Bipartite Permutation Graphs with Application to the Minimum Buffer Size Problem*

Ten-Hwang Lai and Shu-Shang Wei

Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210

E-mail: lai@cis.ohio-state.edu

September 23, 1994

Abstract. This paper presents a new characterizations of bipartite permutation graphs and a structure theorem for (0,1)-matrices with a special consecutive 1's property. These results lead to a linear time algorithm for the minimum buffer size problem when restricted to bipartite permutation graphs; this problem arises in relational database systems and is NP-hard for a general graph.

1 Introduction

Let $G = (U, V, E)$ be a bipartite graph, where U is the *left node set* and V the *right node set*. A subset $S \subseteq U$ (or V) is called a *segment* if its elements are consecutive in U (or in V). For any node x in G, let $\bar{x}$ denote the set of all nodes that are adjacent to x; i.e., $\bar{x} = \{y : (x, y) \in E\}$. A labeling of the nodes of V has the *convex property* if for each node u in U, $\bar{u}$ is a segment in the labeling of V. A labeling with the convex property is a *convex labeling*. G is said to be *convex on* V if there exists a convex labeling for V. The term *convex on* U is similarly defined. A *convex graph* is a bipartite graph that is convex on at least one node set. A *biconvex graph* is one that is convex on both node sets.

Booth and Lueker [1] described an algorithm that determines whether a given bipartite graph $G = (U, V, E)$ is convex on U (or V) and if so, generates a convex labeling for the node set. Applying the algorithm twice, on U and V respectively, one can test whether a bipartite graph is biconvex, and if it is, obtain a biconvex labeling for the graph. Booth and Lueker's algorithm has a time complexity of $O(|U| + |V| + |E|)$.

A subclass of biconvex graphs is the class of bipartite permutation graphs. A graph is a *permutation graph* if there is some pair of labelings of the nodes such that an edge

*This work was supported in part by National Science Foundation under Grant CCR-9010589.

exists between nodes x and y if and only if x precedes y in one of the labelings while y precedes x in the other [3, 8]. The fact that bipartite permutation graphs are biconvex is a direct result from a Characterization Theorem [8] that indicates that the following statements are equivalent for a bipartite graph $G = (U, V, E)$:

(i) G is a bipartite permutation graph.

(ii) There is a strong ordering of $U \cup V$. (A *strong ordering* is a labeling of U and a labeling of V such that for all $(u_a, v_b) \in E$ and $(u_{a'}, v_{b'}) \in E$, where u_a and $u_{a'}$ are in U and v_b and $v_{b'}$ are in V, if $a < a'$ and $b' < b$, then $(u_a, v_{b'}) \in E$ and $(u_{a'}, v_b) \in E$.)

(iii) There exists a convex labeling of V which has the additional *enclosure property* that for every pair of nodes x, y in U, if $\bar{x} \supseteq \bar{y}$ then $\bar{x} - \bar{y}$ is a segment.

Based on these characterizations, Spinrad et al. developed an $O(|U| + |V| + |E|)$ time algorithms that determines whether a bipartite graph $G(U, V, E)$ is a permutation graph.

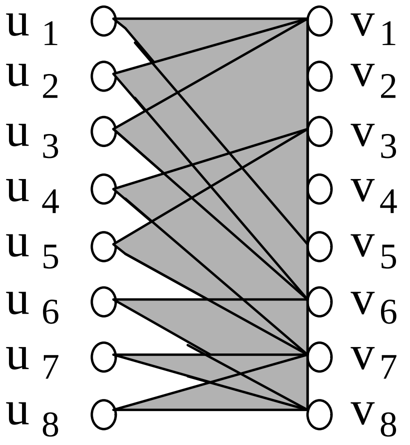
Let $G = (U, V, E)$ be a bipartite graph, and let the nodes be labeled as $U = \{u_1, \ldots, u_m\}$ and $V = \{v_1, \ldots, v_n\}$. If V is convex, then a condition stronger than the enclosure property is the *forward property*: for every pair of nodes u_i, u_j in U, if $i < j$ then $\bar{u}_i \preceq \bar{u}_j$, where "$\preceq$" is a partial ordering defined as below. For any segment $S = \{x_a, x_{a+1}, \ldots, x_b\}$ in U or V, define $first(S)$ and $last(S)$ respectively to be the index of the first element in S and that of the last element in S (i.e., $first(S) = a$ and $last(S) = b$). For two segments A and B in the same node set (both in U or both in V), define $A \preceq B$ if and only if $first(A) \leq first(B)$ and $last(A) \leq last(B)$.

