NC Algorithms for Comparability Graphs, Interval Graphs, and Unique Perfect Matchings

Dexter Kozen[1]
Umesh V. Vazirani[2]
Vijay V. Vazirani[3]

TR 86-799

December 1986

Department of Computer Science
Cornell University
Ithaca, NY 14853

1 Supported by NSF grant DCR-8602663.
2 Supported by NSF grant MCS82-04506 and an IBM Graduate Fellowship.
3 Supported by NSF gran DCR85-03611 and an IBM Faculty Development Award.

NC Algorithms for Comparability Graphs, Interval Graphs, and Unique Perfect Matchings

Dexter Kozen[1]
Cornell University

Umesh V. Vazirani[2]
University of California, Berkeley

Vijay V. Vazirani[3]
Cornell University

Abstract. Laszlo Lovasz recently posed the following problem: "Is there an *NC* algorithm for testing if a given graph has a unique perfect matching?" We present such an algorithm for bipartite graphs. We also give *NC* algorithms for obtaining a transitive orientation of a comparability graph, and an interval representation of an interval graph. These enable us to obtain an *NC* algorithm for finding a maximum matching in an incomparability graph.

[1]Supported by NSF grant DCR-8602663.

[2]Supported by NSF grant MCS82-04506 and an IBM Graduate Fellowship.

[3]Supported by NSF grant DCR85-03611 and an IBM Faculty Development Award.

1. Introduction

Karp, Upfal and Wigderson [KUW] have recently shown that the maximum matching problem is in Random NC^3 (RNC^3). This result has since been improved to RNC^2 by Mulmuley, Vazirani, and Vazirani [MVV]. It remains open whether there is a deterministic NC algorithm for this problem. A first step might be to obtain an NC algorithm for testing if a graph has a perfect matching. An RNC algorithm for this problem exists, based on a method of Lovasz [Lo] (see [BGH]). Rabin and Vazirani [RV] give an NC algorithm for obtaining perfect matchings in graphs having a unique perfect matching. Laszlo Lovasz recently posed the following problem: "Is there an NC algorithm for testing if a given graph has a unique perfect matching?" We present such an algorithm for bipartite graphs.

We also give NC algorithms for obtaining a transitive orientation of a given comparability graph, and an interval representation for a given interval graph. The first sequential algorithms for these problems were obtained by Gilmore and Hoffman [GH] and Ghouila-Houri [Gh]. More efficient algorithms were obtained by Even, Pnueli, and Lempel [EPL,PLE] and Golumbic [Go]. The NC comparability graph algorithm, together with the NC two-processor scheduling algorithm of Helmbold and Mayr [HM] give an NC algorithm for maximum matching in incomparability graphs, a class of graphs containing all interval graphs. This suggests that the maximum matching problem may be in NC.

2. Testing for Unique Perfect Matching in Bipartite Graphs

The problem can be stated as follows:

Input: An undirected bipartite graph $G=(X,Y,E)$.

Output: A unique perfect matching if one exists.

Let $|X|=|Y|=n$. Let A be the $n \times n$ adjacency matrix for G, i.e. $A(x,y)=1$ if the edge $xy \in E$, 0 otherwise. For permutation σ on $\{1, 2, \ldots, n\}$, define

$$value(\sigma) = \prod_{i=1}^{n} A(i, \sigma(i)).$$

The permutation σ corresponds to a perfect matching in G iff $value(\sigma)=1$. The determinant of A is given by

$$|A| = \sum_{\sigma} sign(\sigma) \cdot value(\sigma).$$

Thus if G has a unique perfect matching, then $|A| = \pm 1$, and if G has no perfect matching, then $|A|=0$.

If G has a unique perfect matching $M \subseteq E$, there is a simple NC algorithm [RV] for obtaining it; however that algorithm does not check whether M is in fact unique. If the edge xy is in M, then the graph G_{xy} obtained from G by removing edge xy will have no perfect matching. If the edge xy is not in M, then G_{xy} will still have a unique perfect matching. Thus $|A_{xy}|=0$ if $xy \in M$, and $|A_{xy}|=\pm 1$ if $xy \notin M$, where A_{xy} is the adjacency matrix of G_{xy}. By computing the determinants $|A_{xy}|$ in parallel for all edges xy, M can be determined in NC.

We now give a method for verifying in NC that M is unique.

Lemma 2.1. G has a unique perfect matching iff A is nonsingular and there exist permutation matrices P, Q such that PAQ is upper triangular.

Proof. Multiplying A on the left and right by permutation matrices P and Q amounts to permuting the sets X and Y. If A is nonsingular, then there must exist a perfect matching M, since $|A| \neq 0$. If in addition there exist permutation matrices P, Q such that PAQ is upper triangular, then M is unique and lies along the diagonal of PAQ, since there are no other permutations σ with $value(\sigma) \neq 0$.

