ELSEVIER

Theoretical Computer Science 147 (1995) 249–265

Theoretical Computer Science

Efficient parallel algorithms for doubly convex-bipartite graphs☆

Chang-Wu Yu, Gen-Huey Chen*

Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan

Received August 1992; revised May 1994
Communicated by M.S. Paterson

Abstract

Suppose that $G = (S, T, E)$ is a bipartite graph. An ordering of $S(T)$ has the adjacency property if for each vertex in $T(S)$, its adjacent vertices in $S(T)$ are consecutive in the ordering. If there exist orderings of S and T which have the adjacency property, G is called a doubly convex-bipartite graph. In this paper, a parallel algorithm is proposed to recognize a doubly convex-bipartite graph. The algorithm runs 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.

1. Introduction

An undirected graph G with vertices $\{v_1, v_2, \ldots, v_n\}$ is called a *permutation graph* [10] if there exists a permutation π on $N = \{1, 2, \ldots, n\}$ such that for all $i, j \in N$,

$$(i - j)(\pi^{-1}(i) - \pi^{-1}(j)) < 0$$

if and only if v_i and v_j are joined by an edge in G. Pictorially, draw the vertices $v_1, v_2, \ldots, v_n$ in order on a line, and $v_{\pi(1)}, v_{\pi(2)}, \ldots, v_{\pi(n)}$ 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.

An undirected graph G is called a *circle graph* [10] if there exists a set C of chords on a circle and a 1–1 correspondence between C and the set of vertices of G such that two vertices are adjacent in G if and only if their corresponding chords intersect. The

☆ This research is supported by the National Science Council of the Republic of China with the grant NSC82-0408-E-002-409.

* Corresponding author. E-mail: ghchen@csie.ntu.edu.tw.

SSDI 304-3975(94)00220-7

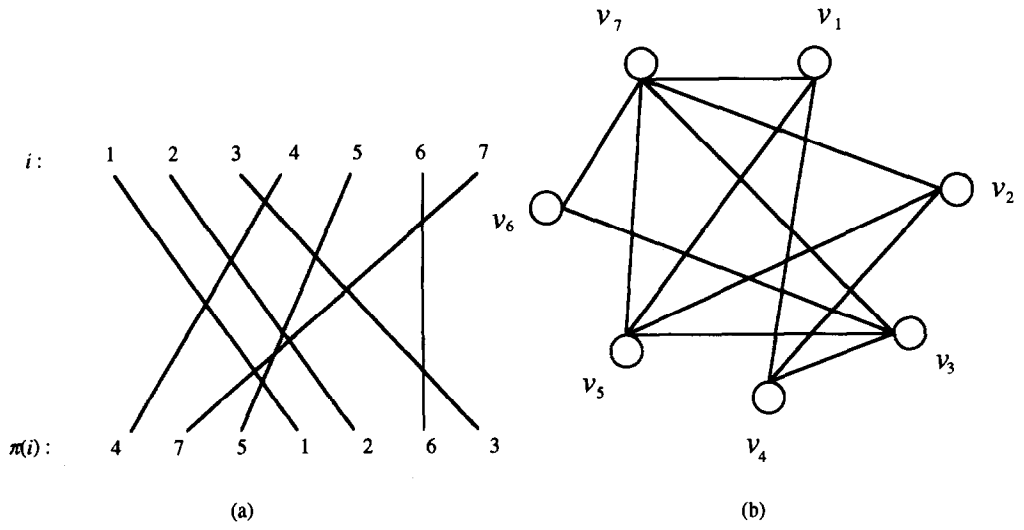

Fig. 1. An example. (a) A permutation π. (b) Its corresponding permutation graph.

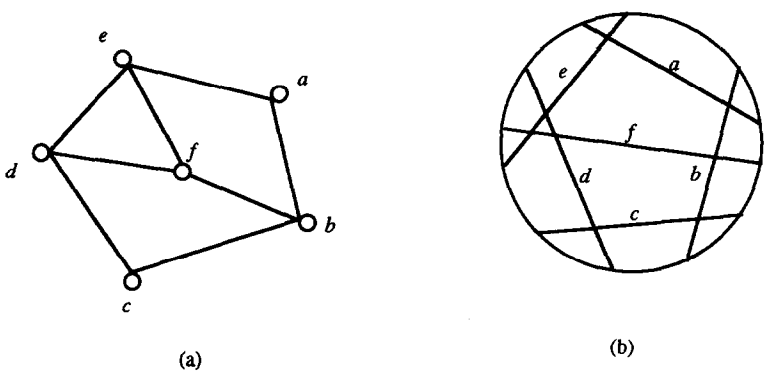

Fig. 2. An example. (a) A circle graph. (b) One of its circle-graph models.

set C is called a *circle-graph model* for G. Fig. 2 shows a circle graph along with one of its circle-graph models.

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*. A circle graph which is also bipartite is called a *bipartite-circle graph*.

Let $G = (S, T, E)$ denote a bipartite graph, where $S \cup T$ is the set of vertices and E is the set of edges. 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 an ordering. An ordering of $S(T)$ has the *enclosure property* if for every pair of vertices $t_1, t_2 \in T(S)$ satisfying $N(t_1) \subseteq N(t_2), N(t_2) -$

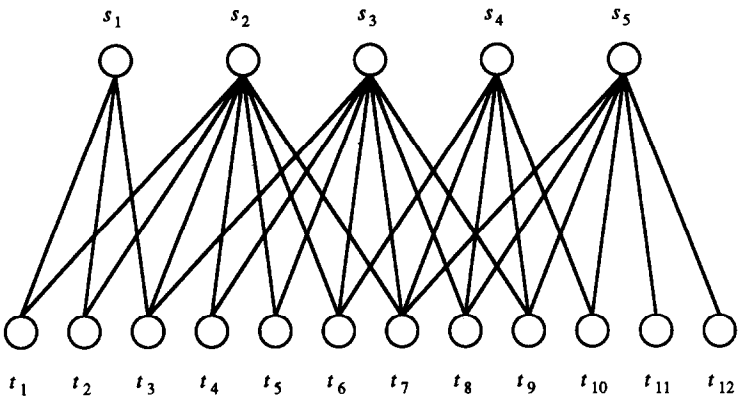

Fig. 3. An ordering of S and an ordering of T which have the adjacency and the enclosure properties.

$N(t_1)$ contains consecutive vertices in the ordering. See Fig. 3, where an example is shown. 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. They form a strong ordering if for all $s_i, s_j \in S$, $t_k, t_l \in T$ ($i < j$ and $k < l$), (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. 3 forms a strong ordering.

The graph $G = (S, T, E)$ is called a *convex-bipartite graph* if there are orderings of S or T having the adjacency property, and a *doubly convex-bipartite graph* if there are orderings of S and T having the adjacency property [12]. See Fig. 4, where a doubly convex-bipartite graph is shown.

Glover [9] showed a practically important application of doubly convex-bipartite graphs in industry. Lipski and Preparata [12] solved the maximum matching problem on both doubly convex-bipartite graphs and convex-bipartite graphs. Dekel and Sahni [6] developed an efficient parallel algorithm that finds maximum matchings in convex-bipartite graphs. Their result may also be used to solve several scheduling problems.