Definition 1 A bipartite graph $G = (U, V, E)$ is *forward-convex* if there exists a labeling of $U \cup V$ such that V is convex and the labeling has the forward property. Such a labeling is called a *forward-convex labeling.*

Figure 1 shows an example forward-convex graph and its adjacency matrix; the labeling in the figure is a forward-convex labeling.

Recently, while studying database systems, we observed [11] that those bipartite graphs (called *page connectivity graphs* in [2, 6, 5]) which are used to model certain join operations in relational database systems are often forward-convex. This motivated our interests in studying forward-convex graphs.

It turns out that forward-convex graphs and bipartite permutation graphs are the same class of graphs. Thus, forward-convex is another characteristic property of permutation graphs. In this paper we report two results concerning forward-convex graphs (bipartite permutation graphs). First, we establish another characteristic property for bipartite permutation graphs. This yields an algorithm that determines whether a bipartite graph is forward-convex, and if it is, generates a forward-convex labeling for the graph. When translated into the language of (0,1)-matrices, the new characteristic property becomes a structure theorem for forward-convex matrices, which may



(a)

U \ V	1	2	3	4	5	6	7	8
1	1	1	1	1	1	0	0	0
2	1	1	1	1	1	1	0	0
3	1	1	1	1	1	1	0	0
4	0	0	1	1	1	1	1	0
5	0	0	1	1	1	1	1	0
6	0	0	0	0	0	1	1	1
7	0	0	0	0	0	0	1	1
8	0	0	0	0	0	0	1	1

(b)

Figure 1: A forward-convex graph.

be regarded as a variant of Tucker's structure theorem for the consecutive 1's property. Second, we show that an NP-hard problem — the *Minimum Buffer Size Problem* — that has applications in relational database systems can be efficiently solved when restricted to bipartite permutation graphs.

Our recognition algorithm has the same time complexity as the recognition algorithm of [8]. These two algorithms are different in approach, and they generate different labelings. If the graph in question is a bipartite permutation graph, the Spinrad et al. algorithm will generate a labeling with convex and enclosure properties, while our algorithm will generate a forward-convex labeling. A forward-convex labeling definitely has convex and enclosure properties. The reverse is in general not true. Our algorithm for the minimum buffer size problem needs a forward-convex labeling.

2 Forward-Convex and Bipartite Permutation Graphs

This section presents two results: 1) forward-convex graphs and bipartite permutation graphs are equivalent, and 2) a biconvex graph is forward-convex if and only if it does not contain an induced subgraph that is isomorphic to either of the two graphs in Fig. 2. The latter result leads to a linear time algorithm that recognizes a bipartite permutation graph and generates a forward-convex labeling.

A subgraph $G' = (V', E')$ of a graph $G = (V, E)$ is said to be an *induced* subgraph if it is induced by the node set V', i.e., if $E' = \{(u, v) \in E : u, v \in V'\}$.

Throughout this paper, let G_1 and G_2 be the two graphs depicted in Fig. 2. Neither of them is forward-convex.

