THEORY OF MACHINES AND COMPUTATIONS

QUEUES, STACKS AND GRAPHS

S. Even and A. Itai

The Weizmann Institute of Science
Rehovot, Israel

Abstract

The problem of realizing a given permutation through a network of queues in parallel and through a network of stacks in parallel is considered. Each one of these problems is translated into a coloration problem of a suitable graph. In the case of parallel queues, the resulting graph is a permutation graph, and therefore very easy to color. The number of queues necessary to realize the permutation is equal to the chromatic number of the graph. The case of parallel stacks defines two different problems, depending on whether or not we insist on completion of the stacks loading before unloading them. If we accept this condition, the resulting graph is again a permutation graph. However, if we allow unloading of stacks before loading is complete the resulting graph, called the union graph, may not be a permutation graph. An alternative way of defining a union graph as the graph describing the intersection of chords in a circle is shown. In spite of the fact that the set of all union graphs is a proper subset of all graphs, no efficient algorithm to color them has yet been found.

1. Queues in Parallel

A queue is a linear storage device. It has one entrance and one exit. The elements exit in the same order in which they have entered (FIFO). The elements may be stored any length of time and the number of elements that can be stored in any one queue is assumed to be unbounded.

AP 6 x 9 - 10M-10/65

Assume we have, in a queue A, the elements $1,2,\ldots,n$ in their natural order and we want to transfer them into a queue B in such a way that their order is $P(1),P(2),\ldots,P(n)$, where P is some permutation. Clearly, $P^{-1}(i)$ is the place (the number of the place when the places are enumerated from left to right) in which i appears in B after the transfer.

Assume we have m intermediate queues $Q_1,Q_2,\ldots,Q_m$. We may take the first element in A and transfer it to some Q_i (it exits A and enters Q_i). This may be followed by transferring the second element to some Q_j, where i may or may not be identical with j, etc. We shall refer to a transfer from A to some Q_i as <u>loading</u>. No transfers from one Q_i to another (or the same) Q_j is allowed. The only place elements may enter upon exiting a Q_i is to enter B. This type of transfer is called <u>unloading</u>. No unloading of B is allowed. Naturally, this configuration is referred to as a <u>system of m queues in parallel</u>.

Two different modes of operations are defined by either insisting that all n elements be loaded before unloading begins, or not.

It is clear that every permutation P on n elements is realizable through a system of n queues, by first loading each element on a separate intermediate queue and then unloading $P(1)$, then $P(2)$, etc. Our aim is to find the smallest m such that a system of m queues in parallel will suffice to realize the given P. Following Even, Lempel and Pnueli [1], a <u>permutation graph</u> $\vec{G}(N,\vec{R})$ is defined for P where the set of vertices N is $\{1,2,\ldots,n\}$ and the set of directed edges $\vec{R}$ is given by

$$\vec{R} = \{i \to j \mid i < j \text{ and } P^{-1}(i) > P^{-1}(j)\}.$$

<u>Lemma 1</u>: In every realization of P by a system of queues in parallel, if $i \to j$ in $\vec{G}$ then i and j do not pass through the same intermediate queue.

<u>Proof</u>: If $i \to j$ then $i < j$. Thus i exits A before j. If they enter the same Q_k then i is before j in the queue. Thus, $P^{-1}(i) < P^{-1}(j)$. A contradiction.

Q.E.D.

Clearly if $i \neq j$ and $i < j$, then $P^{-1}(i) < P^{-1}(j)$ and there exists a realization in which i and j do pass through the same queue.

Lemma 2: If P is realizable through a system of m queues in parallel, then there exists a partition of N to m blocks such that no two elements in one block are connected in $\vec{G}$.

Proof: Define the block B_i to consist of all the elements which pass through Q_i, and use Lemma 1. Q.E.D.

Lemma 3: If N has a partition into m blocks such that no two elements in one block are connected in $\vec{G}$ then P is realizable through a system of m queues in parallel.

Proof: We can load all the elements of a block B_k into Q_k. Clearly, in each queue the elements are in their natural order. (Loading is completed before unloading begins.)