Conversely, if there is a unique perfect matching M, let R be a permutation matrix such that the edges of M lie on the diagonal of AR. Consider the directed graph H with vertices $\{1, 2, \ldots, n\}$ and edges $\{ij \mid AR(i,j)=1\}$. H has adjacency matrix AR. There is no directed cycle in H, except for the trivial cycles on the diagonal; a nontrivial cycle would correspond to an alternating cycle of M and non-M edges of G, thus M would not be unique. Since H is acyclic, its reflexive transitive closure is a partial order on $\{1, 2, \ldots, n\}$, and thus extends to a total order. Let S be a permutation matrix that reorders $\{1, 2, \ldots, n\}$ in respect of this total order, so that if $i \geq j$ then $S^{-1}ARS(i,j)=0$. In other words, $S^{-1}ARS$ is upper triangular. The result is then obtained by taking $P=S^{-1}$ and $Q=RS$. $\square$

Thus, in order to check whether the matching M is unique, we first compute the matrix AR, which can be done by the method of [RV] described above. Then the matching M in G will be unique iff the directed graph H described in the proof of Lemma 1 has no cycles. There are very simple NC algorithms to check this. We have shown

Theorem 2.2. There is an NC algorithm for testing if a given bipartite graph has a unique perfect matching

3. Comparability Graphs

An undirected graph is called a *comparability graph* if there is a way to orient the edges so that the resulting binary relation is transitive. In this section we give an NC algorithm for the following problem:

Input: An undirected graph $G=(V,E)$.

Output: A transitive orientation of the edges, or indication that no such orientation exists.

The first polynomial time algorithms for this problem were given independently by Gilmore and Hoffman [GH] and Ghouila-Houri [Gh]. They proved:

Theorem 3.1 [GH, Gh]. A graph G is a comparability graph iff each odd cycle has a triangular chord.

This property is easy to check in NC. However, producing a transitive orientation in case one exists is a harder problem. We use a decomposition theorem that allows a graph to be decomposed according to its comparability structure, allowing separate parts of the graph to be oriented independently. This decomposition was first discovered by Gallai [Ga]; see Kelly [K] for an excellent account in English. Considerations of efficiency require us to be more careful in the development of the Gallai decomposition. Our development, which we give below in full, departs from the standard development [K,M] in several technical respects, perhaps the most important of which is the inclusion of edges of the complementary graph in the definition of the relation $\angle$, the symmetry of G and G^c, and the use of the relation $\measuredangle$. Otherwise, basic notation and terminology follow [K,M].

Let $G=(V,E)$ be an undirected graph without multiple edges or loops. We represent E as a set of ordered pairs such that $E=E^{-1}=\{uv \mid vu \in E\}$. Let $E^c=\{uv \mid u \neq v\}-E$ and $G^c=(V,E^c)$. Note that $(E^c)^c=E$. Many definitions and results below are symmetric in E and E^c; in the statement of such results, we use the letter F to represent either E or E^c, and F^c to represent the other. We will also use the word *edge* to denote any pair $uv \in V^2$, $u \neq v$. We denote the induced subgraph of (V,F) on $A \subseteq V$ by (A,F).

Definition 3.2. A *transitive orientation* of G is a subset T of E such that

(i) for each $uv \in E$, either $uv \in T$ or $vu \in T$ but not both;
(ii) if uv, $vw \in T$, then $uw \in T$.

The graph G is called a *comparability graph* if there exists a transitive orientation. □

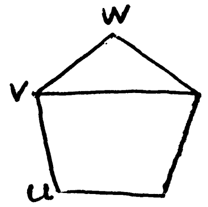
Definition 3.3. For $uv, u'v' \in F$, define $uv \angle u'v'$ if either

(i) $u = u'$ and $vv' \in F^c$, or
(ii) $v = v'$ and $uu' \in F^c$.

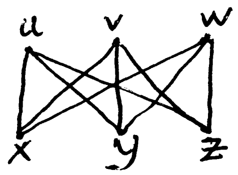
Let $\angle^*$ be the reflexive transitive closure of $\angle$. The $\angle^*$-class of uv is denoted $[uv]$ and is called the *implication class* of uv. The set $[uv] \cup [vu]$ is called the *color class* of uv. The set of vertices touched by edges of $[uv]$ is denoted $V[uv]$. □

Unlike [K], $\angle$ and $\angle^*$ are defined on edges in E^c as well as E, and on directed edges instead of undirected edges. Note, however, that $[uv]=[u'v']$ iff $[vu]=[v'u']$, thus $[vu]=[uv]^{-1}$.

Example 3.4. The graph (a) has exactly two color classes: $[uv] \cup [vu] \subseteq E$ and $[uw] \cup [wu] \subseteq E^c$. The graph (b) has seven color classes, one in E forming a complete bipartite graph between two E^c-connected components, and six color classes in E^c, three in each of the E^c-connected components.