The main result of this section is the following theorem that characterizes the

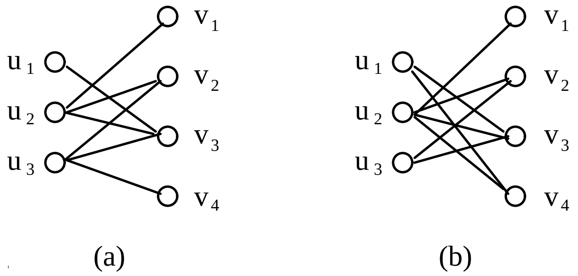


Figure 2: Two non-forward-convex graphs G_1 and G_2.

class of forward-convex graphs. This theorem augments Spinrad et al.'s theorem [8] in characterizing bipartite permutation graphs.

Theorem 1 *The following statements are equivalent for a bipartite graph $G = (U, V, E)$:*

(a) G is a bipartite permutation graph.
(b) G is forward-convex.
(c) G is biconvex and contains no induced subgraph that is isomorphic to G_1 or G_2.

Proof. (a) $\Rightarrow$ (b): By Spinrad et al.'s characterization theorem, it suffices to show that every strong labeling can be converted to a forward-convex labeling. Thus, let G be a bipartite permutation graph, and let $U = \{u_1, \ldots, u_m\}$ and $V = \{v_1, \ldots, v_n\}$ be a strong labeling. Assume that in this labeling there is no isolated node between non-isolated nodes (otherwise relabel the nodes to satisfy this condition.) Let u_a be any node in U, and suppose $(u_a, v_i) \in E$ and $(u_a, v_j) \in E$. Consider any node v_k, $i < k < j$. Since no isolated node exists between v_i and v_j, v_k is not isolated and thus joined to some u_b by an edge. Due to the strong property of the labeling, (u_a, v_k) is an edge in E. So, G is convex on V. Now consider any two nodes u_a and u_b with $a < b$. The strong property implies $first(\bar{u}_a) \leq first(\bar{u}_b)$ and $last(\bar{u}_a) \leq last(\bar{u}_b)$, so the labeling is forward-convex.

(b) $\Rightarrow$ (a): Suppose that G has a forward-convex labeling: $U = \{u_1, \ldots, u_m\}$ and $V = \{v_1, \ldots, v_n\}$. Let $(u_a, v_b) \in E$ and $(u_{a'}, v_{b'}) \in E$, where $1 \leq a < a' \leq m$ and $1 \leq b' < b \leq n$. Since the labeling is forward-convex on V, $\bar{u}_a \preceq \bar{u}_{a'}$ and hence $first(\bar{u}_a) \leq first(\bar{u}_{a'}) \leq b' < b \leq last(\bar{u}_a)$. Thus, $(u_a, v_{b'}) \in E$. A similar argument leads to $(u_{a'}, v_b) \in E$. So the labeling has the strong ordering property.

(b) $\Rightarrow$ (c): This follows directly from the following two simple observations:

- If G is forward-convex, then every induced subgraph of G is forward-convex.
- Neither G_1 nor G_2 is forward-convex.

(c) $\Rightarrow$ (b): The proof is constructive. We develop an algorithm that generates a forward-convex labeling for any given graph satisfying condition (c). The algorithm is described in the Section 4. □

Corollary 1 *Let G be a bipartite permutation graph without isolated nodes. A labeling of G has the forward-convex property if and only if it has the strong ordering property.*

Since the strong property is symmetric between U and V, it follows from this corollary that a labeling is forward-convex on U if and only if it is forward-convex on V.

3 Structure Theorem for Forward-Convex Matrices

A bipartite graph $G = (U, V, E)$ can be represented by an adjacency matrix $M(1..m, 1..n)$ such that $M(i, j) = 1$ if $(u_i, v_j) \in E$, and $M(i, j) = 0$ if $(u_i, v_j) \notin E$. With this representation, properties of bipartite graphs can be translated to properties of (0,1)-matrices, and vice versa.

A $(0, 1)$-matrix M is said to have the *consecutive 1's property for rows* if the columns of M can be permuted so that the 1's in each row appear consecutively [10]. It is clear that G is convex on V (U) if and only if its adjacency matrix has the consecutive 1's property for rows (columns).