There can be no element b which precedes $P(1)$ in its queue. (This would have implied that $b < P(1)$, and since $P^{-1}(b) > 1$, it would have implied that $b \rightarrow P(1)$, and they cannot belong to the same block.) Thus, $P(1)$ can be unloaded. Assume $P(1), P(2), \ldots P(\ell)$ have been unloaded. $P(\ell+1)$ is the first element in its queue, by a similar argument.

Q.E.D.

Note that in the realization of the proof we first complete the loading before unloading begins. Thus, if a realization without this restriction exists with m queues, by Lemma 2 we have a partition into m blocks such that no two elements in one block are connected in $\vec{G}$, and by the proof of Lemma 3 we get a realization satisfying the restriction. Thus, we have proved the following:

Lemma 4: Every permutation which is realizable by a system of m queues in parallel is also realizable by the same system with completion of loading before unloading begins.

Thus, the problem of finding the smallest m such that P is realizable by a system of m queues in parallel is equivalent to minimal coloration of $\vec{G}$. Very efficient algorithms for coloring permutation graphs were given by Even, Lempel and Pnueli [1].

2. Stacks in Parallel-Loading Before Unloading

A stack is also a linear storage device. It has only one orifice. The elements exit in the reverse order of their entrance (LIFO). Here too, the elements may be stored any length of time and the number of elements that can be stored in one stack at any time is unbounded.

Again we shall assume that the elements of N are stored on an input queue A in their natural order, and that they have to be transferred to an output queue B in the order specified by P. Assume we have m intermediate stacks $S_1, S_2, \ldots, S_m$. We may transfer from A to any one of the stacks, and from any stack to B. No transfer between stacks or reloading A or unloading B is allowed. This is called a system of m stacks in parallel.

The graph $\vec{G}'(N,\vec{R}')$ is defined by

$$\vec{R}' = \{i \to j \mid i < j \text{ and } P^{-1}(i) < P^{-1}(j)\} .$$

$\vec{G}'(N,\vec{R}')$, as the notation suggests, is also a directed graph and is the complement of $\vec{G}$. (That is, if $i < j$ then $i \to j$ in G' if and only if $i \not\to j$ in $\vec{G}$).

Lemma 5: In every realization of P by a system of stacks in parallel, with completion of loading before unloading begins, if $i \to j$ in $\vec{G}'$ then i and j do not pass through the same stack.

Proof: If $i \to j$ then $i < j$. Thus i exits A before j. If they enter the same S_k then j is stored on top of i. Thus, $P^{-1}(j) < P^{-1}(i)$. A contradiction. Q.E.D.

Clearly, if $i \not\to j$ in $\vec{G}'$ and $i < j$, then $P^{-1}(i) > P^{-1}(j)$ and there exists a realization by stacks in parallel, with completion of loading before unloading, in which i and j pass through the same stack. Indeed, j

is stored on top of i, but it is also unloaded before i.

Lemma 6: If P is realizable by a system of m stacks in parallel, with loading completed before unloading begins, then N can be partitioned into m blocks such that no two elements in one block are connected by an edge in $\vec{G}'$.

The proof follows immediately by Lemma 5.

Lemma 7: If N can be partitioned into m blocks such that no two elements which are connected by an edge in $\vec{G}'$ are in the same block, then P is realizable by a system of m stacks in parallel, where loading is completed before unloading begins.

The proof is similar to the proof of Lemma 3. Thus, the problem of finding the smallest number of stacks in a system of parallel stacks, for realizing a given permutation P, when we insist on completion of loading before unloading begins, is equivalent to the coloration problem of $\vec{G}'$. Even, Lempel and Pnueli [1] showed that $\vec{G}'$ is a permutation graph, and thus the problem of its coloration is very easy to solve.