(a)



(b)

□

The significance of implication classes is that a particular orientation of one edge forces an orientation of all the other edges in its implication class. This is expressed in the following lemma.

Lemma 3.5. For any transitive orientation T, if $uv \in T$ and $[uv]=[u'v']$, then $u'v' \in T$. In other words, for each uv, either $[uv] \subseteq T$ or $[uv] \cap T = \emptyset$.

Proof. Immediate from the definition of transitive orientation. □

In view of Lemma 3.5, Theorem 3.1 can be restated: G is a comparability graph iff $[uv] \neq [vu]$. This condition is easily checked in NC by a simple transitive closure procedure on edges. We will henceforth assume that the condition holds of the given graph G.

Lemma 3.6. If $uv \in F$, then $(V[uv], F^c)$ has either one or two connected components. If two, then the edges of $[uv]$ form a complete bipartite graph from one component to the other.

Proof. It can be shown by induction on the definition of $\angle^*$ from $\angle$ that every vertex in $V[uv]$ is connected to either u or v by an path in F^c. Thus $(V[uv], F^c)$ has at most two components. If there are two components, say A, B with $u \in A$ and $v \in B$, then $u'v' \in F$ for every $u' \in A$, $v' \in B$. Since (B, F^c) is connected, $[uv]=[uv']$. Since (A, F^c) is connected, $[uv']=[u'v']$. Thus $[u'v']=[uv]$ for every $u' \in A$ and $v' \in B$. Moreover, this exhausts the edges of $[uv]$, since $V[uv]=A \cup B$. □

Definition 3.7. An edge $uv \in F$ is said to be of *type 1* if $(V[uv], F^c)$ has 1 connected component, *type 2* if $(V[uv], F^c)$ has 2 connected components. □

Definition 3.8. For $uv, u'v' \in F$, define $uv \measuredangle u'v'$ if either

(i) $u=u'$ and $vv' \notin [vu] \cup [uv']$, or
(ii) $v=v'$ and $uu' \notin [uv] \cup [vu']$.

Let $\measuredangle^*$ be the reflexive transitive closure of $\measuredangle$. The $\measuredangle^*$-class of uv is denoted $\llbracket uv \rrbracket$. The set of vertices touched by edges of $\llbracket uv \rrbracket$ is denoted $V\llbracket uv \rrbracket$. □

Lemma 3.9.

(i) $[uv] \subseteq \llbracket uv \rrbracket$, and $[uv]=\llbracket uv \rrbracket$ if uv is of type 1.

(ii) If uv, uw, $vw \in F$, $\llbracket uw \rrbracket = \llbracket vw \rrbracket$, and $u'v' \in [uv]$, then $[u'w]=[uw]$ and $[v'w]=[vw]$.

Proof. The first statement of (i) is immediate from the definitions. For (ii), if $uv \angle uv'$, then $v'w \in F$, otherwise $uw \angle uv' \angle uv$, contradicting $\llbracket uw \rrbracket = \llbracket vw \rrbracket$. Then $v'w \angle vw$, and by the first statement of (i), $\llbracket v'w \rrbracket = \llbracket vw \rrbracket = \llbracket uw \rrbracket$. If $uv \angle u'v$, a symmetric argument shows that $u'w \in F$, $u'w \angle uw$ and $\llbracket u'w \rrbracket = \llbracket uw \rrbracket = \llbracket vw \rrbracket$. The result follows by induction on the definition of $\angle^*$.

For the latter statement of (i), suppose $uv \measuredangle uw$ but $[uv] \neq [uw]$. Then $uw \measuredangle vw$, and either all three edges uv, vw, wu are in F, or all three are in F^c. Let $A=\{u' \mid [u'v']=[uv] \text{ for some } v'\}$, $B=\{v' \mid [u'v']=[uv] \text{ for some } u'\}$. By (ii), $[wu']=[wu]$ for every $u' \in A$, and $[wv']=[wv]$ for every $v' \in B$. Since $[wu] \neq [wv]$, $A \cap B = \varnothing$. Moreover, for every $u' \in A$, $v' \in B$, $u'v' \in F$. Therefore A, B are two F^c-connected components of $V[uv]$, and uv is of type 2. □

Normal subgraphs and kernels

Definition 3.10. A partition π of V is called a *kernel* if for all distinct $A, B \in \pi$ and all $u, u' \in A$, $v, v' \in B$, $[uv]=[u'v']$. A subset $A \subseteq V$ is called *normal* if for all $u \notin A$ and $v, v' \in A$, $[uv]=[uv']$. $\square$

The definition of normal set is related to the definition of *autonomous set* in [K,M]. Our definition differs in that it involves edges in E^c as well as E.

Kernels and normal subsets are related by the following lemma.

Lemma 3.11. A partition π is a kernel iff every $A \in \pi$ is normal. Every normal set A is contained in some kernel.

