An Efficient Parallel Recognition Algorithm For Bipartite-Permutation Graphs

Chang-Wu Yu and Gen-Huey Chen

Abstract—In this paper, we present a parallel recognition algorithm for bipartite-permutation graphs. The algorithm can be executed in $O(\log n)$ time on the CRCW PRAM if $O(n^3/\log n)$ processors are used, or $O(\log^2 n)$ time on the CREW PRAM if $O(n^3/\log^2 n)$ processors are used. Previously, Chen and Yesha have presented another CRCW PRAM algorithm that takes $O(\log^2 n)$ time if $O(n^3)$ processors are used. Compared with Chen and Yesha's algorithm, our algorithm requires either less time and fewer processors on the same machine model, or fewer processors on a weaker machine model. Besides, our algorithm can be applied to determine if two bipartite-permutation graphs are isomorphic.

Index Terms—Bipartite-permutation graph, graph recognition, graph isomorphism, parallel algorithm, and parallel random access machine.

1 Introduction

An undirected graph G with vertex set $\{v_1, v_2, ..., v_n\}$ is called a *permutation graph* [12] if there exists a permutation π on $N = \{1, 2, ..., n\}$ such that for all $i, j \in N$,

$$(i - j)\left(p^{-1}(i) - p^{-1}(j)\right) < 0$$

if and only if v_i and v_j are joined by an edge in G. Pictorially, draw the vertices $v_1, v_2, ..., v_n$ in order on a line, and $v_{\pi(1)}, v_{\pi(2)}, ..., v_{\pi(n)}$ on a parallel line such that for each $i \in N$, v_i is directly above $v_{\pi(i)}$. Next, for each $i \in N$, draw a line segment from v_i on the upper line to v_i on the lower line. Then, there is an edge (v_i, v_j) in G if and only if the line segment for v_i intersects the line segment for v_j. As an illustrative example, Fig. 1 shows a permutation $\pi = (4, 7, 5, 1, 2, 6, 3)$ and its corresponding permutation graph. Permutation graphs, due to their special structural properties, have been used in modeling and solving various problems such as determining intersection-free layouts for connection boards [16], determining optimal schedules for reallocation of memory space in a computer [14], and assigning safe flight altitudes for airline routes [12]. In [24], these three applications are described with more details.

Many researches [1], [2], [3], [4], [5], [6], [10], [13], [15], [17], [19], [20], [21], [22], [25], [26] have been devoted to the study of permutation graphs. In [17], Pnueli et al. proposed an $O(n^3)$ time algorithm that recognizes a permutation graph of n vertices by applying the transitively orientable graph test. Later, Spinrad [19] improved their result by presenting an $O(n^2)$ time recognition algorithm. Besides, Spinrad also gave an algorithm that determines if two permutation graphs are isomorphic with the same time complexity. In [13], [15], the problem of recognizing a permutation graph was shown to be in the NC class. In [21], Supowit solved the coloring problem, the maximum clique problem, the cliques cover problem, and the maximum independent set problem, all in $O(n\log n)$ time. In [10], using dynamic programming, Farber and Keil solved the weighted domination problem and the weighted independent domination problem in $O(n^3)$ time. In [2],

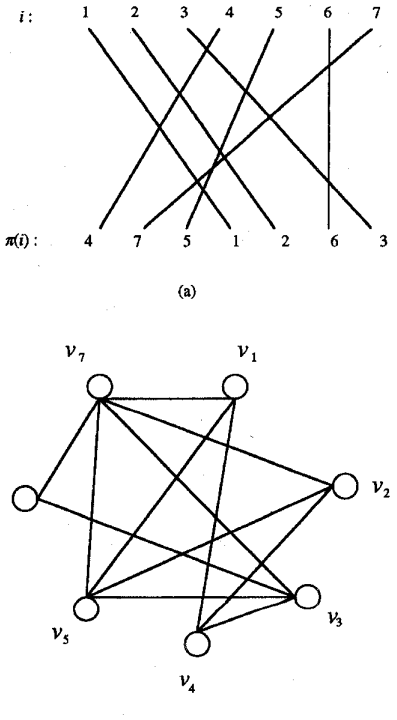


Fig. 1. An example. (a) A permutation π, and (b) its corresponding permutation graph.

• *The authors are with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, R.O.C.. E-mail: ghchen@csie.ntu.edu.tw.*

Manuscript received Mar. 12, 1992; revised Apr.. 8, 1994.
For information on obtaining reprints of this article, please send e-mail to: transactions@computer.org, and reference IEEECS Log Number D95064.

Brandstadt and Kratsch presented an $O(n^2)$ time algorithm for the weighted independent domination problem. In [1], Atallah et al. solved the independent domination set problem in $O(n\log^2 n)$ time. In [22], Tsai and Hsu solved the domination problem and the weighted domination problem in $O(n\log\log n)$ time and $O(n^2\log^2 n)$ time, respectively. In [25], [26], Yu and Chen presented various NC algorithms for permutation graphs.

A *bipartite graph* is a graph whose vertex set can be partitioned into two subsets S and T such that each of its edges has one end in S and the other end in T. A permutation graph which is also bipartite is called a *bipartite-permutation graph*. Bipartite-permutation graphs were mentioned in [3] as models arising if a certain product requires machine parts from two sets which should not differ too much. The properties of bipartite-permutation graphs lead to linear time algorithms for recognition and isomorphism [20]. In addition, there are efficient polynomial time algorithms for a number of problems which are known to be NP-hard or are of unknown complexity for both bipartite graphs and permutation graphs, including hamiltonicity, crossing number, jump number, minimum fill-in, and a number of vertex deletion problems [3].

In [20], Spinrad et al. presented an algorithm that recognizes a bipartite-permutation graph in $O(m + n)$ time. In [6], Chen and Yesha presented a parallel algorithm that recognizes a bipartite-permutation graph in $O(\log^2 n)$ time using $O(n^3)$ processors on the CRCW PRAM (Concurrent-Read Concurrent-Write Parallel Random Access Machine). A graph is *isomorphic* to another if there is a one-to-one mapping from one vertex set to another so that the adjacency property is preserved. The problem of finding such a mapping between two graphs remains open even restricted to bipartite graphs [11]. In [5], Chen designed a parallel algorithm for testing isomorphism of two bipartite-permutation graphs in $O(\log^2 n)$ time using $O(n^3)$ processors on the CRCW PRAM.

In this paper, we present a parallel algorithm that recognizes a bipartite-permutation graph in $O(\log n)$ time using $O(n^3/\log n)$ processors on the CRCW PRAM or $O(\log^2 n)$ time using $O(n^3/\log^2 n)$ processors on the CREW PRAM (Concurrent-Read Exclusive-Write Parallel Random Access Machine). Besides, our algorithm can be applied to solve the problem of testing isomorphism of bipartite-permutation graphs. Compared with Chen and Yesha's algorithm [5], [6], our algorithm requires either less time and fewer processors on the same machine model or fewer processors on a weaker machine model.

The rest of this paper is organized as follows. In the next section, we introduce some necessary definitions and notations. In Section 3, we prove a theorem about bipartite-permutation graphs. Based on the theorem, a parallel algorithm for recognizing bipartite-permutation graphs is presented in Section 4. Finally, we conclude this paper with some remarks in Section 5.

2 Definitions and Notations

Let $G = (S, T, E)$ represent a connected bipartite graph, where $S \cup T$ is the set of vertices, E is the set of edges, and each edge in E has one end in S and the other end in T. Also, let $N(v)$ denote the set of vertices which are adjacent to v in G. An ordering of S (T) has the *adjacency property* if for each vertex $v \in T$ (S), $N(v)$ contains consecutive vertices in the ordering. On the other hand, an ordering of S (T) has the *enclosure property* if for every pair of vertices $t_1, t_2 \in T$ (S) satisfying $N(t_1) \subseteq N(t_2)$, $N(t_2) - N(t_1)$ contains consecutive vertices in the ordering. An illustrative example is shown in Fig. 2.