As we will see later in this paper, bipartite-permutation graphs are a subclass of doubly convex-bipartite graphs, which are a subclass of bipartite-circle graphs. Spinrad et al. [14] presented an $O(m + n)$ time recognition algorithm for bipartite-permutation graphs, where n and m are numbers of vertices and edges, respectively. Chen and Yesha [3] 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). Recently, Yu and Chen [16] improved the work of Chen and Yesha by presenting a parallel algorithm that runs in $O(\log n)$ time using $O(n^3/\log n)$ processors on the CRCW PRAM or

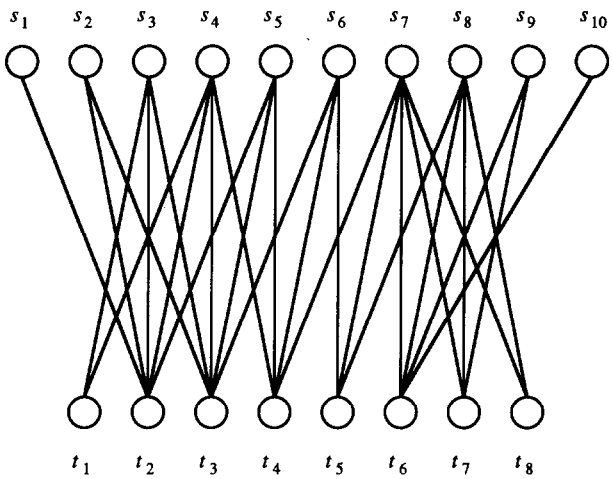

Fig. 4. A doubly convex-bipartite graph.

$O(\log^2 n)$ time using $O(n^3/\log^2 n)$ processors on the CREW PRAM (concurrent-read exclusive-write parallel random access machine).

Testing for the consecutive 1's property for a (0, 1)-matrix is an important problem. Given a (0, 1)-matrix, this problem asks whether it is possible to permute its columns so that there are consecutive 1's in each of its rows. If it is so, we are also interested in knowing how the columns should be permuted. The problem has applications in many fields such as genetics, archaeology, information science, operations research, and so on [2]. Klein and Reif [11] show that this problem can be solved in $O(\log^3 n)$ time using $O(n^2)$ processors on the CREW PRAM, as a result of parallel algorithms for *PQ*-trees. Recently, Chen and Yesha [2] further reduced the time complexity to $O(\log^2 n)$ time by using $O(n^3)$ processors on the CRCW PRAM. With slight modifications, the algorithm of Chen and Yesha can recognize convex-bipartite graphs (and therefore doubly convex-bipartite graphs) with the same time complexity and processor complexity.

In this paper, we show that for the purpose of recognizing doubly convex-bipartite graphs, there exist faster algorithms requiring fewer processors than that of Chen and Yesha. We present a parallel algorithm that recognizes a doubly convex-bipartite 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. Compared with the result of Chen and Yesha [2], our algorithms need less time and fewer processors on the same computation model, or the same time but fewer processors on a weaker model.

The rest of this paper is organized as follows. In the next section, we introduce some properties about doubly convex-bipartite graphs, which form the basis of our algorithms. In Section 3, we present a parallel algorithm for recognizing doubly convex-bipartite graphs. In Section 4, we conclude the paper with some final remarks.

2. Properties

Before introducing properties of doubly convex-bipartite graphs, we first give some necessary definitions and notations.

The *complement* of a graph G is a graph having the same vertex set as G, whose any two vertices are adjacent if and only if they are not adjacent in G. For a bipartite graph $G = (S, T, E)$, we define a graph $G^*_{S,T}$ whose vertex set is S and whose edge set is defined as follows: for all $s_i, s_j \in S$, (s_i, s_j) is an edge of $G^*_{S,T}$ if and only if $N(s_i) \subseteq N(s_j)$ in G. 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 that have both ends in $U \cup V$ is called the subgraph of G *induced by* U and V, and is denoted by $G_{U,V}$.

Suppoose that $G = (S, T, E)$ is a bipartite graph, and $s_1, s_2, \ldots, s_{|S|}$ and $t_1, t_2, \ldots, t_{|T|}$ are orderings of S and T, respectively, having the adjacency property. Then G can be expressed as a rectilinear polygon: we only need to place a square at the position (t_i, s_j) for each edge $(s_j, t_i) \in E$. The polygon has a shape like Fig. 5 or its reflection with respect to a vertical line. Without loss of generality, we consider the polygon looks like Fig. 5 in the rest of this paper. As we will see later, many of the properties introduced in this section become clearer by the aid of the polygon.

As a consequence of the adjacency properties of S and T, there are two structural properties about the polygon: (1) each row (column) of the polygon contains a

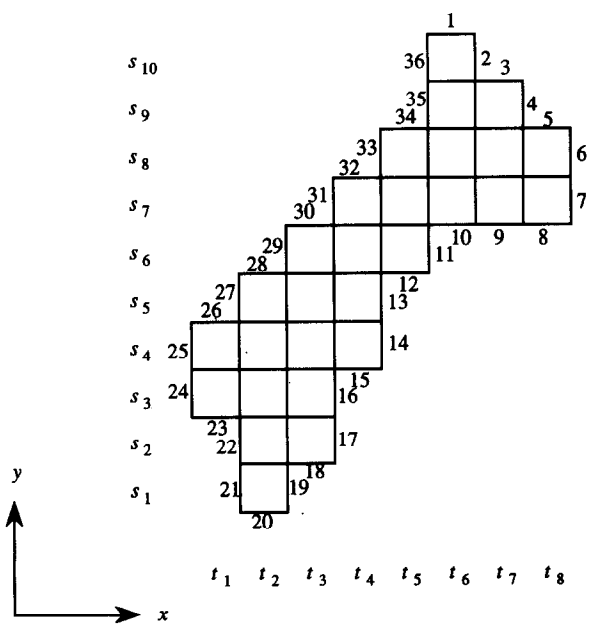

Fig. 5. The rectilinear polygon constructed from the doubly convex-bipartite graph of Fig. 4.

continuous block of squares; (2) the right (left) boundary of the polygon consists of two portions: one is nondecreasing and the other is nonincreasing from bottom to top. Intuitively, each row and each column of the polygon represent a vertex in S and a vertex in T, respectively, and the specified square represents an edge connecting the two vertices.

A row is *maximal* if there exists no row that can properly cover it. For example, rows s_4, s_6, s_7 in Fig. 5 are maximal. A row is *right maximal* (*left maximal*) if it is maximal and has the greatest (smallest) x-coordinate. If more than one candidate exists, select the uppermost (lowest) one as the right (left) maximal row. For example, row s_7 is right maximal and row s_4 is left maximal.