Proof. The direction $(\rightarrow)$ of the first statement is immediate from the definitions. Conversely, let A, B be distinct elements of π, $u, u' \in A$, $v, v' \in B$. Since B is normal, $[uv]=[uv']$, and since A is normal, $[uv']=[u'v']$. Therefore $[uv]=[u'v']$. Since singletons are trivially normal, every normal set A is contained in a kernel, namely $\{A\} \cup \{\{u\} \mid u \notin A\}$. $\square$

Lemma 3.12. For any $uv \in F$, the connected components of $(V[uv], F^c)$ are normal sets.

Proof. Since $V[uv]$ is connected by edges in the color class of uv, for any $s \notin V[uv]$, either all st, $t \in V[uv]$, are in F or all are in F^c, otherwise there would be some edge in $[uv]$, say uv itself, such that $su \in F$ iff $sv \in F^c$. But then either $[su]=[vu]$ or $[sv]=[uv]$, contradicting the assumption that $s \notin V[uv]$. If s is F-connected to all $t \in V[uv]$, then all edges from s to a connected component of $(V[uv], F^c)$ are in the same implication class. If s is F^c-connected to all $t \in V[uv]$, then all st, $t \in V[uv]$ are in the same implication class; the argument is the same as in Lemma 3.6. If uv is of type 2, then the edges from one connected component of $(V[uv], F^c)$ to the other are all in the same implication class, by Lemma 3.6. $\square$

Lemma 3.13. $V[\![uv]\!]$ is the smallest normal set containing uv.

Proof. First we show that $V[\![uv]\!]$ is normal. Since $V[\![uv]\!]$ is connected by edges in $[\![uv]\!]$ and $[\![vu]\!]$, for any $s \notin V[\![uv]\!]$, all edges st with $t \in V[\![uv]\!]$ are in the same implication class, otherwise there would be some edge in $[\![uv]\!]$, say uv itself, such that $[su] \neq [sv]$. But then either $[\![su]\!]=[\![vu]\!]$ or $[\![sv]\!]=[\![uv]\!]$, contradicting the assumption that $s \notin V[\![uv]\!]$.

Now suppose that A is normal and $u, v \in A$. If $v' \notin A$ such that $uv \angle uv'$, then $[vv'] \neq [uv']$, which is impossible if A is normal. Therefore $V[\![uv]\!] \subseteq A$. $\square$

Lemma 3.14. If A, B are normal, then either $A \subseteq B$, $B \subseteq A$, or $A \cap B = \varnothing$.

Proof. Suppose $u \in A - B$, $v \in A \cap B$, $w \in B - A$. Then $[uv]=[uw]$ since B is normal. But since A is normal, this contradicts Lemmas 3.9(i) and 3.13. □

Lemma 3.15. If A is normal in G and $B \subseteq A$, then B is normal in (A, E) iff B is normal in G.

Proof. Trivially, if B is normal in G then it is normal in (A, E). Conversely, if B is normal in (A, E), then for every vertex $u \in A - B$, the set of edges uv, $v \in B$, are all in the same implication class. Since A is normal in G, for any $u \in V - A$, the set of edges uv, $v \in B$, are all in the same implication class. Thus B is normal in G. □

Simple graphs, quotient graphs and comparability morphisms

Definition 3.16. A graph is called *simple* if it has no nontrivial kernels. Equivalently, a graph is *simple* if it has no nontrivial normal subsets. □

Example 3.17. The graph 3.4(a) is simple. The graph 3.4(b) has a nontrivial kernel $\pi = \{\{u, v, w\}, \{x, y, z\}\}$. □

Definition 3.18. If π is a kernel, the *quotient graph* G/π is obtained by collapsing each $A \in \pi$ to a single node:

$$G/\pi = (\pi, \{AB \mid A, B \in \pi,\ A \neq B,\ uv \in E \text{ for some } u \in A, v \in B\}.$$

The map $f_\pi : G \to G/\pi$ is defined by

$f_\pi(u) = A$, where A is the unique element of π containing u,
$f_\pi(uv) = f_\pi(u) f_\pi(v)$, for u, v such that $f_\pi(u) \neq f_\pi(v)$. □

Example 3.19. The graph of Example 3.4(a) is simple, thus has no quotients except itself and the trivial one-vertex graph. The graph G of Example 3.4(b) has a nontrivial quotient G/π, where π is the kernel of Example 3.17. G/π consists of a pair of directed edges between two vertices. □

The map f_π may be called a *comparability morphism*, since it preserves the relation $\angle^*$:

Lemma 3.20. If π is a kernel in G, $f = f_\pi$, $f(u) \neq f(v)$, and $f(u') \neq f(v')$, then $[uv]=[u'v']$ in G iff $[f(uv)]=[f(u'v')]$ in G/π. In other words, if $f(u) \neq f(v)$, then $f^{-1}([f(uv)])=[uv]$.