3. Stacks in Parallel - Without Restrictions on Unloading

Unlike the situation in queues in parallel, the removal of the restriction on completion of loading before unloading begins creates a completely new problem. Consider, for example, the identity permutation I ($I(i) = i$ for $i=1,2,\ldots,n$). $\vec{G}'$ is a completely connected graph (an edge between every two vertices) and therefore its chromatic number is n. However, one stack is sufficient to realize it: Load 1, unload 1, load 2, unload 2, etc.

Our aim is to define a graph whose coloration problem is equivalent to that of finding the smallest number of stacks necessary in a parallel system for realizing a given permutation P.

Define for every $k=1,2,\ldots,n$ a graph $\vec{G}^k(V^k,\vec{R}^k)$ as follows:

$$V^k = \{i \mid i < k \text{ and } P^{-1}(k) < P^{-1}(j)\}$$

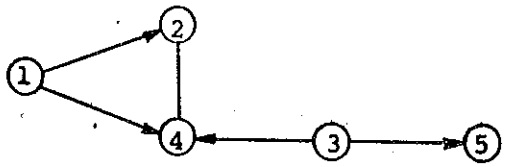
$$\vec{R}^k = \{i \to j \mid i,j \in V^k \text{ and } i < j \text{ and } P^{-1}(i) < P^{-1}(j)\}.$$

The union graph, $\vec{G}_U(V,\vec{R})$ is defined by

$$V = \bigcup_{k=1}^{n} V^k \quad \text{and} \quad \vec{R} = \bigcup_{k=1}^{n} \vec{R}^k.$$

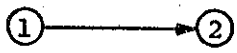
Each $\vec{G}^k$ is a subgraph of $\vec{G}'$ of the previous section. Therefore, $\vec{G}_U$ is also a subgraph of $\vec{G}'$, but is not necessarily identical to it.

Example 1: $P = [3,5,1,2,4]$. $\vec{G}'(N,\vec{R}')$ is shown below.

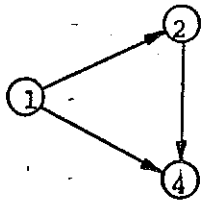


Now, $\vec{G}^1$, $\vec{G}^2$ and $\vec{G}^4$ are empty.

$\vec{G}^3$ consists of the vertices 1 and 2:



$\vec{G}^5$ is shown below.



Thus, the union graph, $\vec{G}_U$ is identical with $\vec{G}^5$ and is not equal to $\vec{G}'$.

Lemma 8: If $i \to j$ in $\vec{G}_U$ then there is no realization of P by a system of stacks in parallel in which i and j pass through the same stack.

Proof: If $i \to j$ in $\vec{G}_{U,k}$ then there exists an element k_{-1} such that $i \to j_{-1}$ in $\vec{G}^{k}$. Thus, $i < j < k$ and $P^{-1}(k) < P^{-1}(i) < P^{-1}(j)$. Therefore, i and j are loaded on the stacks before k leaves A and k is unloaded from the stack it passes through before i and j leave the stacks in which they are stored. Thus, if both i and j are stored on the same stack, j is stored on top of i and must be unloaded first. This contradicts $P^{-1}(i) < P^{-1}(j)$.

Q.E.D.

Let S be a set of elements which pass through a given stack in some realization. Lemma 8 implies that $S \cap V$ is an independent set in $\vec{G}_U$ (a set of vertices such that no two of them are connected by an edge). Thus, a realization of P by a system of stacks in parallel implies a coloration of the vertices of $\vec{G}_U$.

Conversely, let $\{C'_1, C_2, \ldots, C_m\}$ be a coloration of the vertices of $\vec{G}_U$. (Each C_i is the set of vertices which are colored by the i-th color. Clearly, no two vertices in one C_i are connected by an edge in $\vec{G}_U$.) Let $C_1 = C'_1 \cup (N-V)$. Thus, $\{C_1, C_2, \ldots, C_m\}$ is a partition of N.

Lemma 9: If $\{C_1, C_2, \ldots, C_m\}$ is a partition of N such that $\{C_1 \cap V, C_2 \cap V, \ldots, C_m \cap V\}$ is a legal coloration of $\vec{G}_U$ then there exists a realization of P by a system of m stacks $S_1, S_2, \ldots, S_m$ in parallel in which the elements of C_ℓ pass through S_ℓ.