A (0,1)-matrix M is said to have the *forward-convex property for rows* if the rows and/or columns of M can be permuted so that the 1's in each row are consecutive and, in addition, the rows of 1's, when examined from top to bottom, shift gradually from left to right. (See Fig. 1(b) for illustration.) Similarly, M is said to have the *forward-convex property for columns* if the rows and/or columns of M can be permuted so that the 1's in each column are consecutive and, in addition, the columns of 1's, when examined from left to right, shift gradually from top to bottom. It is clear that a matrix has the forward-convex property for rows if and only if it has the forward-convex property for columns. It is also clear that the following lemma is true.

Lemma 1 *A bipartite graph $G = (U, V, E)$ is forward-convex if and only if its adjacency matrix is forward-convex (for rows and columns).*

Tucker [10] proved a structure theorem for (0,1)-matrices with the consecutive 1's property: a (0,1)-matrix M has the consecutive 1's property for columns if and only if no submatrix of M is a member of five specific configurations. A similar theorem for matrices with the consecutive 1's property for rows and columns was also proved in the same paper. We desire to obtain a similar structure theorem for forward-convex matrices.

U \ V	1	**2**	3	**4**	**5**	6	**7**	8	9	10	11
1	0	**0**	1	**1**	**0**	0	**0**	0	0	0	0
2	0	0	1	1	1	0	0	0	0	0	0
3	1	**1**	1	**1**	**1**	0	**0**	0	0	0	0
4	0	**0**	1	**1**	**1**	1	**1**	0	0	0	0
5	0	0	0	0	1	1	1	1	0	0	0
6	0	**0**	0	**0**	**0**	1	**1**	1	1	1	1
7	0	0	0	0	0	0	1	1	1	1	0
8	0	0	0	0	0	0	0	1	1	0	0

M

U \ V	1	2	3	4
1	0	1	0	0
2	1	1	1	0
3	0	1	1	1
4	0	0	0	1

M'

Figure 3: Example submatrix.

Suppose G' and G'' are bipartite graphs, and M' and M'' are their adjacency matrices, respectively. Matrix M' is said to be *isomorphic* to matrix M'' if and only if graph G' is isomorphic to graph G''. It is not hard to see that M' is isomorphic to M'' if and only if M' can be obtained from M'' by a sequence of operations, where each operation is an interchange of two rows or two columns or a transposition of the matrix.

Let M_1 and M_2 be the adjacency matrices of G_1 and G_2, respectively:

$$M_1 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \quad \text{and} \quad M_2 = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}. \tag{1}$$

A *submatrix* of a matrix is a matrix consisting of those elements at the intersections between a number of rows and a number of columns. For example, in Fig. 3, M' is a submatrix of M obtained from rows 1, 3, 4, 6 and columns 2, 4, 5, 7.

The adjacency matrix of a subgraph of G is not necessarily a submatrix of M. However, if the subgraph is an *induced* subgraph, then its matrix is a submatrix of M. Conversely, each submatrix of M defines an induced subgraph of G. These immediately lead to the following.

Lemma 2 *A bipartite graph G contains an induced subgraph isomorphic to G_1 or G_2 if and only if its adjacency matrix M contains a submatrix isomorphic to M_1 or M_2.*

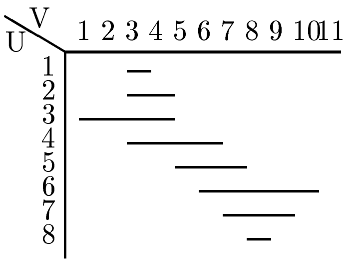
The "(b) $\Leftrightarrow$ (c)" part of Theorem 1 is equivalent to the the following structure theorem for forward-convex matrices. The former theorem is expressed in terms of graphs, while the latter is in terms of (0,1)-matrices.

Theorem 2 *A (0,1)-matrix has the forward-convex property if and only if it has the consecutive 1's properties for rows and columns and, in addition, none of its submatrices is isomorphic to M_1 or M_2.*

The "$\Rightarrow$" part of the theorem follows directly from Lemma 2 and the "(b) $\Rightarrow$ (c)" part of Theorem 1. The next section is devoted to proving the other direction.