The polygon can be partitioned into three parts: bottom region (R_B), middle region (R_M), and top region (R_T). The middle region consists of three parts: upper part, center part, and lower part, which are separated by the right and left maximal rows. The upper part contains those rows above the right maximal row whose leftmost squares have the same x-coordinate as the right maximal row. The lower part contains those rows below the left maximal row whose rightmost squares have the same x-coordinate as the left maximal row. The right maximal row, the left maximal row, and those between them constitute the center part. For example, see Fig. 5 where the center part contains rows s_4, s_5, s_6, s_7. No upper part and lower part exists in this example. The rows above the middle region constitute the top region, and the rows below the middle region constitute the bottom region. See Fig. 5 where the top region contains rows s_8, s_9, s_{10} and the bottom region contains rows s_1, s_2, s_3.

In the following, we introduce sets A and C of rows which will be mentioned very often in the subsequent discussion. The set A contains all the maximal rows. The set C contains the rows of the middle region with the removal of maximal rows.

Definition 1. 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. $p(s_i) = \min\{j \mid t_j \in N(s_i)\}$. $q(s_i) = \max\{j \mid t_j \in N(s_i)\}$. $A = \{s_i \mid s_i \in S$ 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 $N(s_j) \cap N(s_k) \subseteq N(s_i)$; (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 $N(s_k)\}$.

For example, $A = \{s_4, s_6, s_7\}$, $B = \{s_1, s_2, s_3, s_5, s_8, s_9, s_{10}\}$, and $C = \{s_5\}$ for Fig. 5. A further explanation of sets A, B, C is given in Fig. 6. It also provides a further explanation of the top, middle, and bottom regions.

Throughout this section, unless mentioned particularly, we assume that $G = (S, T, E)$ is connected and $N(s_i) \neq N(s_j)$ for all $s_i, s_j \in S$, $s_i \neq s_j$. The following two lemmas state basic properties of the rectilinear polygon. Since they are clear by observing the rectilinear polygon, their proof is omitted.

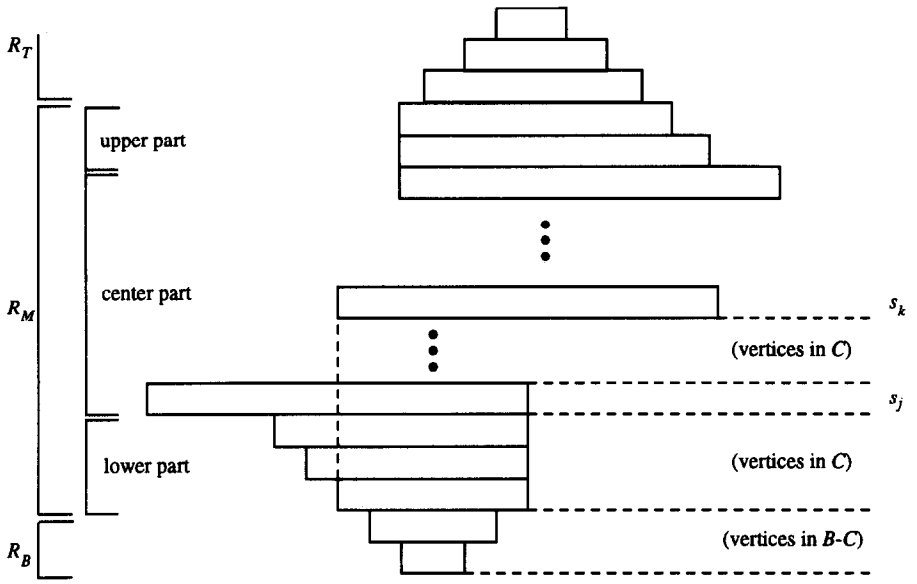

Fig. 6. A further explanation of sets A, B, C, where s_j (s_k) is the first (second) vertex in A.

Lemma 1. *Suppose that $G = (S, T, E)$ is a doubly convex-bipartite graph. Then, $R_M = A \cup C$.*

Lemma 2. *Suppose that $G = (S, T, E)$ is a doubly convex-bipartite graph. Then, for all s_i, s_j, where $s_i, s_j \in R_T$ or $s_i, s_j \in R_B$, we have $N(s_i) \subseteq N(s_j)$ or $N(s_j) \subseteq N(s_i)$.*

Lemma 3. *If $G = (S, T, E)$ is a doubly convex-bipartite graph, then G is a bipartite-circle graph.*

Proof. We show that G is a bipartite-circle graph by constructing one of its circle-graph models. Suppose that $s_1, s_2, \ldots, s_{|S|}$ and $t_1, t_2, \ldots, t_{|T|}$ are orderings of S and T, respectively, which have the adjacency property. We first construct a rectilinear polygon from G by placing a square at the point (t_i, s_j) for each edge $(s_j, t_i) \in E$. The rectilinear polygon has a shape like Fig. 5 or its reflection with respect to a vertical line. Then, we traverse the boundary of the polygon clockwise (starting at any edge), and label the edges with numbers from 1 to $2(|S| + |T|)$. Finally, we arrange these numbers around a circle, and generate a chord joining i and j if they are on the same row or on the same column in the rectilinear polygon (see Fig. 7). It is easy to see that the set of chords represents a circle-graph model for G. So, G is a bipartite-circle graph. □

The following lemma is straightforward as a consequence of Definition 1.

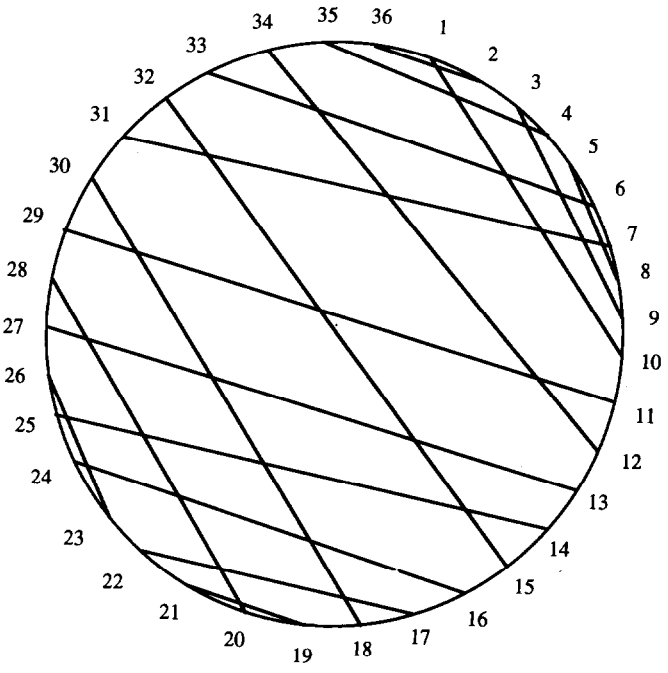

Fig. 7. A circle-graph model for the double convex-bipartite graph of Fig. 4.