Proof: Let us use the following rule: Assume $P(1), P(2), \ldots, P(k-1)$ are already on B. If P(k) is available (on top of one of the S_i's) for unloading, unload it and repeat the rule. If not, load the next element in line on A to its corresponding stack and repeat the rule.

Let us prove now, by induction on k, that it will never happen that $P(1), P(2), \ldots, P(k-1)$ are already on B, P(k) is out of A and yet it is not available for unloading.

First consider P(1). If it is already out of A,

then its exit out of its stack cannot be blocked, because when it first enters the stack, the rule guarantees that it is immediately unloaded. Now, assume the statement is true for $i = 1,2,\ldots,k-1$, and that $P(1),P(2),\ldots,P(k-1)$ are already on B and $P(k)$ is already out of A. If $P(k)$ is not on top in its stack, then some element a is on top of it. Let the elements on B be $P(1),P(2),\ldots,P(i-1)$ when a is loaded. By the inductive hypothesis $P(i)$ is not loaded yet. Thus, $P(i) > a$. We also know that $i < k$, $a > P(k)$ and $P^{-1}(a) > k$. Summarizing, we have

$$P(k) < a < P(i) \quad \text{and} \quad P^{-1}(P(i)) < P^{-1}(P(k)) < P^{-1}(a).$$

Thus in $\vec{G}^{P(i)}$, $P(k) \to a$ and $P(k)$ and a cannot have the same color in $\vec{G}_U$.

Q.E.D.

Lemmas 8 and 9 imply that the problem of finding the smallest m such that P is realizable by a system of m stacks in parallel, when unloading is permitted before loading is completed, is equivalent to the problem of finding minimum coloration of its union graph. Let us now show an example of a union graph which is not a permutation graph. In fact it is not even transitive.

Example 2: $P = [5,2,7,4,1,6,3]$.

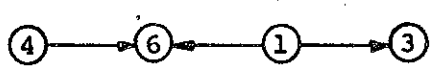
$\vec{G}^1$ and $\vec{G}^3$ are empty.

$\vec{G}^2$ is 

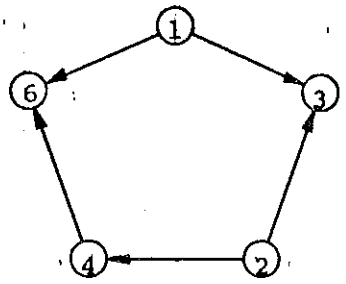
$\vec{G}^4$ is 

$\vec{G}^5$ is 

$\vec{G}^6$ is 

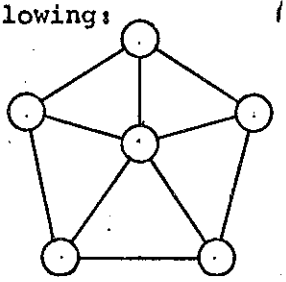
$\vec{G}^7$ is 

Thus, $\vec{G}_U$ is as follows:



Clearly, there is no way of redirecting its edges to make it transitive.

One may suspect that every undirected graph is the layout of some union graph, and that the coloration problem for union graphs is as hard as for graphs in general. Unfortunately, we have not found an efficient algorithm for coloring union graphs as yet. However, pessimism is not justified. First, the fact that the edges of the union graph are directed may be of help, as in the case of transitive graphs [1]. Second, there exist undirected graphs which are not the layout of any union graph. One such graph is the following:



We postpone the proof of this fact to the next section.

Lemma 10: If $i \to j$ in $\vec{G}_U$ and both i and j are vertices of $\vec{G}^k$, then $i \to j$ in $\vec{G}^k$.