Proof. By definition of kernel, $uv \in E$ iff $f(uv) \in E/\pi$. If $uv \angle uv'$ in G, then either $f(v)=f(v')$, in which case $f(uv)=f(uv')$ in G/π; or $f(v) \neq f(v')$, in which case $f(uv) \angle f(uv')$ in G/π. In either case $f(uv') \angle^* f(uv')$. Conversely, if $f(uv) \angle f(uv')$ in G/π, then $f(v) \neq f(v')$, and $uv \angle uv'$ in G. The result follows by induction on the definition of $\angle^*$. □

Lemma 3.21. A is normal in G/π iff $f_\pi^{-1}(A)$ is normal in G. The partition ρ is a kernel in G/π iff $\{f_\pi^{-1}(A) \mid A \in \rho\}$ is a kernel in G.

Proof. Let $f_\pi(u) \neq f_\pi(v)$ and $f_\pi(u) \neq f_\pi(v')$. Then $u \notin f_\pi^{-1}(A)$ and $v, v' \in f_\pi^{-1}(A)$ iff $f_\pi(u) \notin A$ and $f_\pi(v)$, $f_\pi(v') \in A$. By Lemma 3.20, $[uv]=[uv']$ iff $[f_\pi(uv)]=[f_\pi(uv')]$. Therefore $f_\pi^{-1}(A)$ is normal iff A is normal.

The second statement is immediate from the first. □

Lemma 3.22. π is a maximal kernel in G iff G/π is simple.

Proof. If ρ is a nontrivial kernel in G/π, then by Lemma 3.21, $\{f_\pi^{-1}(A) \mid A \in \rho\}$ is a nontrivial kernel in G that properly contains π, thus π was not maximal. Conversely, if π is not maximal, then by Lemma 3.14, there is a nontrivial kernel $\sigma \neq \pi$ such that each element of σ is a union of elements of π. Then $\sigma=\{f_\pi^{-1}(f_\pi(A)) \mid A \in \sigma\}$. Taking $\rho=\{f_\pi(A) \mid A \in \sigma\}$ in Lemma 3.21, it follows that ρ is a nontrivial kernel in G/π. □

Theorem 3.23. Every graph G decomposes into a product of simple graphs.

Proof. Let T be the tree of all normal subsets of G, ordered by inclusion. This tree exists, by Lemma 3.14. For any normal subset A, the successors of A in T are the maximal proper normal subsets of A, by Lemma 3.15. By Lemma 3.11, this set forms a maximal kernel π in (A,E), therefore by Lemma 3.22, $(A,E)/\pi$ is simple. □

Theorem 3.24. Let $\pi=\{V_1, \ldots, V_n\}$ be a kernel and let $G_i=(V_i,E)$. There is a one-to-one correspondence between transitive orientations T of G and tuples $(T_1, \ldots, T_n, T/\pi)$ of transitive orientations of $G_1, \ldots, G_n$ and G/π, respectively.

Proof. Let $f=f_\pi$. Given a transitive orientation T of G, let

$$T_i = T \cap V_i^2,$$
$$T/\pi = \{f(uv) \mid f(u) \neq f(v),\ uv \in T\}.$$

T_i is just the orientation induced on G_i as a subgraph of G and is clearly transitive. If $f(u) \neq f(v)$, then $f(uv) \in T/\pi$ iff $f(vu) \notin T/\pi$ by Lemma 3.5, so T/π is a valid orientation. T/π is transitive, since if $f(uv), f(vw) \in T/\pi$ then $uv, vw \in T$ by Lemma 3.5, therefore $uw \in T$. Also, $f(u) \neq f(w)$, otherwise $[uv]=[wv]$, and by

Lemma 3.5 both wv and vw would be in T. Thus $f(uw) \in T/\pi$.

Conversely, if T_i, $1 \le i \le n$, and T/π are given, let

$$T = \bigcup_{1 \le i \le n} T_i \cup f^{-1}(T/\pi).$$

If $f(u)=f(v)=V_i$, $u,v \in V_i$, then $uv \in T$ iff $uv \in T_i$ iff $vu \notin T_i$ iff $vu \notin T$. If $f(u) \ne f(v)$, then $uv \in T$ iff $f(uv) \in T/\pi$ iff $f(vu) \notin T/\pi$ iff $vu \notin T$, so T is a valid orientation. To show T is transitive, suppose $uv, vw \in T$. There are three essential cases, from which the others follow by symmetry:

(i) $f(u)=f(v)=f(w)=V_i$
(ii) $f(u)=f(v)=V_i$, $f(w) \ne V_i$
(iii) $f(u) \ne f(v)$ and $f(v) \ne f(w)$.