The combination of an ordering of S and an ordering of T is called a *strong ordering* if any two edges (s_i, t_l), $(s_j, t_k) \in E$ imply $(s_i, t_k) \in E$ and $(s_j, t_l) \in E$, where $s_i, s_j \in S$, $t_k, t_l \in T$, s_i precedes s_j in the ordering of S, and t_k precedes t_l in the ordering of T. To make the definition clearer, let us imagine the vertices of S: $s_1, s_2, \ldots, s_{|S|}$ arranged on a line and the vertices of T: $t_1, t_2, \ldots, t_{|T|}$ on a parallel line. There is a strong ordering of S and T if for all $s_i, s_j \in S$, $t_k, t_l \in T$, (s_i, t_k) and (s_j, t_l) exist whenever (s_i, t_l) and (s_j, t_k) cross. The combination of the ordering of S and the ordering of T shown in Fig. 2 forms a strong ordering.

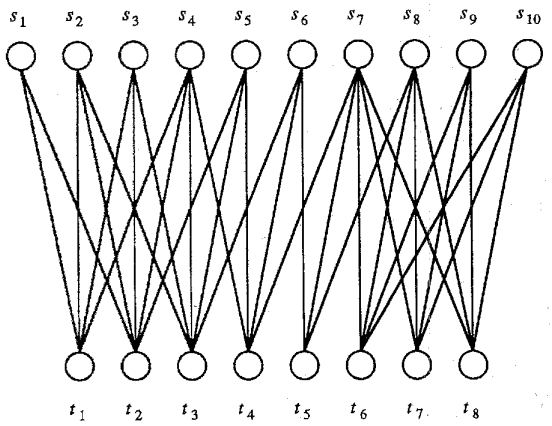


Fig. 2. An ordering of S and an ordering of T which have the adjacency and the enclosure properties.

Suppose that U and V are nonempty subsets of S and T, respectively. The subgraph of G whose vertex set is $U \cup V$ and whose edge set contains those edges of G having both ends in $U \cup V$ is called the subgraph of G *induced by* U and V, and is denoted by $G_{U,V}$.

Suppose that $s_1, s_2, \ldots, s_{|S|}$ and $t_1, t_2, \ldots, t_{|T|}$ are orderings of S and T, respectively. We define the following notations.

$$p(s_i) = min\{ j \mid t_j \in N(s_i)\}; p(t_i) = min\{ j \mid s_j \in N(t_i)\}.$$

$$q(s_i) = max\{ j \mid t_j \in N(s_i)\}; q(t_i) = max\{ j \mid s_j \in N(t_i)\}.$$

$A = \{s_i \mid s_i \in S$ and there does not exist $s_j \in S$ such that $N(s_i)$ is a proper subset of $N(s_j)\}$.

$B = S - A$.

$C = \{ s_i \mid s_i \in B$ and one of the following two conditions holds:

1) there exists exactly one vertex $s_j \in A$ and there exists another vertex $s_k \in A$ such that $N(s_i)$ is a proper subset of $N(s_j), N(s_j) \cap N(s_k) \neq \emptyset$, and $N(s_j) \cap N(s_k) \subseteq N(s_i)$, and

2) there exist exactly two vertices $s_j, s_k \in A$ $(j \neq k)$ such that $N(s_j) \cap N(s_k) = N(s_i)$ and $N(s_i)$ is a proper subset of $N(s_j)$ and $N(s_k)$}.

3 A Fundamental Theorem

In this section, we prove a theorem about bipartite-permutation graphs, which form the basis of the parallel algorithm in the next section.

LEMMA 1. *(Theorem 1 in [20]). The following statements are equivalent for a bipartite graph $G = (S, T, E)$.*

1) *G is a bipartite-permutation graph.*
2) *There is a strong ordering of S and T.*
3) *There exists an ordering of S (or T) which has the adjacency and enclosure properties.*

The ordering of S and the ordering of T, which form a strong ordering, mentioned in statement 2) of Lemma 1 have the adjacency and enclosure properties, provided G is connected. This can be seen from the proof of Theorem 1 in [20] and is stated as the following lemma.

LEMMA 2. *Suppose that $G = (S, T, E)$ is a connected bipartite graph and there exists a strong ordering of S and T. Then, the ordering of S and the ordering of T, which form the strong ordering, have the adjacency and enclosure properties.*

Suppose that $G = (S, T, E)$ is a connected bipartite-permutation graph. By Lemma 2, G can be expressed as a rectilinear polygon with a shape like Fig. 3 or Fig. 4. The polygon is constructed by placing a square at the position (t_j, s_i) for each edge $(s_i, t_j) \in E$, where $s_i \in S$ and $t_j \in T$. There are two structural properties about the polygon: 1) each row (column) of the polygon contains a continuous block of squares; 2) the right (left) boundary of the polygon is non-decreasing (Fig. 3) or non-increasing (Fig. 4) in x-coordinate. If the combination of the ordering of S and the ordering of T forms a strong ordering, the polygon has a unique shape like Fig. 3. Fig. 3 shows the corresponding polygon of Fig. 2.

For each of the two polygons, we can partition it into three parts: top region (R_T), middle region (R_M), and bottom region (R_B). Without loss of generality, we assume that $s_1, s_2, \ldots, s_{|S|}$ (with the adjacency property) is the ordering of S associated with the polygon. Then, $R_T = \{s_1, s_2, \ldots, s_{x-1}\}$, $R_M = \{s_x, s_{x+1}, \ldots, s_y\}$, and $R_B = \{s_{y+1}, s_{y+2}, \ldots, s_{|S|}\}$, where $s_1, s_2, \ldots, s_{x-1} \notin A \cup C$, $s_x, s_y \in A \cup C$, and $s_{y+1}, s_{y+2}, \ldots, s_{|S|} \notin A \cup C$. For example, $R_T = \{s_1, s_2, s_3\}$, $R_M = \{s_4, s_5, s_6, s_7\}$, and $R_B = \{s_8, s_9, s_{10}\}$ for the polygon of Fig. 3. As we will see later, many properties of G are introduced from the polygon.

In the rest of this section, we assume that $N(s_i) \neq N(s_j)$ for all $s_i, s_j \in S$, $s_i \neq s_j$. The following lemma states some basic properties of the rectilinear polygon. Since they are clear by observing the rectilinear polygon, their proof is omitted.

LEMMA 3. *Suppose that $G = (S, T, E)$ is a connected bipartite-permutation graph, and $R_T = \{s_1, s_2, \ldots, s_{x-1}\}$, $R_M = \{s_x, s_{x+1}, \ldots, s_y\}$, and $R_B = \{s_{y+1}, s_{y+2}, \ldots, s_{|S|}\}$. Then, we have*

1) $s_x, s_y \in A$;

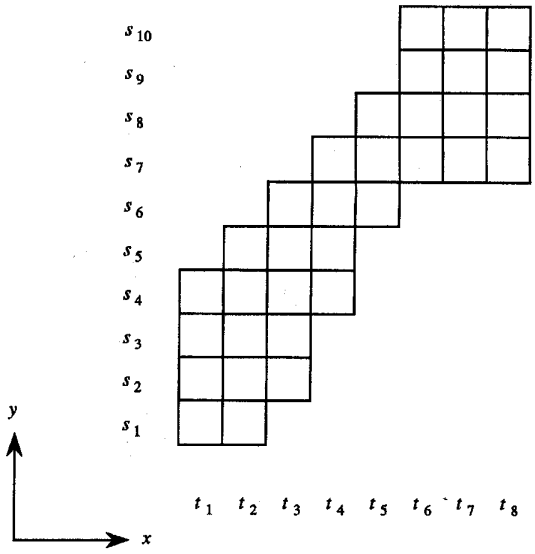


Fig. 3. The corresponding rectilinear polygon of Fig. 2.