U \ V	1	2	3	4	5	6	7	8	9	10	11
1	0	0	1	1	0	0	0	0	0	0	0
2	0	0	1	1	1	0	0	0	0	0	0
3	1	1	1	1	1	0	0	0	0	0	0
4	0	0	1	1	1	1	1	0	0	0	0
5	0	0	0	0	1	1	1	1	0	0	0
6	0	0	0	0	0	1	1	1	1	1	1
7	0	0	0	0	0	0	1	1	1	1	0
8	0	0	0	0	0	0	0	1	1	0	0

(a)



(b)

Figure 4: Example of an adjacency matrix and its line representation.

4 Recognizing Forward-Convex graphs and Forward-Convex Matrices

Since there is a one-one correspondence between bipartite graphs and (0,1)-matrices, we will not distinguish between these two notions. In particular, we will not distinguish between Theorem 2 and the "(b) $\Leftrightarrow$ (c)" part of Theorem 1.

Given a biconvex graph containing no induced subgraph isomorphic to G_1 or G_2, we want to generate a forward-convex labeling for the graph. If the given graph is not biconvex and/or has G_1 or G_2 as an induced subgraph, we want to detect this fact, too.

Let $G = (U, V, E)$ be the given bipartite graph. Applying twice Booth and Lueker's algorithm to a bipartite graph, we can either produce a biconvex labeling for the graph or conclude that it is not biconvex (and thus not forward-convex). Suppose that G is biconvex, and the labeling $U = \{u_1, \ldots, u_m\}$ and $V = \{v_1, \ldots, v_n\}$ has the convex property on both U and V. Furthermore, since a graph is forward-convex (resp. satisfies condition (c) of Theorem 1) if and only if each of its connected components is forward-convex (or satisfies condition (c)), we further assume that G is connected.

For convenience, we will represent a row of consecutive 1's by a line segment. For instance, the matrix in Fig. 4(a) is represented by the diagram in Fig. 4(b).

4.1 Properties of Biconvex Graphs

Let $G = (U, V, E)$ be a connected biconvex graph, and let $U = \{u_1, \ldots, u_m\}$ and $V = \{v_1, \ldots, v_n\}$ be a biconvex labeling. Relative to this labeling, let M be the adjacency matrix of G.

The following lemma shows the relationship among the rows of consecutive 1's in M: when examined from top to bottom, these rows first expand, then move to the right (or to the left), and finally shrink (see Fig. 4(b)).

Lemma 3 *Let $G = (U, V, E)$ be a connected biconvex graph with biconvex labeling $U = \{u_1, u_2, \ldots, u_m\}$ and $V = \{v_1, v_2, \ldots, v_n\}$. Let u_0 and u_{m+1} be two dummy nodes such that $\bar{u}_0 = \bar{u}_{m+1} = \emptyset$. Then, there exist indices p and q, $1 \leq p \leq q \leq m$, such that one of the following conditions holds:*

$$\emptyset = \bar{u}_0 \subseteq \bar{u}_1 \subseteq \cdots \subseteq \bar{u}_p \preceq \cdots \preceq \bar{u}_q \supseteq \cdots \supseteq \bar{u}_m \supseteq \bar{u}_{m+1} = \emptyset \tag{2}$$

or

$$\emptyset = \bar{u}_0 \subseteq \bar{u}_1 \subseteq \cdots \subseteq \bar{u}_p \succeq \cdots \succeq \bar{u}_q \supseteq \cdots \supseteq \bar{u}_m \supseteq \bar{u}_{m+1} = \emptyset. \tag{3}$$

Proof. Let a be such that $first(\bar{u}_a)$ is smallest among all $first(\bar{u}_i)$'s, and let b be such that $last(\bar{u}_b)$ is largest among all $last(\bar{u}_i)$'s. Ties are resolved arbitrarily. Let $p = \min\{a, b\}$ and $q = \max\{a, b\}$. One readily sees that Eq, 2S holds if $a \leq b$, and the other holds otherwise. □