Lemma 4. *For each vertex $s_i \in B$, there exists a vertex $s_j \in A$ such that $N(s_i)$ is a proper subset of $N(s_j)$.*

Lemma 5. *Suppose that $G = (S, T, E)$ is a bipartite graph. Then, $G_{A,T}$ is connected.*

Proof. Suppose that $G_{A,T}$ is disconnected and contains $r > 1$ connected components. Therefore, T can be partitioned into $T_1, T_2, \ldots, T_r$ each of which is contained in one component of $G_{A,T}$. Then for each vertex $s_i \in B$, we have $N(s_i) \subseteq T_k$ for some k, $1 \leqslant k \leqslant r$ (if this is not true, then there exists a vertex $s_i \in B$ such that $N(s_i)$ is distributed over at least two subsets of T, which is in contradiction to Lemma 4). However, this implies that G is also disconnected, which is a contradiction. □

Lemma 6 (Spinrad et al. [14, Theorem 1]). *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.*

Note that the ordering of S and the ordering of T, which form a strong ordering, mentioned in statement (2) of Lemma 6 have the adjacency and enclosure properties,

provided G is connected. This can be seen from the proof of Theorem 1 in [14] and is stated as the following lemma.

Lemma 7. *Suppose that $G = (S, T, E)$ is a 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.*

Lemmas 6 and 7 imply that bipartite-permutation graphs are a subclass of doubly convex-bipartite graphs.

Lemma 8. *Suppose that $G = (S, T, E)$ is a doubly convex-bipartite graph. Then $G_{A,T}$ is a connected bipartite-permutation graph.*

Proof. The connectivity of $G_{A,T}$ is assured by Lemma 5. By Lemma 6, it is sufficient to show that there exists an ordering of T having the adjacency and enclosure properties. The existence of an ordering of T having the adjacency property is assured by the fact that G is a doubly convex-bipartite graph. On the other hand, any ordering of T has the enclosure property with respect to A, because no two vertices $s_i, s_j \in A$ exist so that $N(s_i) \subset N(s_j)$ or $N(s_j) \subset N(s_i)$. Therefore, $G_{A,T}$ is a bipartite-permutation graph. □

Lemma 9. *Suppose that $G = (S, T, E)$ is a doubly convex-bipartite graph. Then, $G_{A \cup C,T}$ is a connected bipartite-permutation graph.*

Proof. Since $G_{A,T}$ is connected (by Lemma 5), $G_{A \cup C,T}$ is also connected. By Lemma 6, it is sufficient to show that there exists an ordering of T having the adjacency and enclosure properties. It is clear that there exist orderings of T having the adjacency property. Suppose to the contrary that for each ordering of T having the adjacency property, it does not have the enclosure property. Then, there exist two vertices $s_i, s_j \in A \cup C$, where $N(s_j) \subseteq N(s_i)$, such that $N(s_i) - N(s_j)$ contains nonconsecutive vertices in the ordering of T. This means that $N(s_j)$ is a proper subset of $N(s_i)$, and $p(s_i) < p(s_j)$, $q(s_i) > q(s_j)$. So, $s_j \in C$. We consider two cases as follows.

Case 1: $s_i \in A$. Since $s_j \in C$, there exists a vertex $s_k \in A$ $(s_k \neq s_i)$ such that $N(s_i) \cap N(s_k) \neq \emptyset$ and $N(s_i) \cap N(s_k) \subseteq N(s_j)$, which is in contradiction to $p(s_i) < p(s_j)$ and $q(s_i) > q(s_j)$, because $N(s_i)$ and $N(s_k)$ are not proper subsets of each other and they contain consecutive vertices in the ordering of T.

Case 2: $s_i \in C$. Since $s_i \in C$, there exists a vertex $s_r \in A$ such that $N(s_i)$ is a proper subset of $N(s_r)$. Thus, we have $p(s_r) \leqslant p(s_i) < p(s_j)$ and $q(s_r) \geqslant q(s_i) > q(s_j)$, which imply $t_{p(s_r)} \notin N(s_j)$ and $t_{q(s_r)} \notin N(s_j)$. Also, since $s_j \in C$ and $N(s_j) \subset N(s_i)$, there exists a vertex $s_k \in A$ $(s_k \neq s_r)$ such that $N(s_r) \cap N(s_k) \neq \emptyset$ and $N(s_r) \cap N(s_k) \subseteq N(s_j)$, which is in contradiction to $t_{p(s_r)} \notin N(s_j)$ and $t_{q(s_r)} \notin N(s_j)$, because $N(s_r)$ and $N(s_k)$ are not proper subsets of each other and they contain consecutive vertices in the ordering of T. □

Lemma 10. *Suppose that $G = (S, T, E)$ is a doubly convex-bipartite graph. Then*
(1) *the number of orderings of A with the adjacency property is only two:*
(2) *the set $\{s_a, s_b\}$ is unique, where s_a and s_b are the first and the last vertices in the orderings of A with the adjacency property.*

Proof. Let us consider the graph $G_{A,T}$. Suppose that $s_1, s_2, \ldots, s_{|A|}$ and $t_1, t_2, \ldots, t_{|T|}$ are orderings of A and T, respectively, with the adjacency property. We can represent $G_{A,T}$ by a rectilinear polygon. Since $G_{A,T}$ is connected (by Lemma 8), it can be easily seen from the rectilinear polygon that for any three consecutive vertices s_i, s_{i+1}, s_{i+2}, $1 \leqslant i \leqslant |A| - 2$, there exist vertices t_j and t_k such that $t_j \in N(s_i) \cap N(s_{i+1})$ but $t_j \notin N(s_{i+2})$, and $t_k \in N(s_{i+1}) \cap N(s_{i+2})$ but $t_k \notin N(s_i)$. This implies that for any ordering of A with the adjacency property, we have either $r(s_i) < r(s_{i+1}) < r(s_{i+2})$ or $r(s_i) > r(s_{i+1}) > r(s_{i+2})$ for all i, $1 \leqslant i \leqslant |A| - 2$, where $r(s_i)$ denotes the position of s_i in the ordering. Consequently, the orderings of A with the adjacency property are either $s_1, s_2, \ldots, s_{|A|}$ or $s_{|A|}, \ldots, s_2, s_1$. Here, $\{s_a, s_b\} = \{s_1, s_{|A|}\}$. □

Lemma 11. *Suppose that $G = (S, T, E)$ is a doubly convex-bipartite graph. Then, for any strong ordering of $A \cup C$ and T, denoted by $s_1, s_2, \ldots, s_{|A \cup C|}, t_1, t_2, \ldots, t_{|T|}$, we have $s_1, s_{|A \cup C|} \in A$.*

Proof. Suppose that $s_1, s_2, \ldots, s_{\alpha-1}, s_\alpha, \ldots, s_\beta, \ldots, s_{|A \cup C|}, t_1, t_2, \ldots, t_{|T|}$ is a strong ordering of $A \cup C$ and T (its existence is assured by Lemmas 6 and 9). Then, by Lemma 7, the ordering of $A \cup C$ and the ordering of T have the adjacency and enclosure properties. We assume, to the contrary, $s_1 \in C$ or $s_{|A \cup C|} \in C$.