Case (i) follows from the transitivity of T_i. In case (ii), $f(uw)=f(vw)$, and $f(vw) \in T/\pi$. Therefore $f(uw) \in T/\pi$ and $uw \in T$. Finally, in case (iii), it cannot be that $f(u)=f(w)$, otherwise $[uv]=[wv]$, and then by Lemma 3.20 $[f(uv)]=[f(wv)]$, so by Lemma 3.5, $f(wv) \in T/\pi$, contradicting the fact that $f(vw) \in T/\pi$. Then $f(uv) \in T/\pi$ and $f(vw) \in T/\pi$, therefore $f(uw) \in T/\pi$, and therefore $uw \in T$. □

Theorem 3.25. The following are all the normal subsets of G:

(i) singletons $\{u\}$, $u \in V$;
(ii) $V[\![uv]\!]$, $u, v \in V$.

Proof. These sets are normal, by Lemma 3.13. Let A be any normal set with at least two elements. Let π be a maximal proper kernel in A. Then $(A, E)/\pi$ contains at least two vertices $f_\pi(u)$, $f_\pi(v)$, where $u, v \in A$. By Lemma 3.13, $V[\![uv]\!]$ is the smallest normal set containing u and v, therefore by Lemma 3.14, $f_\pi(u) \cup f_\pi(v) \subseteq V[\![uv]\!] \subseteq A$. But then $A = V[\![uv]\!]$, otherwise π was not maximal. □

Classification of simple graphs

The following theorem classifies all simple graphs into one of three types. It is a much simplified restatement of Theorems 1.2 and 1.8 of [K].

Theorem 3.26. A graph G is simple iff either:

(i) G is a clique;
(ii) G is totally disconnected (so that G^c is a clique);
(iii) G contains at least four vertices and has exactly two color classes, one each in E and E^c.

Proof. First we show that any simple graph must be of one of the three given forms. Let G be simple. Any simple graph of three or fewer vertices is of the form (i) or (ii), by inspection. Thus assume G has at least 4 vertices.

If G has a type 2 edge uv, then by Lemma 3.12, $[uv]$ is a singleton, otherwise G has a nontrivial kernel. Using Lemma 3.9, it can be shown that $V[\![uv]\!]$ is a clique of type 2 edges, all of which form singleton implication classes. By Lemma 3.13, $V[\![uv]\!]$ is normal, and since G is simple, $V[\![uv]\!]=V$. Then G is of the form (i).

If all uv are of type 1, then $V[uv]=V$ for every $u, v \in V$, by Lemmas 3.9(i) and 3.13. We argue that for any uv, $u'v'$ in F, either $[uv]=[u'v']$ or $[uv]=[v'u']$. Suppose not. Since $V=V[u'v']$, there exists $w \in V$ such that either $[uw]=[u'v']$ or $[uw]=[v'u']$. In either case, $[uw]\neq[uv]$. Then $vw \in F$, and one of $[vw]\neq[vu]$, $[vw]\neq[uw]$. Without loss of generality, assume the former. Then $uw \measuredangle vw$, so by Lemma 3.9(ii), $w \notin V[uv]$, contradicting the fact that $V=V[uv]$. We have shown that there are exactly two implication classes in F, therefore one color class in F.

We now show that each of the types (i)-(iii) is simple. For (i), a clique cannot contain any nontrivial normal subset, because all implication classes are singletons, so $V[\![uv]\!]=V$ for all u, $v \in V$. By Lemma 3.13, G is simple. Case (ii) is symmetric. For G of type (iii), suppose there is a nontrivial maximal proper kernel π. By Lemma 3.22, G/π is simple, thus G/π is of one of the forms (i)-(iii). At least one implication class vanishes when passing to a nontrivial quotient, and Lemma 3.20 implies that each implication class in G/π is the image under f_π of an implication class in G. Thus G/π cannot have any of the following forms, otherwise G has too many implication classes: form (iii); form (i) or (ii) with three or more vertices; form (i) or (ii) with two vertices, so that $\pi=\{A, B\}$, and each of A, B contains at least two vertices, or one of A, B contains at least three vertices. The only other alternative is that G has three vertices, which is explicitly ruled out in the definition of type (iii). $\square$

An NC algorithm for obtaining a transitive orientation

Consider the following algorithm for producing a transitive orientation of a comparability graph:

(1) Determine all classes $[uv]$ and $[\![uv]\!]$.

(2) Find all normal sets.

(3) Create the tree of normal sets, ordered by inclusion.

(4) For each normal set A, find a transitive orientation of the quotient $(A, E)/\pi$, where π is a maximal proper kernel in A.

(5) Let these orderings induce an ordering on G under the maps f_{π}^{-1}, according to Theorem 3.24.