Proof: Since $i \to j$ in $\vec{G}_U$ then there exists an ℓ such that $i < j < \ell$ and $P^{-1}(\ell) < P^{-1}(i) < P^{-1}(j)$. However, since i and j are in $\vec{G}^k$, then $i < k$, $j < k$ and $P^{-1}(k) < P^{-1}(i)$ and $P^{-1}(k) < P^{-1}(j)$. It follows that

$$i < j < k \quad \text{and} \quad P^{-1}(k) < P^{-1}(i) < P^{-1}(j) .$$

Thus, $i \to j$ in $\vec{G}^k$. Q.E.D.

A graph H is called a vertex-subgraph of a graph G if the set of vertices of H is a subset of the set of vertices of G and all the edges of G connecting vertices which appear in H are also edges of H. H has no other edges. (This is sometimes called a section graph.)

Lemma 10 implies that each $\vec{G}^k$ is a vertex-subgraph of $\vec{G}_U$.

Lemma 11: If $\ell \varepsilon V$ then there exists a $k \neq \ell$ such that $\vec{G}^\ell$ is a vertex-subgraph of $\vec{G}^k$.

Proof: Since $\ell \varepsilon V$, there exists a k such that $\ell \varepsilon V^k$. Thus, $\ell < k$ and $P^{-1}(k) < P^{-1}(\ell)$. If $i \varepsilon V^\ell$ then $i \leq \ell$ and $P^{-1}(\ell) < P^{-1}(i)$. Thus, $i < k$ and $P^{-1}(k) < P^{-1}(i)$ and $i \varepsilon V^k$. Furthermore, if $i \to j$ in $\vec{G}^\ell$ then $i < j < \ell$ and $P^{-1}(\ell) < P^{-1}(i) < P^{-1}(j)$. Thus, $i < j < k$ and $P^{-1}(k) < P^{-1}(i) < P^{-1}(j)$. Therefore $i \to j$ in $\vec{G}^k$.

Q.E.D.

Corollary: $\vec{G}_U$ is the union of graphs $\vec{G}^k$ such that $k \not\varepsilon V$.

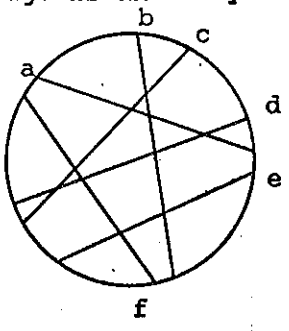
The following lemma is not difficult to prove.

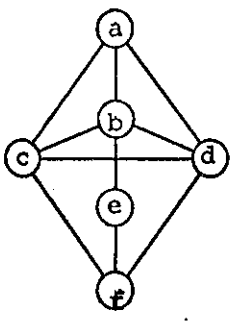
Lemma 12: Every vertex-subgraph of the union graph of some permutation is the union graph of some permutation.

4. Circle Graphs and their Relation to Union Graphs

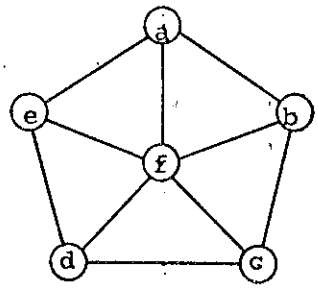
Let C be a set of chords drawn in a circle. Define the corresponding undirected circle graph G(C,R) by representing each chord by a vertex; two vertices are connected by an edge if and only if the corresponding chords intersect.

Example 3:

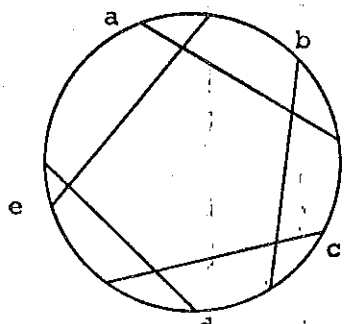




Let us show now an undirected graph which is not the circle graph of any circle with chords:



The only way of getting a circuit a—b—c—d—e—a in a circle graph is by drawing five chords as follows:



It is now impossible to add a sixth chord which will intersect all the existing five chords.

Next, let us describe an algorithm for generating a permutation whose union graph has a layout isomorphic to the given circle graph of a given circle with chords. (We assume the number of chords is not zero.)