Note that Eq. 2 and Eq. 3 are symmetric to each other in the sense that one becomes the other if the nodes of U are numbered in reversed order. Thus, in order to determine if G is forward-convex, we may assume without loss of generality that that Eq. 2 holds. In other words, we assume that the rows of consecutive 1's in the matrix first expand, then move to the right (rather than the left), and finally shrink. There may be more than one pair of (p, q) satisfying Eq. 2. By first choosing p to be as large as possible and then choosing q to be as small as possible (under the condition that $p \leq q$), we may further assume the following:

$$\bar{u}_p \neq \bar{u}_{p+1} \tag{4}$$

and

$$\text{if} \quad p < q \quad \text{then} \quad \bar{u}_{q-1} \neq \bar{u}_q. \tag{5}$$

Let $G = (U, V, E)$ be a given connected biconvex graph satisfying Eqs. 2, 4 and 5. Define $p' > p$ to be the smallest index such that $\bar{u}_{p'} \neq \bar{u}_p$. Similarly, define $q' < q$ to be the largest index such that $\bar{u}_{q'} \neq \bar{u}_q$. By Eqs. 4 and 5, we have (1) $p' = p + 1$, and (2) $q' = q - 1$ if $p < q$. When $p = q$, q' could be smaller than $q - 1$.

Let A be the set of segments (rows of consecutive 1's) in the top p segments that have overlaps with $\bar{u}_{p'}$; that is,

$$A = \{\bar{u}_i : 1 \leq i \leq p, \bar{u}_i \cap \bar{u}_{p'} \neq \emptyset\}. \tag{6}$$

Similarly, define

$$B = \{\bar{u}_j : q \leq j \leq m, \bar{u}_j \cap \bar{u}_{q'} \neq \emptyset\}. \tag{7}$$

Let

$$A' = \{\bar{u}_1, \ldots, \bar{u}_p\} - A \tag{8}$$

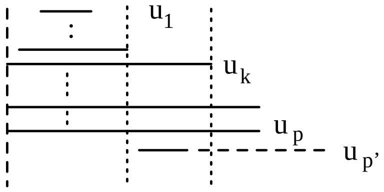


Figure 5: A is left-aligned.

	U \ V	1	2	3	4	5	6	7	8	9	10	11
A'	1	0	0	0	1	0	0	0	0	0	0	0
A'	2	0	0	1	1	0	0	0	0	0	0	0
A'	3	0	1	1	1	1	0	0	0	0	0	0
A	k = 4	1	1	1	1	1	1	1	0	0	0	0
A	5	1	1	1	1	1	1	1	1	0	0	0
A	p = 6	1	1	1	1	1	1	1	1	1	0	0
	p'= 7	0	0	0	0	0	1	1	1	1	1	0
	8	0	0	0	0	0	0	1	1	1	0	0

Relabeled $\longrightarrow$

U \ V	**4**	**3**	**2**	**5**	**1**	6	7	8	9	10	11
1	1	0	0	0	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0	0	0	0
3	1	1	1	1	0	0	0	0	0	0	0
k = 4	1	1	1	1	1	1	1	0	0	0	0
5	1	1	1	1	1	1	1	1	0	0	0
p = 6	1	1	1	1	1	1	1	1	1	0	0
p'= 7	0	0	0	0	0	1	1	1	1	1	0
8	0	0	0	0	0	0	1	1	1	0	0

Figure 6: Example relabeling process when A is left-aligned.

and

$$B' = \{\bar{u}_q, \ldots, \bar{u}_m\} - B. \tag{9}$$

The segments in A' (B', resp.) have no overlaps with $\bar{u}_{p'}$ ($\bar{u}_{q'}$, resp.).

A set of segments $S = \{s_1, \ldots, s_k\}$ is said to be *left-aligned* if $first(s_1) = \cdots = first(s_k)$. Similarly, S is *right-aligned* if $last(s_1) = \cdots = last(s_k)$. For convenience, define an empty set to be both left- and right-aligned. If S is both left- and right-aligned, then all segments in S are equal.