Step (1) is a transitive closure computation on edges. Step (2) determines the sets $V[\![uv]\!]$ from this information, which by Theorem 3.25 gives all normal sets. Step (3) involves the construction of the Hasse diagram of the inclusion relation on normal sets. Step (4) first computes a representation of the quotient graph $(A, E)/\pi$, where π is the set of descendants of A in the tree of normal sets. This is a maximal proper kernel, by Lemmas 3.11 and 3.14. If $(A, E)/\pi$ is a simple graph of type (i), then any arbitrary ordering among the $m!$ possible orderings of the m vertices of $(A, E)/\pi$ gives a transitive orientation. If $(A, E)/\pi$ is a simple graph of type (ii), then there is only one transitive orientation, namely as an antichain. If $(A, E)/\pi$ is a simple graph of type (iii), then there are only two possible transitive orientations, corresponding to the two implication classes of $(A, E)/\pi$. Finally, (5) is a matter of notation. All these computations can easily be done in NC using standard algorithms. We have proved

Theorem 3.27. There is an NC algorithm for checking if a given graph $G=(V, E)$ is a comparability graph, and if so, for obtaining a transitive orientation of its edges. □

A stronger result can be shown: there is an NC algorithm for enumerating all transitive orientations of a given undirected graph. By this we mean that there are NC circuits which, given an undirected graph and a binary number k, produce the k^{th} transitive orientation in some ordering. This can be obtained from the Gallai decomposition. We leave the details to the reader.

4. Interval Graphs

In this section we give an *NC* algorithm for the following problem.

Input: An undirected graph $G=(V,E)$.

Output: An interval graph representation of G, or an indication that no such representation exists.

Definition 4.1 [GH]. Let σ be a linearly ordered finite set. An *interval* α of σ is any set of contiguous elements of σ. Let I be a set of intervals on σ. The pair (σ, I) is an *interval representation* of the undirected graph $G=(V,E)$ if there exists a bijection $f: V \to I$ such that $uv \in E$ iff the intervals $f(u)$ and $f(v)$ intersect. An undirected graph $G=(V,E)$ is an *interval graph* iff it has an interval representation. $\square$

Define an *incomparability graph* to be an undirected graph G whose complement G^c is a comparability graph. The following characterization immediately gives us an *NC* algorithm for checking if G is an interval graph:

Theorem 4.2 [GH]. G is an interval graph iff (i) G is an incomparability graph, and (ii) every cycle of length 4 in G has a diagonal.

Our *NC* algorithm for obtaining an interval representation for G is essentially a parallelization of the sequential algorithm of Gilmore and Hoffman [GH], using the transitive orientation algorithm of §3. We give below this sequential algorithm:

(1) Transitively orient the edges of G^c.

(2) Find a set of maximal cliques in G such that every vertex and every edge is in at least one such clique. Let this set of cliques be denoted σ. For each pair of cliques $C_1, C_2 \in \sigma$, there is an edge of G^c connecting some vertex in C_1 to some vertex in C_2 (otherwise $C_1 \cup C_2$ would be a clique). Gilmore and Hoffman prove that under the orientation assigned in step 1, either all such edges are directed from C_1 to C_2, or else all are directed from C_2 to C_1. Moreover, if we define $C_1 \leq C_2$ if these edges are directed from C_1 to C_2, then this linearly orders σ [GH].

(3) For any vertex $v \in G$, let $\alpha(v)$ be the set of maximal cliques to which v belongs. These cliques will be contiguous, and thus will form an interval of σ [GH]. Let I be the set of all the intervals corresponding to the vertices of G. Then (σ, I) is an interval representation of G.

We will now parallelize this algorithm:

(1) Transitively orient G^c using Theorem 3.27.

(2) Do in parallel for each edge $uv \in G$: find a maximal clique C_{uv} containing uv.

(3) Remove redundant cliques. Let σ be the list of cliques remaining.

(4) Do in parallel for each pair $C_i, C_j \in \sigma$, $i<j$: examine in parallel all pairs of vertices $u \in C_i$, $v \in C_j$, and consult the orientation obtained in (1) to determine the order of C_i and C_j. This gives the linear order on σ.

(5) Do in parallel for each vertex $v \in G$: obtain the interval $\alpha(v)$. This will yield the set of intervals I.

To accomplish (2), we can use the parallel maximal clique algorithm of Karp and Wigderson [KW], or the more efficient algorithm of Luby [Lu]. We thus obtain:

Theorem 4.3. There is an *NC* algorithm which checks if a given graph $G=(V, E)$ is an interval graph, and if so obtains an interval representation for it.

5. Parallel Matching Algorithms

We will use Theorem 3.27 to obtain parallel maximum matchings in incomparability graphs. Recall that an *incomparability graph* is an undirected graph G whose complement G^c is a comparability graph, and that interval graphs form a subclass of incomparability graphs.

Our algorithm will make use of a fast parallel algorithm for the two-processor scheduling problem. This problem can be stated as follows:

> *Input*: A directed acyclic graph $G=(V,E)$ whose vertices represent unit time jobs and whose edges specify precedence constraints among the jobs.
>
> *Output*: An optimal schedule for the jobs on two identical processors satisfying all the precedence constraints.

Two jobs which are scheduled in the same time unit on the two processors are said to be *paired*. The connection between this problem and the maximum matching problem is established by the following theorem:

Theorem 5.1 (Fujii, Kasami, and Ninomiya [FKN]). Let $G=(V,E)$ be a directed, acyclic graph, and let G^{*c} be the complement of its transitive closure. Then the paired jobs in an optimal schedule for G form a maximum matching in G^{*c}.