Algorithm 1:

(1) Choose a point on the circle which is not an end point of a chord. Call it q (the first artificial vertex). Also set $i \leftarrow 1$.

(2) Move (clockwise, starting at q) along the circle until you reach an end point of a chord, p.

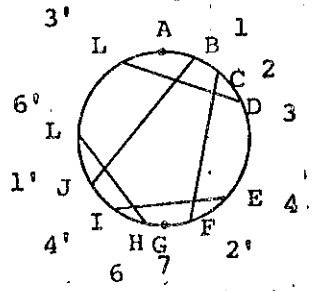
(3) Let p have the label i and the other end point of the same chord have the label i'. Increment i ($i \leftarrow i + 1$).

(4) Move (clockwise, starting at p) along the circle. If you reach an unlabeled end point of a chord, call it p and go to Step (3). If you reach a labeled end point of a chord (it must have a primed label), let q have the label i' and increment i (i ← i + 1) .

(5) Move (clockwise, starting at the last labeled end point found in Step (4)), ignore additional labeled end points of chords as you go along. If you reach an artificial vertex, stop. If you reach an unlabeled end point of a chord p (a new chord) put a new artificial vertex, q , on the arc between the last end point encountered and q and go to Step (3).

The resulting permutation is achieved by reading the primed vertices off the circle, starting from the first artificial vertex and tracing clockwise.

Example 4: Consider a circle with five chords which has a pentagon as its circle graph:



Assume we start at point A ; this is the first artificial vertex. (If we start between B and C we get a different permutation.) Going clockwise, we label B as 1 and J' as 1' ,C as 2 and F as 2' , D as 3 and L as 3' , E as 4 and I as 4' . Next we encounter F which is already labeled 2' . The present value of i is 5 , thus A is labeled 5' . We continue to trace from F and encounter an end point of a new chord, H . Thus, we put a new artificial vertex, G , between F and H . We label H as 6 and K as 6' . Next, we encounter I which is already labeled 4' . We label G as 7' and continue tracing from I . No unlabeled vertices are encountered until we reach the artificial vertex A and stop. The resulting permutation is:

$$p = [5,2,7,4,1,6,3]$$

See Example 2.

Theorem I: Assume P is a permutation which is derived from a circle with chords through Algorithm 1. The underlying undirected graph of the union graph of P is identical with the circle graph.

Proof: Assume the algorithm assigns the end points of a given chord the labels i and i'. There exists an artificial vertex before i' (the one nearest to it) with a label k', where $k > i$. Thus, i is a vertex in the union graph.

An artificial vertex is assigned a number after all the vertices which precede it are labeled. Thus, its label is higher than these and it is not a vertex in the union graph.

Thus, we have already shown that the circle graph and the union graph have the same set of vertices.

Next, assme that $i—j$ in the circle graph; that is, the chord i intersects the chord j in the circle. (Here the chord names are as assigned in the algorithm.) Thus, the labels i, j, i', j' appear in this order on the circle. Consider the last artificial vertex before j, say k'. The algorithm implies that $k > j$ and therefore, $i \rightarrow j$ in the union graph.

Finally, assume $i \rightarrow j$ in the union graph. Thus, there exists a k, not in the union graph (by the Corollary following Lemma 11) such that $i < j < k$ and $P^{-1}(k) < P^{-1}(i) < P^{-1}(j)$. Thus, i' appears before j' on the circle, and k' appears before i'. If i' appears before j, then there must be an artificial vertex between i' and j, and $k < j$. Therefore i,j,i',j' must appear in this order and $i—j$ in the circle graph.

Q.E.D.

Now assume we are given a permutation P. The following algorithm will construct a circle with chords such that the underlying undirected graph of the union graph of

P is identical with the circle graph. Let V be the set of vertices in the union graph. Let us call the numbers in N-V artificial vertices (they will turn out to be similar to the artificial vertices defined in Algorithm 1).

Algorithm 2:

(1) Draw a circle. Choose n points around it and label them P(1),P(2),...,P(n) , in this order, with primes.