If each "$\subseteq$" in Eq. 2 can be changed to "$\preceq$" by relabeling the nodes, then the graph is forward-convex. The following lemma shows a case in which "$\subseteq$" can be easily converted to "$\preceq$". In this lemma, $\bar{u}_k$ is the first element of A and $\bar{u}_l$ is the last element of B; i.e.,

$$A = \{\bar{u}_k, \ldots, \bar{u}_p\} \quad \text{and} \quad B = \{\bar{u}_q, \ldots, \bar{u}_l\}. \tag{10}$$

Lemma 4 *If A is left-aligned, then the nodes from v_1 up to the last node of $\bar{u}_{k-1}$ can be relabeled so that $\bar{u}_1 \preceq \cdots \preceq \bar{u}_p$; the relabeling does not affect the "$\subseteq$" or "$\preceq$" relationships among $\bar{u}_p, \ldots, \bar{u}_m$. (See Fig. 5 for illustration.) Similarly, if B is right-aligned, then the nodes from the first node of $\bar{u}_{l+1}$ down to the last node of V can be relabeled in a way that makes $\bar{u}_q \preceq \cdots \preceq \bar{u}_m$ and that does not affect the "$\subseteq$" or "$\preceq$" relationships among $\bar{u}_1, \ldots, \bar{u}_q$.*

Proof. Assume A to be left-aligned. Also, assume $A' \neq \emptyset$ (or we immediately have $\bar{u}_1 \preceq \ldots \preceq \bar{u}_p$). Then, $first(\bar{u}_p) < first(\bar{u}_{p'})$. Let k be such that $A' = \{\bar{u}_1, \ldots, \bar{u}_{k-1}\}$ and $A = \{\bar{u}_k, \ldots, \bar{u}_p\}$. Since $\bar{u}_1 \subseteq \ldots \subseteq \bar{u}_k$, the nodes from v_1 to the last node of $\bar{u}_{k-1}$ can

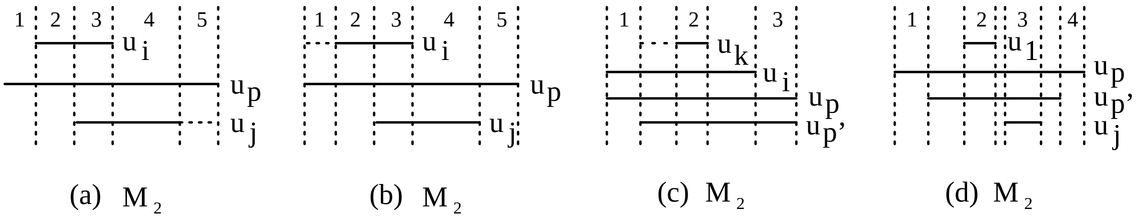


Figure 7: Cases where M contains a submatrix isomorphic to M_2.

be relabeled so that $\{\bar{u}_1, \ldots, \bar{u}_k\}$ becomes left-aligned (see Fig. 6 for illustration). This operation does not change the contents of $\bar{u}_k, \ldots, \bar{u}_m$. Thus, after relabeling, $\bar{u}_1, \ldots, \bar{u}_p$ are all left-aligned and, therefore, $\bar{u}_1 \preceq \ldots \preceq \bar{u}_p$. The "$\subseteq$" or "$\preceq$" relationships among $\bar{u}_p, \ldots, \bar{u}_m$ remain intact. $\square$

4.2 Case $p = q$

The next lemma establishes the "(c) $\Rightarrow$ (b)" part of Theorem 1 for the case $p = q$.