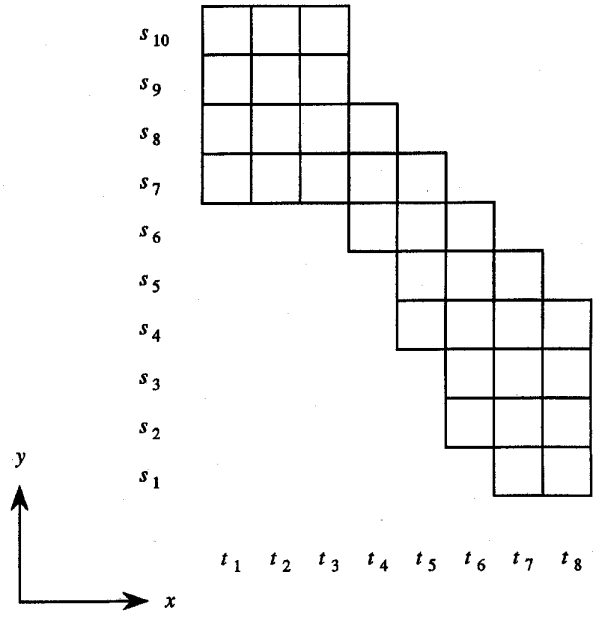


Fig. 4. Another rectilinear polygon representing $G = (S, T, E)$, where there exist an ordering of S and an ordering of T having the adjacency property.

2) $R_M = A \cup C$;
3) $N(s_i) \subset N(s_j)$ *or* $N(s_j) \subset N(s_i)$ *for all* s_i, s_j, *where* $s_i, s_j \in R_T$ *or* $s_i, s_j \in R_B$;
4) *for any strong ordering of* $A \cup C$ *and* T, *denoted by* $s_{r(1)}, s_{r(2)}, \ldots, s_{r(|A \cup C|)}, t_1, t_2, \ldots, t_{|T|}$, $N(s_{r(i)}) \subset N(s_{r(1)})$ *or* $N(s_{r(i)}) \subset N(s_{r(|A \cup C|)})$ *for all* $s_{r(i)} \in S - (A \cup C)$.

LEMMA 4. *Suppose that $G = (S, T, E)$ is a connected bipartite-permutation graph. Then, there are exactly two orderings of A with the adjacency property, and they are the reverse of each other.*

PROOF. We first show that the graph $G_{A,T}$ is connected. Suppose to the contrary that $G_{A,T}$ is disconnected and contains $r > 1$ connected components. Then, T can be partitioned into $T_1, T_2, \ldots, T_r$ each of which is contained in one component of $G_{A,T}$. It is easy to see that for each vertex $s_i \in B$, there exists a vertex $s_j \in A$ and some k, $1 \le k \le r$, such that $N(s_i) \subset N(s_j) \subseteq T_k$. This implies that G is also disconnected, which is a contradiction.

Suppose that $s_{o(1)}, s_{o(2)}, \ldots, s_{o(|A|)}$ is an ordering of A with the adjacency property. Since $G_{A,T}$ is connected, it can be easily seen from the corresponding polygon of G that for any three consecutive vertices $s_{o(i)}, s_{o(i+1)}, s_{o(i+2)}$, $1 \le i \le |A| - 2$, there exist vertices $t_j, t_k \in T$ such that $t_j \in N(s_{o(i)}) \cap N(s_{o(i+1)})$ but $t_j \notin N(s_{o(i+2)})$, and $t_k \in N(s_{o(i+1)}) \cap N(s_{o(i+2)})$ but $t_k \notin N(s_{o(i)})$. This implies that for any ordering of A with the adjacency property, either $pos(s_{o(i)}) < pos(s_{o(i+1)}) < pos(s_{o(i+2)})$ or $pos(s_{o(i)}) > pos(s_{o(i+1)}) > pos(s_{o(i+2)})$ must hold for all i, $1 \le i \le |A| - 2$, where $pos(s_{o(i)})$ denotes the position of $s_{o(i)}$ in the ordering. Consequently, the orderings of A with the adjacency property are either $s_{o(1)}, s_{o(2)}, \ldots, s_{o(|A|)}$ or $s_{o(|A|)}, \ldots, s_{o(2)}, s_{o(1)}$. □

In the following discussion, unless specified particularly, we use $s_{o(1)}, s_{o(2)}, \ldots, s_{o(|A|)}$ to denote either of the two orderings of A with the adjacency property.

LEMMA 5. *Suppose that $G = (S, T, E)$ is a connected bipartite-permutation graph. Then, for each $s_i \in C$, there exists a unique j, $1 \le j \le |A| - 1$, such that*

1) $N(s_{o(j)}) \cap N(s_{o(j+1)}) \subseteq N(s_i)$;
2) *either $N(s_i) \subset N(s_{o(j)})$ or $N(s_i) \subset N(s_{o(j+1)})$ holds;*
3) *s_i is between $s_{o(j)}$ and $s_{o(j+1)}$ in any strong ordering of S and T.*

PROOF. (Part (1)). Considering an arbitrary strong ordering of S and T, we know by Lemma 3 that for each $s_i \in C$, there exist $s_{o(j)}, s_{o(j+1)} \in A$, $1 \le j \le |A| - 1$, such that s_i is between $s_{o(j)}$ and $s_{o(j+1)}$. So, by Lemma 2, for any vertex $t_z \in T$, we have $t_z \in N(s_i)$ if $t_z \in N(s_{o(j)}) \cap N(s_{o(j+1)})$. This implies $N(s_{o(j)}) \cap N(s_{o(j+1)}) \subseteq N(s_i)$.

(Part (2)). Suppose that there exists a vertex $s_i \in C$ such that $N(s_{o(j)}) \cap N(s_{o(j+1)}) \subseteq N(s_i)$ and $N(s_i)$ is not contained in $N(s_{o(j)})$ and $N(s_{o(j+1)})$. Since $G_{A,T}$ is connected (known from the proof of Lemma 4), $N(s_{o(j)}) \cap N(s_{o(j+1)})$ is not empty. Let $t_1, t_2, \ldots, t_{|T|}$ be an ordering of T with the adjacency property, and $N(s_{o(j)}) = \{t_w, t_{w+1}, \ldots, t_x\}$, $N(s_{o(j+1)}) = \{t_y, t_{y+1}, \ldots, t_z\}$, where $y < w < z < x$ or $w < y < x < z$.

If $y < w < z < x$, then $N(s_{o(j)}) \cap N(s_{o(j+1)}) = \{t_w, t_{w+1}, \ldots, t_z\}$. Since $s_i \in C$, there exists a vertex $s_{o(k)} \in A$ such that $N(s_i) \subset N(s_{o(k)})$, where $k \ne j$ and $k \ne j + 1$. Here, we assume $k > j + 1$. For the case of $k < j$, the proof is similar. Let $N(s_{o(k)}) = \{t_p, t_{p+1}, \ldots, t_q\}$, where $p \le w$ and $q \ge z$. If $q > z$, then we have $t_{z+1} \in N(s_{o(k)}) \cap N(s_{o(j)})$ and $t_{z+1} \notin N(s_{o(j+1)})$, which contradicts the adjacency property of the ordering of A. So, we have $q = z$. However, this implies either $N(s_{o(k)}) \subseteq N(s_{o(j+1)})$ or $N(s_{o(j+1)}) \subseteq N(s_{o(k)})$, which is again a contradiction because $s_{o(k)}, s_{o(j+1)} \in A$ and $N(s_x) \ne N(s_y)$ for all $s_x, s_y \in S$, $s_x \ne s_y$. Similarly, the case of $w < y < x < z$ also leads to a contradiction.