If $s_1 \in C$, we have $N(s_1) \subset N(s_\alpha)$, where $s_1, s_2, \ldots, s_{\alpha-1} \in C$ and $s_\alpha \in A$. Otherwise, there is a vertex $s_\beta \in A$, where $\beta > \alpha$, such that $N(s_1) \subset N(s_\beta)$. Hence, there is a vertex t_i such that $t_i \in N(s_1)$, $t_i \notin N(s_\alpha)$, and $t_i \in N(s_\beta)$, which contradicts the fact that the ordering of $A \cup C$ has the adjacency property.

Now, let $N(s_1) = \{t_x, t_{x+1}, \ldots, t_y\}$ and $N(s_\alpha) = \{t_u, t_{u+1}, \ldots, t_v\}$. We have $u \leqslant x < y \leqslant v$, but $u \neq x$ or $y \neq v$. If $u < x$, then $t_u \notin N(s_1)$ and the edges $(s_1, t_x), (s_\alpha, t_u)$ cross, which contradicts the fact that $s_1, s_2, \ldots, s_{\alpha-1}, s_\alpha, \ldots, s_\beta, \ldots, s_{|A \cup C|}, t_1, t_2, \ldots, t_{|T|}$ is a strong ordering of $A \cup C$ and T. On the other hand, if $u = x$, then $y < v$ must hold, which implies $t_v \notin N(s_1)$. Since $s_1 \in C$, there exists a vertex $s_\beta \in A$, where $s_\beta \neq s_\alpha$ and $\beta > \alpha$, such that $N(s_\alpha) \cap N(s_\beta) \neq \emptyset$ and $N(s_\alpha) \cap N(s_\beta) \subseteq N(s_1)$. Let $N(s_\beta) = \{t_p, t_{p+1}, \ldots, t_q\}$. If $p \geqslant u$, then $q \geqslant v$ because $s_\beta \in A$. This implies $t_v \in N(s_\alpha) \cap N(s_\beta) \subseteq N(s_1)$, which is a contradiction. So, we have $p < u$, which implies $t_p \notin N(s_\alpha)$. Since $\beta > \alpha$, the edges $(s_\beta, t_p), (s_\alpha, t_u)$ cross, which implies $t_p \in N(s_\alpha)$ (according to the definition of a strong ordering). This is again a contradiction.

Similarly, $s_{|A \cup C|} \in C$ also leads to a contradiction. □

Note that the strong ordering in Lemma 11 is obtained as follows (refer to Fig. 6): move the vertices in C (originally below s_j) above s_j and reverse their order, and do the same for the vertices in C which are above the last vertex in A.

Lemma 12. *Suppose that $G = (S, T, E)$ is a doubly convex-bipartite graph. Then, for any strong ordering of $A \cup C$ and T, denoted by $s_1, s_2, \ldots, s_{|A \cup C|}, t_1, t_2, \ldots, t_{|T|}$, we have $N(s_i) \subset N(s_1)$ or $N(s_i) \subset N(s_{|A \cup C|})$ for all $s_i \in S - (A \cup C)$.*

Proof. Suppose that $s_{o(1)}, s_{o(2)}, \ldots, s_{o(\alpha)}, \ldots, s_{o(\beta)}, \ldots, s_{o(|S|)}$ is an ordering of S with the adjacency property, where $s_{o(1)}, s_{o(2)}, \ldots, s_{o(\alpha-1)} \in S - (A \cup C)$, $s_{o(\alpha)} \in A$, $s_{o(\beta)} \in A$, and $s_{o(\beta+1)}, s_{o(\beta+2)}, \ldots, s_{o(|S|)} \in S - (A \cup C)$. By Lemma 1, we have $\{s_{o(1)}, s_{o(2)}, \ldots, s_{o(\alpha-1)}, s_{o(\beta+1)}, \ldots, s_{o(|S|)}\} = S - (A \cup C)$. According to the structural properties of the rectilinear polygon, we have $N(s_{o(i)}) \subset N(s_{o(\alpha)})$ for $1 \leqslant i \leqslant \alpha - 1$, and $N(s_{o(i)}) \subset N(s_{o(\beta)})$ for $\beta + 1 \leqslant i \leqslant |S|$.

By Lemmas 7 and 11, if the vertices in C are removed from $s_1, s_2, \ldots, s_{|A \cup C|}$, an ordering of A with the adjacency property is obtained. Besides, s_1 and $s_{|A \cup C|}$ are the first and the last vertices in the ordering. By Lemma 10, we have $\{s_1, s_{|A \cup C|}\} = \{s_{o(\alpha)}, s_{o(\beta)}\}$. □

From the proof of Lemma 12, we know that if $N(s_i) \neq N(s_j)$ for all $s_i, s_j \in S$, $s_i \neq s_j$, then the set $\{s_1, s_{|A \cup C|}\}$ is unique, where $s_1, s_2, \ldots, s_{|A \cup C|}, t_1, t_2, \ldots, t_{|T|}$ denotes any strong ordering of $A \cup C$ and T. In the following discussion, for any strong ordering of $A \cup C$ and T, we use s_a and s_b to denote the first and the last vertices, respectively, in the ordering of $A \cup C$.

Lemma 13. *Suppose that $G = (S, T, E)$ is a doubly convex-bipartite graph. Then, the complement of $G^*_{(S-(A \cup C)) \cup \{s_a, s_b\}, T}$ is a bipartite graph.*

Proof. To show that the complement of $G^*_{(S-(A \cup C)) \cup \{s_a, s_b\}, T}$ is a bipartite graph, it is sufficient to show that $G^*_{(S-(A \cup C)) \cup \{s_a, s_b\}, T}$ contains two disjoint complete subgraphs. The latter is a consequence of Lemmas 2 and 12. □

Lemma 14. *Suppose that $G = (S, T, E)$ is a bipartite graph. Then, there exist orderings of S with the adjacency property, if the following three conditions are satisfied.*

(1) *$G_{A \cup C, T}$ is a connected bipartite-permutation graph.*

(2) *For all $s_i \in S - (A \cup C)$, either $N(s_i) \subset N(s_a)$ or $N(s_i) \subset N(s_b)$, where $(s_a =) s_{r(1)}, s_{r(2)}, \ldots, s_{r(|A \cup C|)} (= s_b), t_1, t_2, \ldots, t_{|T|}$ is an arbitrary strong ordering of $A \cup C$ and T.*

(3) *The complement of $G^*_{(S-(A \cup C)) \cup \{s_a, s_b\}, T}$ is a bipartite graph.*

Proof. The existence of the strong ordering $(s_a =) s_{r(1)}, s_{r(2)}, \ldots, s_{r(|A \cup C|)} (= s_b)$, $t_1, t_2, \ldots, t_{|T|}$ is assured by condition (1) (see Lemma 6). Conditions (2) and (3) imply that $(S - (A \cup C)) \cup \{s_a, s_b\}$ can be split into two disjoint subsets $W_1 = \{s_{o(1)}, s_{o(2)}, \ldots, s_{o(u)}\}$ and $W_2 = \{s_{g(1)}, s_{g(2)}, \ldots, s_{g(v)}\}$ such that $N(s_{o(1)}) \subseteq N(s_{o(2)}) \subseteq \cdots \subseteq N(s_{o(u)})$ and $N(s_{g(1)}) \subseteq N(s_{g(2)}) \subseteq \cdots \subseteq N(s_{g(v)})$, where $\{s_{o(u)}, s_{g(v)}\} = \{s_a, s_b\}$. In the following, we show that $s_{o(1)}, s_{o(2)}, \ldots, s_{o(u)} (= s_a = s_{r(1)}), s_{r(2)}, \ldots, s_{r(|A \cup C|-1)}, (s_{r(|A \cup C|)} = s_b =) s_{g(v)}, s_{g(v-1)}, \ldots, s_{g(1)}$, which is an ordering of S, has the adjacency property.