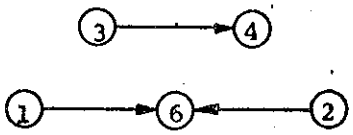
(2) Set k ← P(1) and i ← 1 . (k is an artificial vertex.)

(3) If i is an artificial vertex, reset k to be the next artificial vertex following i in P , and go to Step (5).

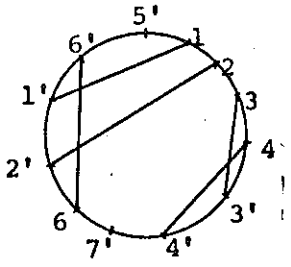
(4) Draw vertex i in the arc between k' and the next primed vertex, after other unprimed vertices which have already been drawn in this arc.

(5) Increment i (i ← i + 1) . If i < n , go to Step (3). Otherwise, stop.

Example 5: Let P = [5,3,4,7,2,1,6] . Its union graph is



and the artificial vertices are 5 and 7 .



First we draw the points 5',3',4',7',2',1',6' on the circle in this order. We start with k = 5 and i = 1 . Next, vertices 1,2,3 and 4 are drawn by Steps (3), (4), (5) applied four times. Now i = 5 and 5 is an

artificial vertex. We set $k = 7$, $i = 6$ and vertex 6 is drawn between 7' and 2'. Now $i = 7$, and we stop. The resulting chords and labels are shown above.

Theorem 2: Let P be any permutation. Algorithm 2 will construct a circle with chords whose circle graph is identical with the underlying undirected graph of P's union graph $\vec{G}_U(V,\vec{R})$.

Proof: For every i, i' is drawn on the circle in Step (1). Note that n (the highest integer in P) is always artificial and that the subsequence of artificial vertices of P is always an increasing sequence. Assuming that $n > 1$, it will never happen that Step (3) is applied to $i = n$, and if $i < n$, then there is a next artificial vertex to become the value of k.

The algorithm will also draw every non artificial vertex i. Now, assume $i \to j$ in $\vec{G}_U$. Thus there exists a least artificial vertex k such that $i < j < k$ and $P^{-1}(k) < P^{-1}(i) < P^{-1}(j)$. Thus, i' is drawn before j' on the circle. Clearly, i is drawn before j, and when the running artificial vertex is k, j is drawn before i'. Thus the four vertices i, j, i', j' appear in this order and $i—j$ in the circle graph.

If $i—j$ in the circle graph, and $i < j$, then i, j, i', j' appear in this order on the circle. Also, j comes after some artificial vertex $k > j$. Thus, $i < j < k$ and $P^{-1}(k) < P^{-1}(i) < P^{-1}(j)$, and $i \to j$ in $\vec{G}_U$.

Q.E.D.

Corollary: Every circle graph is the layout of some union graph, and the layout of every union graph is a circle graph.

Thus, the coloration of circle graphs is the same problem as the coloration of union graph. Also, since we have shown a graph which is not a circle graph (Example 3), the same graph is not the layout of any union graph.

The interested reader may find related results in references 2,3 and 4. Some of our results, and additional

results on related subjects are reported in 5. The circle graph has been suggested to us by Yagil [6].

References

1. S. Even, A. Lempel, and A. Pnueli, "Permutation Graphs and Transitive Graphs", to appear in the JACM.

2. D. E. Knuth, The Art of Computer Programming. Vol. 1, Addison-Wesley (1969), pp. 234-239 and 531-534.

3. D. E. Knuth, The Art of Computer Programming, Vol. 3. To appear.

4. A. Pnueli, A. Lempel, and S. Even, "Transitive Orientation of Graphs and Identification of Permutation Graphs", Canadian J. of Math. Vol. 23, No. 1, pp.160-175 (1971).

5. R. Tarjan, "Sorting Using Networks of Queues and Stacks". Unpublished.

6. S. Yagil, "Linear Programming for Traffic Light Optimization". I.B.M. Technical Information Exchange, April (1966).