(Part (3)). $G\{s_i, s_{o(j)}, s_{o(j+1)}\}$, $N(s_i) \cup N(s_{o(j)}) \cup N(s_{o(j+1)})$ is a connected bipartite-permutation graph, because $N(s_{o(j)}) \cap N(s_{o(j+1)}) \ne \emptyset$ and $N(s_{o(j)}) \cap N(s_{o(j+1)}) \subseteq N(s_i)$. Clearly, we can obtain a strong ordering of $\{s_i, s_{o(j)}, s_{o(j+1)}\}$ and $N(s_i) \cup N(s_{o(j)}) \cup N(s_{o(j+1)})$ from an arbitrary strong ordering of S and T. Since $N(s_i) \subset N(s_{o(j)})$ or $N(s_i) \subset N(s_{o(j+1)})$, s_i is between $s_{o(j)}$ and $s_{o(j+1)}$ in any strong ordering of S and T, as a consequence of Lemma 3 with $S = \{s_i, s_{o(j)}, s_{o(j+1)}\}$, $T = N(s_i) \cup N(s_{o(j)}) \cup N(s_{o(j+1)})$, $C = \{s_i\}$, and $A = \{s_{o(j)}, s_{o(j+1)}\}$. □

LEMMA 6. *Suppose that $G = (S, T, E)$ is a connected bipartite-permutation graph, and $s_1, s_2, \ldots, s_{|S|}, t_1, t_2, \ldots, t_{|T|}$ is an arbitrary strong ordering of S and T. If $s_i, s_j \in S$ and $i < j$, then $p(s_i) \le p(s_j)$ and $q(s_i) \le q(s_j)$.*

PROOF. If $p(s_i) > p(s_j)$, then edges $(s_i, t_{p(s_i)})$, $(s_j, t_{p(s_j)})$ cross. This implies $(s_i, t_{p(s_j)}) \in E$, which is a contradiction. Similarly, the case of $q(s_i) > q(s_j)$ also leads to a contradiction. □

LEMMA 7. *Suppose that $G = (S, T, E)$ is a connected bipartite-permutation graph.*

1) *Let $s_1, s_2, \ldots, s_{|S|}, t_1, t_2, \ldots, t_{|T|}$ be an arbitrary strong ordering of S and T. Then, $N(s_a) \cap N(s_c) \subseteq N(s_a) \cap N(s_b)$ for $1 \le a < b < c \le |S|$ or $1 \le c < b < a \le |S|$.*
2) *If there exist $s_i, s_k \in C (s_i \ne s_k)$ such that $N(s_{o(j)}) \cap N(s_{o(j+1)}) \subseteq N(s_i)$ and $N(s_{o(j)}) \cap N(s_{o(j+1)}) \subseteq N(s_k)$ for some j, $1 \le j \le |A| - 1$, then either $|N(s_{o(j)}) \cap N(s_i)| \ne |N(s_{o(j)}) \cap N(s_k)|$ or $|N(s_{o(j+1)}) \cap N(s_i)| \ne |N(s_{o(j+1)}) \cap N(s_k)|$.*

PROOF. (Part (1)). An immediate consequence of Lemma 6.

(Part (2)). Suppose that $s_1, s_2, \ldots, s_{|S|}, t_1, t_2, \ldots, t_{|T|}$ is a strong ordering of S and T. Also, let $N(s_{o(j)}) = \{t_m, t_{m+1}, \ldots, t_n\}$ and $N(s_{o(j+1)}) = \{t_p, t_{p+1}, \ldots, t_q\}$, where $m < p < n < q$ ($p < m < q < n$ is impossible by Lemma 6). Then, $N(s_{o(j)}) \cap N(s_{o(j+1)}) = \{t_p, t_{p+1}, \ldots, t_n\}$.

Suppose to the contrary $|N(s_{o(j)}) \cap N(s_i)| = |N(s_{o(j)}) \cap N(s_k)|$ and $|N(s_{o(j+1)}) \cap N(s_i)| = |N(s_{o(j+1)}) \cap N(s_k)|$, and let $N(s_i) = \{t_u, t_{u+1}, \ldots, t_v\}$ and $N(s_k) = \{t_w, t_{w+1}, \ldots, t_x\}$, where $m \le u \le p$, $m \le w \le p$, $n \le v \le q$, and $n \le x \le q$. Since $N(s_i) \ne N(s_k)$ is assumed, we have $w \ne u$ or $v \ne x$. If $w \ne u$, then $|N(s_{o(j)}) \cap N(s_i)| \ne |N(s_{o(j)}) \cap N(s_k)|$, which is a contradiction. If $v \ne x$, then $|N(s_{o(j+1)}) \cap N(s_i)| \ne |N(s_{o(j+1)}) \cap N(s_k)|$, which is also a contradiction. □

LEMMA 8. *Suppose that $G = (S, T, E)$ is a connected bipartite-permutation graph. If $|A| \ge 2$, then there exists no $s_i \in S - (A \cup C)$ such that $N(s_i) \subset N(s_{o(1)})$ and $N(s_i) \subset N(s_{o(|A|)})$.*

PROOF. Suppose that $s_1, s_2, \ldots, s_{|S|}, t_1, t_2, \ldots, t_{|T|}$ is an arbitrary strong ordering of S and T. By Lemma 2, we know that the ordering $s_1, s_2, \ldots, s_{|S|}$ has the adjacency property, which implies that the ordering of A embedded in the ordering of S also has the adjacency property. By Lemma 4, we have $\{s_{o(1)}, s_{o(|A|)}\} = \{s_k, s_l\}$, for some k, l, $1 \le$

$k \le l \le |S|$. By Lemma 3, we have either $1 \le i \le k-1$ or $l+1 \le i \le |S|$ for each $s_i \in S-(A \cup C)$.

Suppose that there exists some vertex $s_\alpha \in S-(A \cup C)$ such that $N(s_\alpha) \subset N(s_k)$ and $N(s_\alpha) \subset N(s_l)$. Here, we assume $1 \le \alpha \le k-1$. For the case of $l+1 \le \alpha \le |S|$, the proof is similar. We let $N(s_k) = \{t_m, t_{m+1}, \ldots, t_n\}$, $N(s_l) = \{t_p, t_{p+1}, \ldots, t_q\}$, and $N(s_\alpha) = \{t_u, t_{u+1}, \ldots, t_v\}$ (because $N(s_\alpha) \subseteq N(s_k) \cap N(s_l)$), where $m < p < n < q$ and $p < u < v \le n$. Since $\alpha < k$, $m < p \le u$, and (s_α, t_u), $(s_k, t_m) \in E$, we have $(s_\alpha, t_m) \in E$ because the strong ordering of S and T. However, this is a contradiction. □

We say that a k-tuple $(i_1, i_2, \ldots, i_k)$ is smaller than another k-tuple $(j_1, j_2, \ldots, j_k)$, denoted by $(i_1, i_2, \ldots, i_k) < (j_1, j_2, \ldots, j_k)$, if $i_v < j_v$ and $i_u = j_u$ for all $1 \le u \le v$, where $1 \le v \le k$.

THEOREM 1. *Suppose that $G = (S, T, E)$ is a connected bipartite-permutation graph. Then, the ordering of S that can form a strong ordering with an ordering of T is unique provided the ordering of A is specified. Besides, this ordering, denoted by $s_1, s_2, \ldots, s_{|S|}$, can be uniquely defined by $\psi(s_1) < \psi(s_2) < \ldots < \psi(s_{|S|})$, where $\psi(s_i)$, $1 \le i \le |S|$, is defined as follows. (Let $s_{o(1)}, s_{o(2)}, \ldots, s_{o(|A|)}$ be the specified ordering of A that is embedded in the ordering of S.)*

CASE 1. $s_i \in A$.