For each vertex $t_i \in T$ and any two vertices $s_j, s_k \in N(t_i)$, where $s_j \neq s_k$, we consider the following six cases.

Case 1: $j = r(x)$, $k = r(y)$, and $x < y$. We have $s_{r(z)} \in N(t_i)$ for $x \leqslant z \leqslant y$ by Lemma 7.

Case 2: $j = o(x)$, $k = o(y)$, and $x < y$. We have $s_{o(z)} \in N(t_i)$ for $x \leqslant z \leqslant y$, because $t_i \in N(s_{o(x)}) \subseteq N(s_{o(x+1)}) \subseteq \cdots \subseteq N(s_{o(y)})$.

Case 3: $j = g(x)$, $k = g(y)$, and $x < y$. We have $s_{g(z)} \in N(t_i)$ for $x \leqslant z \leqslant y$, because $t_i \in N(s_{g(x)}) \subseteq N(s_{g(x+1)}) \subseteq \cdots \subseteq N(s_{g(y)})$.

Case 4: $j = o(x)$ and $k = r(y)$. We have $\{s_{o(x)}, s_{o(x+1)}, \ldots, s_{o(u)} (= s_{r(1)}), s_{r(2)}, \ldots, s_{r(y)}\} \subseteq N(t_i)$, by combining Cases 1 and 2.

Case 5: $j = r(x)$ and $k = g(y)$. We have $\{s_{r(x)}, s_{r(x+1)}, \ldots, (s_{r(k)} =) s_{g(v)}, s_{g(v-1)}, \ldots, s_{g(y)}\} \subseteq N(t_i)$, by combining Cases 1 and 3.

Case 6: $j = o(x)$ and $k = g(y)$. We have $\{s_{o(x)}, s_{o(x+1)}, \ldots, s_{o(u)} (= s_{r(1)}), s_{r(2)}, \ldots, (s_{r(k)} =) s_{g(v)}, s_{g(v-1)}, \ldots, s_{g(y)}\} \subseteq N(t_i)$, by combining Cases 1–3. □

Lemma 15. *Suppose that $G = (S \cup \{z\}, T, E)$ is a bipartite graph, where $z \notin S$ and $N(z) = N(s_i)$ for some $s_i \in S$. Then, an ordering of S: $s_1, s_2, \ldots, s_{i-1}, s_i, s_{i+1}, \ldots, s_{|S|}$ has the adjacency property if and only if an ordering of $S \cup \{z\}$: $s_1, s_2, \ldots, s_{i-1}, s_i, z, s_{i+1}, \ldots, s_{|S|}$ or $s_1, s_2, \ldots, s_{i-1}, z, s_i, s_{i+1}, \ldots, s_{|S|}$ has the adjacency property.*

Proof. ($\Rightarrow$) Suppose that $s_1, s_2, \ldots, s_{i-1}, s_i, z, s_{i+1}, \ldots, s_{|S|}$ does not have the adjacency property. Then, there exists a vertex $t_j \in T$ such that $N(t_j) = \{s_k, s_{k+1}, \ldots, s_{i-1}, s_i, s_{i+1}, \ldots, s_l\}$ (i.e., $z \notin N(t_j)$). This means $t_j \in N(s_i) = N(z)$, which is a contradiction. Similarly, we can prove that $s_1, s_2, \ldots, s_{i-1}, z, s_i, s_{i+1}, \ldots, s_{|S|}$ has the adjacency property.

($\Leftarrow$) It is trivial. □

The following lemma is an immediate consequence of Lemma 15.

Lemma 16. *Suppose that $G = (S \cup \{z_{p,q} \mid 1 \leqslant p \leqslant j \text{ and } 1 \leqslant q \leqslant k_p\}, T, E\}$ is a 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 $1 \leqslant p \leqslant j$ and $1 \leqslant q \leqslant k_p$. Then, an ordering of S: $s_1, s_2, \ldots, s_{i_1}, \ldots, s_{i_2}, \ldots, s_{i_j}, \ldots, s_{|S|}$ has the adjacency property if and only if an ordering of $S \cup \{z_{p,q} \mid 1 \leqslant p \leqslant j \text{ and } 1 \leqslant q \leqslant k_p\}$: $s_1, s_2, \ldots, s_{i_1}, z_{1,1}, z_{1,2}, \ldots, z_{1,k_1}, \ldots, s_{i_2}, z_{2,1}, z_{2,2}, \ldots, z_{2,k_2}, \ldots, s_{i_j}, z_{j,1}, z_{j,2}, \ldots, z_{j,k_j}, \ldots, s_{|S|}$ has the adjacency property.*

By Lemma 16, we can rewrite Lemma 14 as follows.

Lemma 17. *Suppose that $G = (S, T, E)$ is a bipartite graph. Define S' to be the largest subset of S such that $N(s_i) \neq N(s_j)$ for all $s_i, s_j \in S'$, $s_i \neq s_j$, and define the sets A, C with respect to S' (replacing S with S' in Definition 1). Then, there exist orderings of S having the adjacency property, if the following three conditions are satisfied.*

(1) $G_{A\cup C,T}$ *is a connected bipartite-permutation graph.*
(2) *For all* $s_i\in S'-(A\cup C)$, *either* $N(s_i)\subset N(s_a)$ *or* $N(s_i)\subset N(s_b)$, *where* $(s_a=)s_{r(1)}, s_{r(2)},\ldots,s_{r(|A\cup C|)}(=s_b), t_1,t_2,\ldots,t_{|T|}$ *is an arbitrary strong ordering of* $A\cup C$ *and* T.
(3) *The complement of* $G^*_{(S'-(A\cup C))\cup\{s_a,s_b\},T}$ *is a bipartite graph.*

If we exchange S and T in Lemma 17, then we obtain the following lemma, which is the dual of Lemma 17.

Lemma 18. *Suppose that* $G=(S,T,E)$ *is a bipartite graph. Define* T' *to be the largest subset of* T *such that* $N(t_i)\neq N(t_j)$ *for all* $t_i,t_j\in T'$, $t_i\neq t_j$, *and define the sets* A',C' *with respect to* T'. *Then, there exist orderings of* T *having the adjacency property, if the following three conditions are satisfied.*
(1) $G_{S,A'\cup C'}$ *is a connected bipartite-permutation graph.*
(2) *For all* $t_i\in T'-(A'\cup C')$, *either* $N(t_i)\subset N(t_a)$ *or* $N(t_i)\subset N(t_b)$, *where* $s_1,s_2,\ldots,s_{|S|}$, $(t_a=)t_{r(1)},t_{r(2)},\ldots,t_{r(|A'\cup C'|)}(=t_b)$ *is an arbitrary strong ordering of* S *and* $A'\cup C'$.
(3) *The complement of* $G^*_{(T'-(A'\cup C'))\cup\{t_a,t_b\},S}$ *is a bipartite graph.*

In Lemma 18, similar to A,C with respect to S' in Lemma 17, the sets A',C' are defined with respect to T'. That is, we replace A, C, S and T with A', C', T' and S, respectively, in Definition 1.