This theorem enabled [FKN] to obtain a fast sequential algorithm for two-processor scheduling, using a maximum matching algorithm as a subroutine. We will do the reverse.

Vazirani and Vazirani [VV] gave a Random *NC* algorithm for the two-processor scheduling problem, using the *RNC* matching algorithm of Karp, Upfal, and Wigderson [KUW] as a subroutine. This gave a weaker version of Theorem 5.2 below which showed the maximum matching problem for interval graphs to be in *RNC*. This was the first non-trivial class of graphs for which maximum matchings could be obtained fast in parallel. It was subsumed by [KUW], who showed that the general matching problem was in *NC*.

More recently, Helmbold and Mayr [HM] have obtained an *NC* algorithm for two-processor scheduling. This allows us to dispense with randomization in the case of incomparability graphs.

Theorem 5.2. There is an *NC* algorithm which, given an incomparability graph $G=(V,E)$, obtains a maximum matching in it.

Proof. Given an incomparability graph G, first obtain a transitive orientation of G^c using Theorem 3.27. The resulting graph is then a transitively closed acyclic digraph; compute an optimal two-processor schedule for it using Theorem 5.1. By [FKN], the list of paired jobs will be a maximum matching in G. $\square$

Acknowledgments

We are grateful to Ashok Chandra, Alan Hoffman, David Kelly, and Laszlo Lovasz for several inspiring and informative discussions.

References

[BCP] A. Borodin, S.A. Cook, and N. Pippenger, "Parallel computation for well-endowed rings and space bounded probabilistic machines," *Information and Control 58:1-3* (1983), 113-136.

[BGH] A. Borodin, J. von zur Gathen, and J. Hopcroft, "Fast parallel matrix and GCD computations," *Proc. 23rd ACM Symp. on Theory of Computing* (1982), 65-71.

[EPL] S. Even, A. Pnueli, and A. Lempel, "Permutation graphs and transitive graphs," *J. Assoc. Comput. Mach. 19* (1972), 400-410.

[FKN] M. Fujii, T. Kasami, and K. Ninomiya, "Optimal sequencing of two equivalent processors," *SIAM J. Comput.* ???

[Ga] T. Gallai, "Transitiv orientierbare Graphen," *Acta Math. Acad. Sci. Hungar. 18* (1967), 25-66.

[GH] P.C. Gilmore and A.J. Hoffman, "A characterization of comparability graphs and of interval graphs," *Canad. J. Math. 16* (1964), 539-548.

[Gh] A. Ghouila-Houri, "Caracterisation des graphes nonorientes dont on peut orienter les aretes de maniere a obtenir le graphe d'une relation d'ordre," *C.R. Acad. Sci. Paris 254* (1962), 1370-1371.

[Go] M.C. Golumbic, "Comparability graphs and a new matroid," *J. Combinatorial Theory 22:1* (Feb. 1977), 68-90.

[K] D. Kelly, "Comparability graphs," in: Graphs and Order, ed. I. Rival, Nato ASI series C, v. 147, D. Reidel (1985), 3-40.

[HM] D. Helmbold and E. Mayr, "Two processor scheduling is in *NC*," to appear.

[KUW] R.M. Karp, E. Upfal, and A. Wigderson, "Finding a maximum matching is in Random *NC*," *Proc. 26th ACM Symp. on Theory of Computing* (1985), 22-32.

[KW] R.M. Karp and A. Wigderson, "A fast parallel algorithm for the maximal independent set problem," *Proc. 25th ACM Symp. on Theory of*

Computing(1984), 266-272.

[Lo] L. Lovasz, "On determinants, matchings, and random algorithms," to appear.

[Lu] M. Luby, "A simple parallel algorithm for the maximal independent set problem," *Proc. 26th ACM Symp. on Theory of Computing* (1985), 1-10.

[M] R.H. Mohring, "Algorithmic aspects of comparability graphs and interval graphs," in: Graphs and Order, ed. I. Rival, Nato ASI series C, v. 147, D. Reidel (1985), 41-102.

[PLE] A. Pnueli, A. Lempel, and S. Even, "Transitive orientation of graphs and identification of permutation graphs," *Canad. J. Math. 23* (1971), 160-175.

[RV] M.O. Rabin and V.V. Vazirani, "Maximum matchings in general graphs through randomization," Report No. TR15-84, Center for Research in Computing Technology, Harvard University, Cambridge, Massachusetts, 1984.

[Tu] W.T. Tutte, "The factors of graphs," *Canad. J. Math. 4* (1952) pp.314-328.

[VV] U.V. Vazirani and V.V. Vazirani, "The two-processor scheduling problem is in Random *NC*," *Proc. 26th ACM Symp. on Theory of Computing* (1985), 11-21.