Define $\psi(s_i) = (k, -|N(s_i)|, 0)$, where $s_i = s_{o(k)}$.

CASE 2. $s_i \in C$.

Define $\psi(s_i) = (k, -|N(s_{o(k)}) \cap N(s_i)|, |N(s_{o(k+1)}) \cap N(s_i)|)$, where $N(s_{o(k)}) \cap N(s_{o(k+1)}) \subseteq N(s_i)$.

CASE 3. $|A| \ge 2$, $s_i \in S-(A \cup C)$, and $N(s_i) \subset N(s_{o(1)})$.

Define $\psi(s_i) = (0, 0, |N(s_i)|)$.

CASE 4. $|A| \ge 2$, $s_i \in S-(A \cup C)$, and $N(s_i) \subset N(s_{o(|A|)})$.

Define $\psi(s_i) = (|A|+1, -|N(s_i)|, 0)$.

CASE 5. $|A| = 1$ and $s_i \in S-(A \cup C)$.

There exist at most two vertices $s_a, s_b \in S-(A \cup C)$ such that $N(s_a)$ and $N(s_b)$ are not a subset of each other, and for each $s_i \in S-(A \cup C)-\{s_a, s_b\}$, either $N(s_i) \subset N(s_a)$ or $N(s_i) \subset N(s_b)$ holds. For each $s_i \in S-(A \cup C)$, define $\psi(s_i) = (0, 0, |N(s_i)|)$ if $N(s_i) \subseteq N(s_a)$, and $\psi(s_i) = (2, -|N(s_i)|, 0)$ if $N(s_i) \subseteq N(s_b)$.

PROOF. Since G is a connected bipartite-permutation graph, there is a strong ordering of S and T by Lemma 1. Besides, by Lemma 2, the ordering of S (and therefore the ordering of A) has the adjacency property. In the following, we show that starting from an ordering of A with the adjacency property, the ordering of S that can form a strong ordering with an ordering of T is unique. Since the number of orderings of A with the adjacency property is only two and they are the reverse of each other (see Lemma 4), we can start from either, say $s_{o(1)}, s_{o(2)}, \ldots, s_{o(|A|)}$.

By Lemma 5, we know that for each $s_i \in C$, s_i must be placed between $s_{o(j)}$ and $s_{o(j+1)}$, where $N(s_{o(j)}) \cap N(s_{o(j+1)}) \subseteq N(s_i)$, in order to form a strong ordering of S and T. Since it is possible that multiple s_is are assigned to the same interval between $s_{o(j)}$ and $s_{o(j+1)}$, we must arrange these s_is properly so as to obtain a strong ordering. This can be easily done by the aid of Lemma 7, which lead to a unique ordering of s_is $\in C$.

As for the remaining vertices (in $S-(A \cup C)$), they must be placed before $s_{o(1)}$ or after $s_{o(|A|)}$, which is known by Lemma 3. If $|A| \ge 2$, by Lemmas 3 and 8, we can uniquely partition $S-(A \cup C)$ into two disjoint subsets $W_1 = \{s_{r(1)}, s_{r(2)}, \ldots, s_{r(u)}\}$ and $W_2 = \{s_{g(1)}, s_{g(2)}, \ldots, s_{g(v)}\}$, where $u+v = |S-(A \cup C)|$, such that $s_{r(1)} \subset s_{r(2)} \subset \ldots \subset s_{r(u)} \subset N(s_{o(1)})$ and $s_{g(1)} \subset s_{g(2)} \subset \ldots \subset s_{g(v)} \subset N(s_{o(|A|)})$. If $|A| = 1$, we partition $S-(A \cup C)$ into $W_1 = \{s_{r(1)}, s_{r(2)}, \ldots, s_{r(u)}\}$ and $W_2 = \{s_{g(1)}, s_{g(2)}, \ldots, s_{g(v)}\}$, where $u+v = |S-(A \cup C)|$, such that $N(s_{r(1)}) \subset N(s_{r(2)}) \subset \ldots \subset N(s_{r(u)})$ and $N(s_{g(1)}) \subset N(s_{g(2)}) \subset \ldots \subset N(s_{g(v)})$. The ordering of vertices in W_1 and the ordering of vertices in W_2 are uniquely defined. Thus, the ordering of S is uniquely determined in order to obtain a strong ordering. Moreover, it is not difficult to check that $\psi(s_1) < \psi(s_2) < \ldots < \psi(s_{|S|})$ uniquely defines this ordering. □

4 A PARALLEL RECOGNITION ALGORITHM

In this section, we present a parallel algorithm which can recognize a connected bipartite-permutation graph efficiently. The outline of the algorithm is described as follows. The algorithm first determines whether the input graph G is bipartite or not. If it is, the largest subset S' of S is found so that $N(s_x) \ne N(s_y)$ for all $s_x, s_y \in S$, $s_x \ne s_y$. The algorithm then computes $\psi(s_i)$s for all $s_i \in S'$, and use them to define an ordering of S'. Finally, the algorithm checks if the ordering of S' has the adjacency and enclosure properties. If yes, G is a bipartite-permutation graph. Otherwise, G is not a bipartite-permutation graph. A formal description of the algorithm is given below.

ALGORITHM 1. /* Suppose that the input graph G is connected*.

STEP 1. Determine whether G is a bipartite graph or not. Let $G = (S, T, E)$, if the answer is yes.

STEP 2. Find the largest subset S' of S such that $N(s_x) \ne N(s_y)$ for all $s_x, s_y \in S'$, $s_x \ne s_y$.

STEP 3. Find the ordering of S' that is uniquely defined by the function ψ.

- 3.1. Partition S' into subsets A, C and $S'-(A \cup C)$.
- 3.2. Find an ordering of A with the adjacency property. Let $s_{o(1)}, s_{o(2)}, \ldots, s_{o(|A|)}$ denote this ordering.
- 3.3. Compute $\psi(s_i)$s for all $s_i \in S'$.
- 3.4. Sort S' increasingly according to the values of $\psi(s_i)$s.

STEP 4. Determine whether G is a bipartite-permutation graph or not.

- 4.1. Determine whether the ordering of S' has the adjacency property or not.

4.2. Determine whether the ordering of S' has the enclosure property or not.

The correctness of Algorithm 1 is discussed below.

LEMMA 9. *Suppose that $G = (S \cup \{z\}, T, E)$ is a connected bipartite graph, where $z \notin S$ and $N(z) = N(s_i)$ for some $s_i \in S$. Then, $s_1, s_2, \ldots, s_{i-1}, s_i, s_{i+1}, \ldots, s_{|S|}, t_1, t_2, \ldots, t_{|T|}$ forms a strong ordering of S and T if and only if $s_1, s_2, \ldots, s_{i-1}, s_i, z, s_{i+1}, \ldots, s_{|S|}, t_1, t_2, \ldots, t_{|T|}$ or $s_1, s_2, \ldots, s_{i-1}, z, s_i, s_{i+1}, \ldots, s_{|S|}, t_1, t_2, \ldots, t_{|T|}$ forms a strong ordering of $S \cup \{z\}$ and T.*

PROOF. ($\Leftarrow$) It is trivial.

($\Rightarrow$)We only prove that $s_1, s_2, \ldots, s_{i-1}, s_i, z, s_{i+1}, \ldots, s_{|S|}, t_1, t_2, \ldots, t_{|T|}$ is a strong ordering of $S \cup \{z\}$ and T. For the other case, i.e., $s_1, s_2, \ldots, s_{i-1}, z, s_i, s_{i+1}, \ldots, s_{|S|}, t_1, t_2, \ldots, t_{|T|}$, the proof is similar.