Theorem 1. *Suppose that* $G=(S,T,E)$ *is a bipartite graph. Define* S' *to be the largest subset of* S *such that* $N(s_i)\neq N(s_j)$ *for all* $s_i,s_j\in S'$, $s_i\neq s_j$, T' *to be the largest subset of* T *such that* $N(t_i)\neq N(t_j)$ *for all* $t_i,t_j\in T'$, $t_i\neq t_j$, *and define the sets* A,C *with respect to* S', *the sets* A',C' *with respect to* T'. *Then, the following two statements are equivalent for* G.
(1) G *is a doubly convex-bipartite graph.*
(2) G *satisfies the following six conditions.*
  (a) $G_{A\cup C,T}$ *is a connected bipartite-permutation graph.*
  (b) *For all* $s_i\in S'-(A\cup C)$, *either* $N(s_i)\subset N(s_a)$ *or* $N(s_i)\subset N(s_b)$, *where* $(s_a=)s_{r(1)},s_{r(2)},\ldots,s_{r(|A\cup C|)}(=s_b),t_1,t_2,\ldots,t_{|T|}$ *is an arbitrary strong ordering of* $A\cup C$ *and* T.
  (c) *The complement of* $G^*_{(S'-(A\cup C))\cup\{s_a,s_b\},T}$ *is a bipartite graph.*
  (d) $G_{S,A'\cup C'}$ *is a connected bipartite-permutation graph.*
  (e) *For all* $t_i\in T'-(A'\cup C')$, *either* $N(t_i)\subset N(t_a)$ *or* $N(t_i)\subset N(t_b)$, *where* $s_1,s_2,\ldots,s_{|S|},(t_a=)t_{r(1)},t_{r(2)},\ldots,t_{r(|A'\cup C'|)}(=t_b)$ *is an arbitrary strong ordering of* S *and* $A'\cup C'$.
  (f) *The complement of* $G^*_{(T'-(A'\cup C'))\cup\{t_a,t_b\},S}$ *is a bipartite graph.*

Proof. Like Lemma 18 which is the dual of Lemma 17, the duals of Lemmas 9, 12 and 13 are also valid. For example, the dual of Lemma 9 is obtained by exchanging S and T and defining the sets A,C with respect to T. By Lemmas 9, 12, 13 and their duals, statement (1) implies statement (2). By Lemmas 17 and 18, statement (2) implies statement (1). So, statement (1) and statement (2) are equivalent. □

3. A parallel recognition algorithm

In this section, based upon Theorem 1, a parallel algorithm is proposed for recognizing connected doubly convex-bipartite graphs.

Algorithm 1.

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

Step 1. Determine if G is a bipartite graph. Let $G = (S, T, E)$, if the answer is yes.

Step 2. Determine if G is a connected doubly convex-bipartite graph, by checking the six conditions in statement (2) of Theorem 1.

2.1. Find the largest subset S' of S such that $N(s_i) \neq N(s_j)$, for all $s_i, s_j \in S'$, $s_i \neq s_j$.

2.2. Partition S' into subsets A, C and $S' - (A \cup C)$.

2.3. Determine if $G_{A \cup C, T}$ is a bipartite-permutation graph. If yes, generate a strong ordering of $A \cup C$ and T, denoted by $(s_a =)s_{r(1)}, s_{r(2)}, \ldots, s_{r(|A \cup C|)}(=s_b), t_1, t_2, \ldots, t_{|T|}$.

2.4. Determine if $N(s_i) \subset N(s_a)$ or $N(s_i) \subset N(s_b)$ holds for all $s_i \in S' - (A \cup C)$.

2.5. Construct the complement of $G^*_{(S' - (A \cup C)) \cup \{s_a, s_b\}, T}$ and determine if it is a bipartite graph.

2.6. Determine if conditions (d)–(f) in statement (2) of Theorem 1 are satisfied, similar to Step 2.1 to Step 2.5.

Step 3. Construct orderings of S and T having the adjacency property, if G is a connected doubly convex-bipartite graph.

3.1. Construct an ordering of S having the adjacency property.

3.2. Construct an ordering of T having the adjacency property.

Recall that doubly convex-bipartite graphs are a subclass of bipartite-circle graphs. So, if G is a doubly convex-bipartite graph, a circle-graph model of G can be constructed by an additional step.

3.3. Construct a circle-graph model for 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.

Now, the complexity of Algorithm 1 is analyzed as follows. Step 1 can be completed in $O(\log n)$ time using $O(n^2)$ 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 $O(n^2)$

processors on the CRCW PRAM [15] or $O(\log^2 n)$ time using $O(n^2/\log^2 n)$ processors on the CREW PRAM [4]. Computing the level numbers of vertices of a tree takes $O(\log n)$ time if $O(n)$ processors are used on the CREW PRAM [15]. The remaining work can be completed in $O(1)$ time using $O(n^2)$ processors on the CREW PRAM.

Step 2.1 can be completed in $O(\log n)$ time using $O(n^3/\log n)$ processors of 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 on the CREW PRAM. Then, the set S' can be determined by executing the connected component algorithm of Shiloach and Vishkin [13], which takes $O(\log n)$ time using $O(n^2)$ processors on the CRCW PRAM, or the connected component algorithm of Chin et al. [4], which takes $O(\log^2 n)$ time using $O(n^2/\log^2 n)$ processors on the CREW PRAM.

Step 2.2 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 on the CREW PRAM.

Step 2.3 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, if the algorithm of Yu and Chen [16] is applied. Step 2.4 takes $O(\log n)$ time using $O(n^2/\log n)$ processors on the CREW PRAM. Since the construction of the graph $G^*_{(S'-(A\cup C))\cup\{s_a,s_b\},T}$ takes $O(\log n)$ time using $O(n^3/\log n)$ processors on the CREW PRAM, Step 2.5 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.

