and (3) add $\{a_i\} \cup A_i^{lr}$ to each of them. The execution times of the first and third steps for all i are $O(n^3)$ and $O(\gamma)$, respectively. Due to Theorem 5, the second step can be done in $O(n^{2.5} + \gamma_i)$ time for each i, where γ_i is the total size of the MIS's of B_i. Since $\sum_{\tau_i = \mu - 1 - |A_i^{lr}|} \gamma_i = O(\gamma)$, we have the following theorem:

Theorem 7. The generation of all maximum cliques of a circular-arc graph can be done in $O(n^{3.5} + \gamma)$ time, where n is the number of vertices of the graph and γ the output size. □

References

[1] A. V. Aho, M. R. Garey and J. D. Ullman, The Transitive Reduction of a Directed Graph, *SIAM J. Comput.* **1** (1972), 131-137.

[2] A. V. Aho, J. E. Hopcroft and J. D. Ullman, "The Design and Analysis of Computer Algorithms," Addison-Wesley, Reading, Mass., 1972.

[3] A. Apostolico and S. E. Hambrusch, Finding Maximum Cliques on Circular-Arc Graphs, *Info. Proc. Lett.* **26** (1987), 209-215.

[4] N. Chiba and T. Nishizeki, Arboricity and Subgraph Listing Algorithms, *SIAM J. Comput.* **14** (1985), 210-223.

[5] D. Coppersmith and S. Winograd, On the Asymptotic Complexity of Matrix Manipulation, *SIAM J. Comput.* **11** (1982), 472-492.

[6] J. F. Desler and S. L. Hakimi, On Finding a Maximum Internally Stable Set of a Graph, *in* "Proc. 4th Annual Princeton Conf. on Information Sciences and Systems," pp. 459-462, Princeton, NJ, 1970.

[7] F. Gavril, Algorithms on Circular-Arc Graphs, *Networks* **1** (1974), 357-369.

[8] M. C. Golumbic, "Algorithmic Graph Theory and Perfect Graphs," Academic Press, New York, NY, 1980.

[9] J. E. Hopcroft and R. E. Karp, An $n^{2.5}$ Algorithm for Maximum Matching in Bipartite Graphs, *SIAM J. Comput.* **2** (1973), 225-231.

[10] J. Y.-T. Leung, Fast Algorithms for Generating All Maximal Independent Sets of Interval, Circular-Arc and Chordal Graphs, *J. Algorithms* **5** (1984), 22-35.

[11] S. Masuda, K. Nakajima, T. Kashiwabara and T. Fujisawa, "Efficient Algorithms for Finding Maximum Cliques of an Overlap Graphs," *Networks*, to appear.

[12] S. Masuda, K. Nakajima, T. Kashiwabara and T. Fujisawa, "Efficient Enumeration of Maximal and Maximum Independent Sets of an Interval Graph and a Circular-Arc Graph," Technical Report UMIACS-TR-87-33, University of Maryland, College Park, MD, 1987.

[13] D. Rotem and J. Urrutia, Finding Maximum Cliques in Circle Graphs, *Networks* **11** (1981), 269-278.

[14] S. Tsukiyama, H. Ariyoshi, I. Shirakawa and H. Ozaki, An Algorithm to Enumerate All Cutsets of a Graph in Linear Time per Cutset, *J. Assoc. Comput. Mach.* **27** (1980), 619-632.

[15] S. Tsukiyama, M. Ide, H. Ariyoshi and I. Shirakawa, A New Algorithm for Generating All the Maximal Independent Sets, *SIAM J. Comput.* **6** (1977), 505-517.

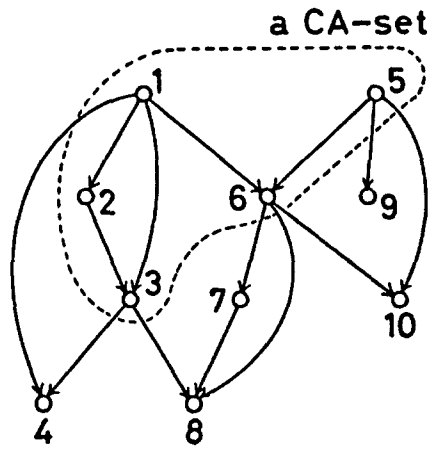


Fig. 1. An example of a CA-set.

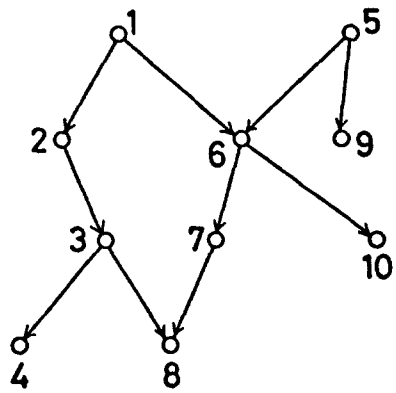


Fig. 2. The transitive reduction for the acyclic digraph of Fig. 1.