Suppose that $s_1, s_2, \ldots, s_{i-1}, s_i, z, s_{i+1}, \ldots, s_{|S|}, t_1, t_2, \ldots, t_{|T|}$ is not a strong ordering of $S \cup \{z\}$ and T. Then, there exist (z, t_p), $(s_j, t_q) \in E$, where z and t_q precede (or succeed) s_j and t_p, respectively, such that $(z, t_q) \notin E$ or $(s_j, t_p) \notin E$.

Since $t_p \in N(z)$, we have $t_p \in N(s_i)$ and thus $(s_i, t_p) \in E$. If z and t_q precede s_j and t_p, respectively, then (s_i, t_p) and (s_j, t_q) cross, which implies (s_i, t_q), $(s_j, t_p) \in E$, because $s_1, s_2, \ldots, s_{i-1}, s_i, s_{i+1}, \ldots, s_{|S|}, t_1, t_2, \ldots, t_{|T|}$ is a strong ordering S and T. Thus, we have $t_q \in N(s_i) = N(z)$. So, (z, t_q), $(s_j, t_p) \in E$, which is a contradiction.

Similarly, we can show that the case of z and t_q succeeding s_j and t_p, respectively, also leads to a contradiction. This completes the proof. □

The following lemma is an immediate consequence of Lemma 9.

LEMMA 10. *Suppose that $G = (S \cup \{z_{p,q} \mid 1 \le p \le j \text{ and } 1 \le q \le k_p\}, T, E)$ is a connected bipartite graph, where $z_{p,q} \notin S$ and $N(z_{p,q}) = N(s_{i_p})$ for some $s_{i_p} \in S$, for $1 \le p \le j$ and $1 \le q \le k_p$. Then, $s_1, s_2 \ldots, s_{i_1}, \ldots, s_{i_2}, \ldots, s_{i_j}, \ldots, s_{|S|}, t_1, t_2, \ldots, t_{|T|}$ forms a strong ordering of S and T if and only if*

$$s_1, s_2, \ldots, s_{i_1}, z_{1,1}, z_{1,2}, \ldots, z_{1,k_1}, \ldots, s_{i_2}, z_{2,1}, z_{2,2}, \ldots, z_{2,k_2}, \ldots, s_{i_j}, z_{j,1}, z_{j,2}, \ldots, z_{j,k_j}, \ldots, s_{|s|}, t_1, t_2, \ldots, t_{|T|}$$

forms a strong ordering of $S \cup \{z_{p,q} \mid 1 \le p \le j \text{ and } 1 \le q \le k_p\}$ and T.

As an immediate consequence of Theorem 1 and Lemmas 1 and 10, it is sufficient to check whether or not the ordering of S' defined by the function ψ has the adjacency and enclosure properties, in order to determine if there exists a strong ordering of S and T. If the answers to Steps 4.1 and 4.2 are all affirmative, then G is a bipartite-permutation graph by Lemmas 1 and 10. Otherwise, G is not a bipartite-permutation graph for the reason below.

Suppose to the contrary that at least one of the answers to Steps 4.1 and 4.2 is negative and G is a bipartite-permutation graph. Then, Theorem 1 assures the existence of a unique ordering of S' which contains the ordering of A found at Step 3.2 and forms a strong ordering with an ordering of T. Besides, the ordering of S' is uniquely defined by the function ψ. By Lemma 2, the ordering of S' has the adjacency and enclosure properties. However, this is a contradiction.

Next, the corresponding permutation π of G is constructed as follows, if G is a bipartite-permutation graph.

STEP 5. Construct the corresponding permutation π of G, if the answer to Step 4 is affirmative.

5.1. Define $\psi(s_i) = \psi(s_j)$ for all $s_i \in S - S'$, where $N(s_i) = N(s_j)$ for some $s_j \in S'$, and sort S nondecreasingly according to the values of $\psi(s_k)$s, all $s_k \in S$. Suppose that $s_1, s_2, \ldots, s_{|S|}$ is the sorted sequence. Also, sort T nondecreasingly according to the values of $\chi(t_k)$s, all $t_k \in T$, where $\chi(t_k) = (p(t_k), q(t_k))$. Suppose that $t_1, t_2, \ldots, t_{|T|}$ is the sorted sequence.

5.2. Sort $S \cup T$ increasingly according to the weights of s_is and t_js, all $s_i \in S$ and all $t_j \in T$, where the weight of s_i is defined as i and the weight of t_j is defined as $p(t_j) - 1/(j+1)$. Suppose that $u_1, u_2, \ldots, u_{|S \cup T|}$ is the sorted sequence.

5.3. Sort $u_1, u_2, \ldots, u_{|S \cup T|}$ increasingly according to their weights, where the weight of u_i, $1 \le i \le |S \cup T|$, is defined as k if $u_i = s_k$ for some k, $1 \le k \le |S|$, and $q(t_j) + 1 - 1/(j+1)$ if $u_i = t_j$ for some j, $1 \le j \le |T|$. Suppose $u_{o(1)}, u_{o(2)}, \ldots, u_{o(|S \cup T|)}$ is the sorted sequence. Then, $\pi = (o(1), o(2), \ldots, o(|S \cup T|))$.

The ordering of T obtained at Step 5.1 is constructed based on Lemma 6 (exchanging S and T), which states that the ordering of T must be defined so that $p(t_i) \le p(t_j)$ and $q(t_i) \le q(t_j)$ for any two vertices $t_i, t_j \in T$, $i < j$, in order to obtain a strong ordering of S and T. The ordering of S and the ordering of T obtained at Step 5.1 form a strong ordering.

The sorted sequence obtained at Step 5.2 satisfies that s_i precedes s_j if and only if $i < j$; t_i precedes t_j if and only if $p(t_i) < p(t_j)$ or $p(t_i) = p(t_j)$ and $i < j$; t_j precedes s_i if and only if $i \le p(t_j)$. The sorted sequence obtained at Step 5.3 satisfies that s_i precedes s_j if and only if $i < j$; t_i precedes t_j if and only if $q(t_i) < q(t_j)$ or $q(t_i) = q(t_j)$ and $i < j$; t_j succeeds s_i if and only if $i \ge q(t_j)$. Thus, the permutation π obtained at Step 5.3 satisfies that for all $i, j \in \{1, 2, \ldots, |S \cup T|\}$, $((i-j)(\pi^{-1}(i) - \pi^{-1}(j)) < 0$ if and only if u_i and u_j are adjacent in G.

Suppose that the input graph G is represented by an $n \times n$ adjacency matrix, where n is the number of vertices in G. Before analyzing the complexity of Algorithm 1, let us consider the following problems: 1) computing $|N(s_i)|$, 2) computing $N(s_i) \cap N(s_j)$, 3) determining if $N(s_i) = N(s_j)$, 4) determining if $N(s_i) \subseteq N(s_j)$, and 5) finding the maximum (or minimum) of a set of n values. It is clear that all these problems can be computed in $O(\log n)$ time using $O(n/\log n)$ processors on the EREW PRAM (Exclusive-Read Exclusive-Write Parallel Random Access Machine).

Now, the complexity of Algorithm 1 is analyzed as follows. Step 1 can be completed in $O(\log n)$ time using $O(m + n)$ processors on the CRCW PRAM or $O(\log^2 n)$ time using $O(n^2/\log^2 n)$ processors on the CREW PRAM. To begin with, a spanning tree of G is constructed (an arbitrary vertex is selected as the root), and the level number of each vertex in the tree is determined. Then, the vertex set is partitioned into two disjoint subsets depending on whether the level numbers of vertices are even or odd. Finally, it is determined if the two end vertices of each edge in G belong to different subsets. Finding a spanning tree of G can be completed in $O(\log n)$ time using $O(m + n)$ processors on the CRCW PRAM [23] or $O(\log^2 n)$ time using $O(n^2/\log^2 n)$ processors on the CREW PRAM [7]. Computing the level numbers of vertices of a tree takes $O(\log n)$ time if $O(n)$ processors are used under the CREW PRAM model [23]. The remaining work can be completed in $O(1)$ time using $O(m)$ processors on the CREW PRAM.