Step 2.6 requires the same time complexities as the total of Steps 2.1–2.5, if sufficient processors are used. In total, Step 2 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 3.1 can be completed in $O(\log n)$ time using $O(n^3/\log n)$ processors on the CREW PRAM. Let $G^c = (W_1, W_2, E')$ denote the complement of $G^*_{(S'-(A\cup C))\cup\{s_a,s_b\},T}$ (G^c is a bipartite graph), where $W_1 = \{s_{o(1)}, s_{o(2)}, \ldots, s_{o(u)}(=s_a)\}$, $W_2 = \{s_{g(1)}, s_{g(2)}, \ldots, s_{g(v)}(=s_b)\}$, $u + v = |S' - (A \cup C)| + 2$, and $N(s_{o(1)}) \subseteq N(s_{o(2)}) \subseteq \cdots \subseteq N(s_{o(u)})$, $N(s_{g(1)}) \subseteq N(s_{g(2)}) \subseteq \cdots \subseteq N(s_{g(v)})$. We can construct a directed graph as follows: W_1 is the vertex set and there is a directed edge from s_i to s_j if $N(s_i) \subseteq N(s_j)$, where $s_i, s_j \in W_1$. The construction of the directed graph requires $O(\log n)$ time, if $O(n^3/\log n)$ processors are used on the CREW PRAM. Then, the ordering of W_1: $s_{o(1)}, s_{o(2)}, \ldots, s_{o(u)}$ can be obtained by sorting W_1 increasingly according to the indegrees of $s_{o(i)}$'s, $i = 1, \ldots, u$. The sorting takes $O(\log n)$ time using $O(n)$ processors on the EREW PRAM [5]. The ordering of W_2: $s_{g(1)}, s_{g(2)}, \ldots, s_{g(v)}$ can be obtained similarly. From the proof of Lemma 14, we know that $s_{o(1)}, s_{o(2)}, \ldots, s_{o(u)}(=s_a = s_{r(1)}), s_{r(2)}, \ldots, s_{r(|A\cup C|-1)}, (s_{r(|A\cup C|)} = s_b =) s_{g(v)}, s_{g(v-1)}, \ldots, s_{g(1)}$ is an ordering of S' having the adjacency property. This ordering can be further expanded

to an ordering of S having the adjacency property by adding all the vertices in $S - S'$. This can be done in $O(\log n)$ time using $O(n)$ processors on the CREW PRAM by the aid of a sorting operation.

Step 3.2 has the same time complexity and processor complexity as Step 3.1.

Step 3.3 takes $O(\log n)$ time if $O(n^2)$ processors are used on the EREW PRAM. The proof of Lemma 3 suggests an approach to obtaining a circle-graph model for G. If we let the squares along the boundary of the corresponding rectilinear polygon form a linked list, then a circle-graph model for G can be obtained by the aid of a list ranking operation.

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

Thus, 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.

Theorem 2. *A connected doubly convex-bipartite 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, where n is the number of vertices in G. Besides, a circle-graph model is constructed for the connected doubly convex-bipartite graph with the same time complexities and processor complexities. Here, the graph representation adopted is the adjacency matrix.*

4. Discussion and conclusions

Many special classes of perfect graphs arise quite naturally in real-world applications including optimization of computer storage, analysis of genetic structure, synchronization of parallel processes, and certain scheduling problems [10]. For example, permutation graphs have been used in modeling and solving various problems such as determining intersection-free layouts for connection boards or optimal schedules for reallocation of memory space in a computer. It is known that bipartite-permutation graphs are a subclass of doubly convex-bipartite graphs, which is a subclass of convex-bipartite graphs. The applications of convex-bipartite graphs and doubly convex-bipartite graphs can be found in [6, 9, 12].

In Section 3, we have assumed that the input graph is connected. In fact, the restriction to a connected graph can be removed. If the input graph is not connected, we simply execute the proposed algorithm for each of its components. The input graph is a doubly convex-bipartite graph if each of its components is a doubly convex-bipartite graph. Also, a circle-graph model of the input graph can be obtained by merging the circle-graph models of the components. So, the time complexity and processor complexity required remain the same for a disconnected input graph.

Given a graph G, the edge-coloring problem is to assign edges of G with colors so that adjacent edges are assigned with different colors. The objective is to minimize the

number of colors used. This problem is known to be NP-hard [7] for a general graph, but is polynomial-time solvable for some special classes of graphs. For example, it can be solved for a bipartite graph in $O(\log^3 n)$ time using $O(m)$ processors on the CREW PRAM [8]. However, it is still open for circle graphs, permutation graphs, and cographs [1]. In [17], considering doubly convex-bipartite graphs, the authors presented a polynomial-time solution to the edge-coloring problem. When an ordering of S and an ordering of T having the adjacency property are given, the algorithm proposed in [17] runs in $O(\log m)$ time using $O(m/\log m)$ processors on the CREW PRAM, which achieves cost optimality.

Acknowledgements

The authors are grateful to the anonymous referees and the editor for their valuable comments which have imporved the readability of this paper a lot.

References

[1] L. Cai and J.A. Ellis, NP-completeness of edge-colouring some restricted graphs, *Discrete Appl. Math.* **30** (1991) 15–27.
[2] L. Chen and Y. Yesha, Parallel recognition of the consecutive ones property with applications, *J. Algorithms* **12** (1991) 375–392.
[3] L. Chen and Y. Yesha, Fast parallel algorithms for bipartite permutation graphs, *Networks* **22** (1993) 29–39.
[4] F.Y. Chin, J. Lam and I. Chen, Efficient parallel algorithms for some graph problems, *Comm. ACM* **25** (1982) 659–655.
[5] R. Cole, Parallel merge sort, *SIAM J. Comput.* **17** (1988) 770–785.
[6] E. Dekel and S. Sahni, A parallel matching algorithm for convex bipartite graphs and applications to scheduling, *J. Parallel Distrib. Comput.* **1** (1984) 185–205.
[7] M.R. Garey and D.S. Johnson, *Computers and Intractability: a Guide to the Theory of* NP-*completeness* (Freeman, San Francisco, CA, 1979).
[8] A. Gibbons and W. Rytter, *Efficient Parallel Algorithms* (Cambridge Univ. Press, Cambridge, 1988).
[9] F. Glover, Maximum matching in a convex bipartite graph, *Naval Res. Logist. Quart.* **14** (1967) 313–316.
[10] M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs* (Academic Press, New York, 1980).
[11] P.N. Klein and J.H. Reif, An efficient parallel algorithm for planarity, *J. Comput. System Sci.* **37** (1988) 190–246.
[12] J.W. Lipski and F.P. Preparata, Efficient algorithms for finding maximum matchings in convex bipartite graphs and related problems, *Acta Inform.* **15** (1982) 329–346.
[13] Y. Shiloach and U. Vishkin, An $O(\log n)$ parallel connectivity algorithm, *J. Algorithms* **3** (1982) 57–67.
[14] J. Spinrad, A. Brandstadt and L. Stewart, Bipartite permutation graphs, *Discrete Appl. Math.* **18** (1987) 279–292.
[15] U. Vishkin, On efficient parallel strong orientation, *Inform. Process. Lett.* **20** (1985) 235–240.
[16] C.W. Yu and G.H. Chen, An efficient parallel recognition algorithm for bipartite-permutation graphs, in: *Proc. Internat. Conf. on Parallel and Distributed Systems* (Hsinchu, Taiwan, 1992) 370–377.
[17] C.W. Yu and G.H. Chen, Efficient parallel algorithms for doubly convex-bipartite graphs, Tech. Report 92-04, Dept. Comput. Sci & Info. Engg., National Taiwan Univ., Taipei, Taiwan, 1992.