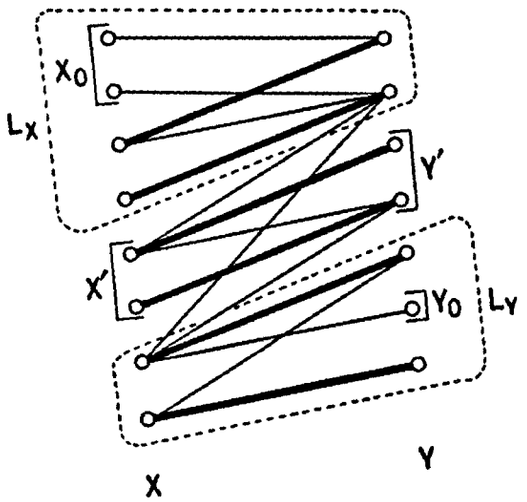


Fig. 3. Sets X_0, Y_0, L_X, L_Y, X' and Y' (the edges of M are drawn in boldface).

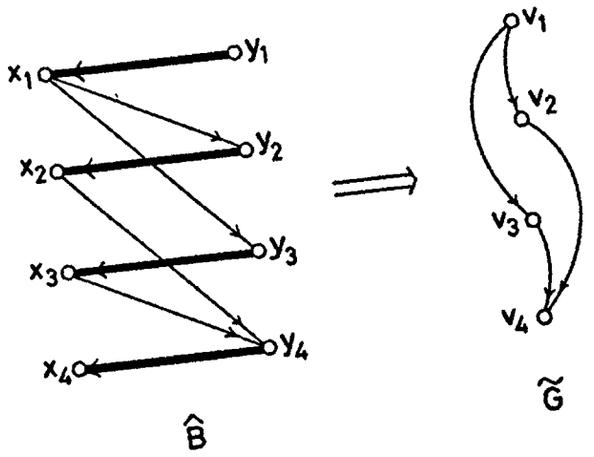


Fig. 4. Construction of graph $\tilde{G}$.

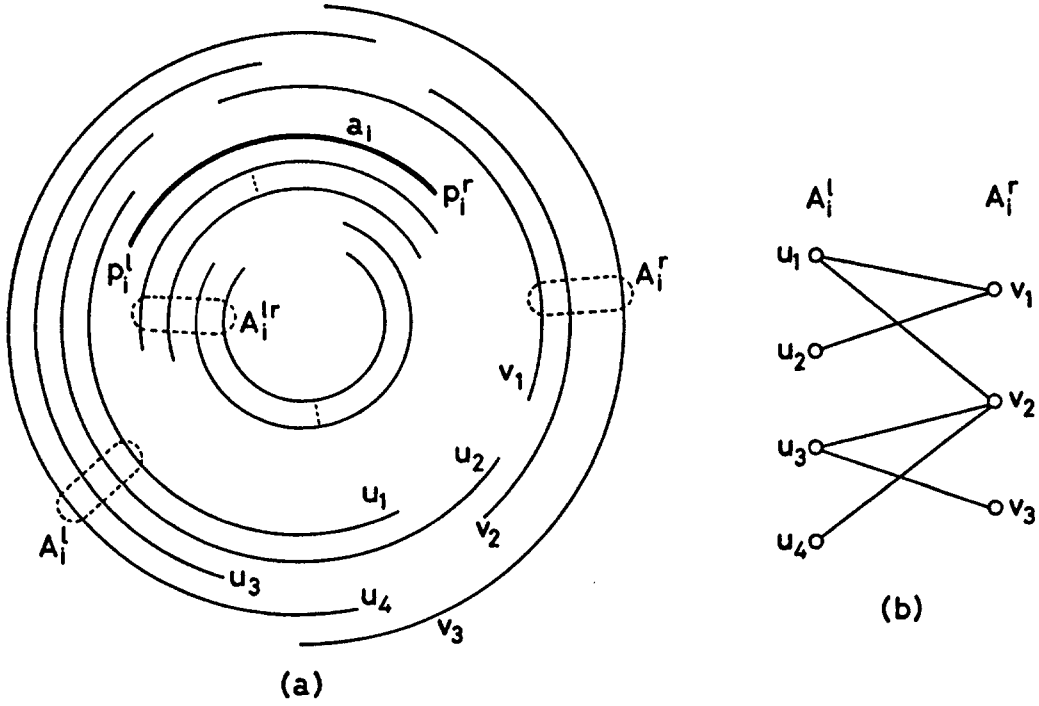


Fig. 5. (a) Sets A_i^l, A_i^r and A_i^{lr}. (b) Graph B_i.