Step 2 can be completed in $O(\log n)$ time using $O(n^3/\log n)$ processors on the CRCW PRAM or $O(\log^2 n)$ time using $O(n^3/\log^2 n)$ processors on the CREW PRAM. First, $O(\log n)$ time is sufficient to determine if $N(s_i) = N(s_j)$ for all $s_i, s_j \in S$, if $O(n^3/\log n)$ processors are used under the CREW PRAM model. Then, the set S' can be determined by executing Shiloach and Vishkin's connected component algorithm [18], which takes $O(\log n)$ time using $O(m + n)$ processors on the CRCW PRAM, or Chin, Lam and Chen's connected component algorithm [7], which takes $O(\log^2 n)$ time using $O(n^2/\log^2 n)$ processors on the CREW PRAM.

Step 3.1 can be computed in $O(\log n)$ time using $O(n^3/\log n)$ processors on the CREW PRAM. At first, a directed graph is constructed; S' is the vertex set and there is a directed edge from s_i to s_j if $N(s_i) \subset N(s_j)$, where $s_i, s_j \in S'$ and $s_i \neq s_j$. Then, the sets A and C can be easily identified in $O(\log n)$ time by the aid of the directed graph, if $O(n^3/\log n)$ processors are used under the CREW PRAM model.

Step 3.2 can be computed in $O(\log n)$ time using $O(n^3/\log n)$ processors on the CRCW PRAM or $O(\log^2 n)$ time using $O(n^3/\log^2 n)$ processors on the CREW PRAM. Suppose that G is a bipartite-permutation graph. Then, if $s_{o(1)}, s_{o(2)}, \ldots, s_{o(|A|)}$ is an ordering of A with the adjacency property, then for each $s_{o(i)} \in A$, we have $N(s_{o(1)}) \cap N(s_{o(i)}) \subset N(s_{o(2)}) \cap N(s_{o(i)}) \subset \ldots \subset N(s_{o(i-1)}) \cap N(s_{o(i)})$, $N(s_{o(|A|)}) \cap N(s_{o(i)}) \subset N(s_{o(|A|-1)}) \cap N(s_{o(i)}) \subset \ldots \subset N(s_{o(i+1)}) \cap N(s_{o(i)})$, and $N(s_{o(i-1)}) \cap N(s_{o(i)})$ and $N(s_{o(i+1)}) \cap N(s_{o(i)})$ do not contain each other (this becomes clearer when we observe the corresponding rectilinear polygon). Thus, if G is a bipartite-permutation graph, we can find an ordering of A with the adjacency property as follows. For each $s_{o(i)} \in A$, we first find a vertex, say $s_{o(p)}$, such that $s_{o(p)} \in A - \{s_{o(i)}\}$ and $|N(s_{o(p)}) \cap N(s_{o(i)})|$ is maximal, and then find another vertex, say $s_{o(q)}$, such that $s_{o(q)} \in A'$ and $|N(s_{o(q)}) \cap N(s_{o(i)})|$ is maximal, where $A' = A - (\{s_{o(i)}\} \cup \{s_{o(j)} \mid s_{o(j)} \in A \text{ and } N(s_{o(j)}) \cap N(s_{o(i)}) \subseteq N(s_{o(p)}) \cap N(s_{o(i)})\})$. The vertices $s_{o(p)}$ and $s_{o(q)}$ can be determined in $O(\log n)$ time using $O(n^3/\log n)$ processors on the CREW PRAM. Note that one of $s_{o(p)}$ and $s_{o(q)}$ is the immediate predecessor of $s_{o(i)}$ and the other is the immediate successor of $s_{o(i)}$ in any ordering of A with the adjacency property (see Lemma 4).

Next, we construct two edges $(s_{o(i)}, s_{o(p)})$ and $(s_{o(i)}, s_{o(q)})$ for each $s_{o(i)} \in A$, and thus an undirected path containing all the vertices of A is formed if G is a connected bipartite-permutation graph. We can detect whether such a path exists or not by executing a connected component algorithm and checking the degree of each vertex. As we have mentioned above, the former can be completed in $O(\log n)$ time using $O(m + n)$ processors on the CRCW PRAM, or $O(\log^2 n)$ time using $O(n^2/\log^2 n)$ processors on the CREW PRAM. The latter requires $O(\log n)$ time using $O(n)$ processors on the EREW PRAM if sorting is performed on edges. The sorting takes $O(\log n)$ time if $O(n)$ processors on the EREW PRAM is used [8].

To gain an ordering of A, we must transform the undirected path into a linked list, and then perform a list ranking operation on the list. On the CREW PRAM, the former takes $O(\log n)$ time using $O(n)$ processors if Vishkin's algorithm [23] is applied, and the latter takes $O(\log n)$ time using $O(n/\log n)$ processors if Cole and Vishkin's algorithm [9] is applied. The remaining work is to check if the found ordering of A, denoted by $s_{o(1)}, s_{o(2)}, \ldots, s_{o(|A|)}$, has the adjacency property. This can be easily done in $O(\log n)$ time using $O(n^2/\log n)$ processors on the CREW PRAM, by checking if $\max\{j \mid s_{o(j)} \in N(t_i)\} - \min\{j \mid s_{o(j)} \in N(t_i)\} = |\{j \mid s_{o(j)} \in N(t_i)\}| - 1$, for all $t_i \in T$.

The bottleneck of Step 3.3 is to determine k satisfying $N(s_{o(k)}) \cap N(s_{o(k+1)}) \subseteq N(s_i)$ (i.e., Case 2), which requires $O(\log n)$ time using $O(n^2/\log n)$ processors on the CREW PRAM. Thus, Step 3.3 can be computed in $O(\log n)$ time using $O(n^3/\log n)$ processors on the CREW PRAM.

Step 3.4 takes $O(\log n)$ time if $O(n)$ processors are used under the EREW PRAM model [8].

In total, Step 3 requires $O(\log n)$ time using $O(n^3/\log n)$ processors on the CRCW PRAM or $O(\log^2 n)$ time using $O(n^3/\log^2 n)$ processors on the CREW PRAM.

Step 4 can be completed in $O(\log n)$ time using $O(n^2/\log n)$ processors on the CREW PRAM. As we have mentioned above, Step 4.1 can be performed in $O(\log n)$ time using $O(n^2/\log n)$ processors on the CREW PRAM. If the ordering of S', denoted by $s_{o(1)}, s_{o(2)}, \ldots, s_{o(|S|)}$, has the adjacency property, then we carry out Step 4.2 by checking if there exist $t_i, t_j \in T$ such that $\min\{k \mid s_{o(k)} \in N(t_i)\} < \min\{k \mid s_{o(k)} \in N(t_j)\} \leq \max\{k \mid s_{o(k)} \in N(t_j)\} < \max\{k \mid s_{o(k)} \in N(t_i)\}$. If no such t_i, t_j exist, then S' has the enclosure property. Otherwise, S' does not have the enclosure property. So, Step 4.2 can be performed in $O(\log n)$ time using $O(n^2/\log n)$ processors on the CREW PRAM.

Since Steps 5.1, 5.2, 5.3 each require $O(\log n)$ time using $O(n^2/\log n)$ processors on the CREW PRAM, Step 5 can be executed with the same time complexity and processor complexity.

Based on the above discussion, we know that Algorithm 1 can be executed in $O(\log n)$ time using $O(n^3/\log n)$ processors on the CRCW PRAM, or $O(\log^2 n)$ time using $O(n^3/\log^2 n)$ processors on the CREW PRAM. Therefore, we have the following theorem, which is the main result of this paper.

THEOREM 2. *A connected bipartite-permutation graph can be recognized in $O(\log n)$ time using $O(n^3/\log n)$ processors on the CRCW PRAM, or $O(\log^2 n)$ time using $O(n^3/\log^2 n)$ processors on the CREW PRAM. Besides, the corresponding permutation π of the bipartite-permutation graph is constructed with the same time complexity and processor complexity.*

In [5], Chen gave a parallel algorithm that tests isomorphism of two bipartite-permutation graphs. Chen's algorithm runs in $O(\log^2 n)$ time using $O(n^3)$ processors on the CRCW PRAM. The bottleneck of Chen's algorithm is to find out a strong ordering of a bipartite-permutation graph. Chen solved the problem by directly applying the result of [6], which needs $O(\log^2 n)$ time using $O(n^3)$ processors on the CRCW PRAM. If a better result, for example Algorithm 1 in this paper, is applied, Chen's result will be improved. Therefore, combining Chen's algorithm and Algorithm 1, we have the following result.

THEOREM 3. *The problem of testing isomorphism of two bipartite-permutation graphs can be solved in* $O(\log n)$ *time using* $O(n^3/\log n)$ *processors on the* CRCW PRAM, *or* $O(\log^2 n)$ *time using* $O(n^3/\log^2 n)$ *processors on the* CREW PRAM.

5 Discussion and Conclusion

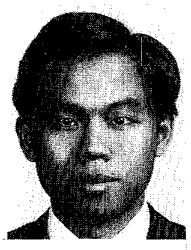
In this paper, we have proposed a parallel algorithm for recognizing a connected bipartite- permutation graph. In fact, the restriction to a connected graph can be released. If the input graph is not connected, we simply execute the proposed algorithm for each of its components. The input graph is a bipartite-permutation graph if each of its components is a bipartite-permutation graph. Also, the corresponding permutation of the input graph can be obtained by merging the corresponding permutations of its components. So, the time complexity and processor complexity required remain the same for a disconnected input graph.

Acknowledgment

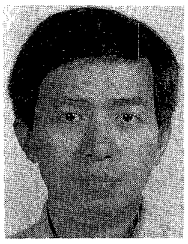
This work was supported by the National Science Council under Contracts NSC 83-0408-E002-018 and NSC 84-0408-E002-004, Taiwan, R.O.C.

References

[1] M.J. Atallah, G.K. Manacher, and J. Urrutia, "Finding a minimum independent dominating set in a permutation graph," Discrete Applied Math., vol. 21, pp. 177-183, 1988.
[2] A. Brandstädt and D. Kratsch, "On domination problems for permutation and other graphs," Theoretical Computer Science, vol. 54, pp. 181-198, 1987.
[3] A. Brandstädt, J. Spinrad, and L. Stewart, "Bipartite permutation graphs are bipartite tolerance graphs," Congressus Numerantium, vol. 58, pp. 165-174, 1987.
[4] L. Chen, "Logarithmic time NC algorithms for comparability graphs and circle graphs," Lecture Notes in Computer Science: Advances in Computing and Information, vol. 497, pp. 383-394, 1991.
[5] L. Chen, "An efficient parallel algorithm for testing isomorphism of bipartite permutation graphs," Proc. First Ann. IEEE Symp. Parallel and Distributed Processing, pp. 24-25, 1989.
[6] L. Chen and Y. Yesha, "Efficient parallel algorithms for bipartite permutation graphs," Networks, vol. 22, no. 1, pp. 29-39, 1993.
[7] F.Y. Chin, J. Lam, and I. Chen, "Efficient parallel algorithms for some graph problems," Comm. ACM, vol. 25, no. 9, pp. 659-665, 1982.
[8] R. Cole, "Parallel merge sort," SIAM J. Computing, vol. 17, no. 4, pp. 770-785, 1988.
[9] R. Cole and U. Vishkin, "Approximate parallel scheduling, part 1: The basic technique with applications to optimal parallel list ranking in logarithmic time," SIAM J. Computing, vol. 17, no. 1, pp. 128-142.
[10] M. Farber and J.M. Keil, "Domination in permutation graphs," J. Algorithms, vol. 6, pp. 309-321, 1985.
[11] M.R. Garey and D.S. Johnson, Computers and Intractability: a Guide to the Theory of NP-Completeness, San Francisco, Calif: Freeman, 1979.
[12] M.C. Golumbic, Algorithmic Graph Theory and Perfect Graphs, New York: Academic Press, 1980.
[13] D. Helmbold and E. Mayr, "Perfect graphs and parallel algorithms," Proc. Int'l Conf. Parallel Processing, pp. 853-860, 1986.
[14] D.E. Knuth, "The Art of Computer Programming," vol. 1, Reading, Mass: Addison Wesley, 1968.
[15] D. Kozen, U.V. Vazirani, and V.V. Vazirani, "NC algorithms for comparability graphs, interval graphs, and testing for unique perfect matching," Proc. Fifth Conf. Foundation of Software Technology and Theoretical Computer Science, New Delhi, pp. 496-503, 1985.
[16] C.L. Liu, Introduction to Combinatorial Mathematics. McGraw-Hill, 1968.
[17] A. Pnueli, A. Lempel, and S. Even, "Transitive orientation of graphs and identification of permutation graphs," Can. J. Math., vol. 23, no. 1, pp. 160-175, 1971.
[18] Y. Shiloach and U. Vishkin, "An O(log n) parallel connectivity algorithm," J. Algorithms, vol. 3, pp. 57-67, 1982.
[19] J. Spinrad, "On comparability and permutation graphs," SIAM J. Computing, vol. 14, no. 3, pp. 658-670, 1985.
[20] J. Spinrad, A. Brandstädt, and L. Stewart, "Bipartite permutation graphs," Discrete Applied Math., vol. 18, pp. 279-292, 1987.
[21] K.J. Supowit, "Decomposing a set of points into chains, with applications to permutation and circle graphs," Information Processing Letters, vol. 21, pp. 249-252, 1985.
[22] K.H. Tsai and W.L. Hsu, "Fast algorithms for the dominating set problem on permutation graphs," Lecture Notes in Computer Science: Algorithms, vol. 450, pp. 109-117, 1990.
[23] U. Vishkin, "On efficient parallel strong orientation," Information Processing Letters, vol. 20, pp. 235-240, 1985.
[24] C.W. Yu, "Recognition problems of bipartite-permutation and other related graphs," PhD dissertation, Dept. Computer Science & Information Enginering, National Taiwan Univ., Taipei, Taiwan, Jun. 1993.
[25] C.W. Yu and G.H. Chen, "Parallel algorithms for permutaion graphs," Tech. Report 91-11, Dept. Computer Science and Information Engineering, National Taiwan Univ., Taipei, Taiwan, July 1991.
[26] C.W. Yu and G.H. Chen, "The weighted maximum independent set problem in permutaion graphs," Tech. Report 91-12, Dept. Computer Science and Information Engineering, National Taiwan Univ., Taipei, Taiwan, July 1991.



Chang-Wu Yu received the BS degree in computer science from Soochow University in June 1987, and the MS degree in computer science from National Tsing Hua University in June 1989, all in Taiwan. In June 1993 he received the PhD degree in computer science from National Taiwan University. His current research interests include graph algorithms, parallel computing, and computational geometry.



Gen-Huey Chen received the BS degree in computer science from National Taiwan University in June 1981, and the MS and PhD degrees in computer science from National Tsing Hua University, Taiwan, in June 1983 and January 1987, respectively. He joined the faculty of the Department of Computer Science and Information Engineering, National Taiwan University, in February 1987, and has been a professor since August 1992. Dr. Chen received the Distinguished Research Award from the National Science Council, Taiwan, in 1993. His current research interests include graph-theoretic interconnection networks, parallel and distributed computing, and design and analysis of algorithms.