Lemma 5 *Let $G = (U, V, E)$ be a connected biconvex graph satisfying Eqs. 2, 4, 5. If $p = q$ in Eq. 2, then the following statements are equivalent.*

1. *G is forward-convex.*
2. *G contains no induced subgraph isomorphic to G_1 or G_2.*
3. *At least one of the following conditions is satisfied, where A and B are as defined in Eqs. 6 and 7.*
   (a) *For all i, j, either $\bar{u}_i \subseteq \bar{u}_j$ or $\bar{u}_i \supseteq \bar{u}_j$.*
   (b) *A is left-aligned and B is right-aligned.*
   (c) *A is right-aligned and B is left-aligned.*

Proof. $(1 \Rightarrow 2)$: This has been proved in Section 2.

$(2 \Rightarrow 3)$: Let M be the adjacency matrix of G, and let M_1 and M_2 be the matrices defined in Eq. 1. In light of Lemma 2, it suffices to show that if none of the three conditions is satisfied, then M contains a submatrix isomorphic to M_1 or M_2.

So, assume that none of the three conditions (a), (b), (c) is satisfied, and consider all possible cases.

Case 1. A is neither left- nor right-aligned. In this case A contains more than one element. Let $A = \{\bar{u}_k, \ldots, \bar{u}_p\}$. Then, $\{\bar{u}_k, \bar{u}_p\}$ is neither left- nor right-aligned. Since (a) is not true, there exist a pair (i, j), $i < p < j$, such that $\bar{u}_i \not\subseteq \bar{u}_j$ and $\bar{u}_i \not\supseteq \bar{u}_j$. Without loss of generality, we may assume $first(\bar{u}_i) \leq first(\bar{u}_j)$ (the other case is similar). Then, $first(\bar{u}_i) < first(\bar{u}_j)$ and $last(\bar{u}_i) < last(\bar{u}_j)$. However, $\bar{u}_i$ and $\bar{u}_j$ may or may not have intersection. If they do have intersection, then $\bar{u}_i$ and $u_{p'}$ also have intersection and, consequently, $k \leq i < p$. Consider four possible cases.

Subcase 1: $\bar{u}_i \cap \bar{u}_j \neq \emptyset$ and $\{\bar{u}_i, \bar{u}_p\}$ is not left-aligned. In this case, $\{\bar{u}_i, \bar{u}_p\}$ is not right-aligned either, as otherwise it would lead to a contradiction that $\bar{u}_i \supseteq \bar{u}_j$. Partitioning $\bar{u}_p$ into five subsets as shown in Fig. 7(a), one may construct from columns 1, 2, 3, 4 a submatrix

$$M' = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

that is isomorphic to M_2.

Subcase 2: $\bar{u}_i \cap \bar{u}_j \neq \emptyset$ and $\{\bar{u}_j, \bar{u}_p\}$ is not right-aligned (Fig. 7(b) illustrates this case). Again, from the figure (particularly, columns 2, 3, 4, 5) one readily sees the existence of a submatrix isomorphic to M_2.

Subcase 3: $\bar{u}_i \cap \bar{u}_j \neq \emptyset$, $\{\bar{u}_i, \bar{u}_p\}$ is left-aligned and $\{\bar{u}_j, \bar{u}_p\}$ is right-aligned. In this case, $\{\bar{u}_p, \bar{u}_{p'}\}$ is also right-aligned since $\bar{u}_p \supseteq \bar{u}_{p'} \supseteq \bar{u}_j$ (recall that $p' = p + 1 = q + 1$). Fig. 7(c) shows the relationship among the four segments $\bar{u}_k, \bar{u}_i, \bar{u}_p, \bar{u}_{p'}$. From these four rows and columns 1, 2 and 3, one may construct a submatrix isomorphic to M_2:

$$M' = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

Subcase 4: $\bar{u}_i \cap \bar{u}_j = \emptyset$. Since $\bar{u}_1 \subseteq \bar{u}_k \subseteq \bar{u}_j$, $\{\bar{u}_1, \bar{u}_p\}$ is neither left-aligned nor right-aligned and $\bar{u}_1 \cap \bar{u}_j = \emptyset$. Since $\bar{u}_p \neq \bar{u}_{p'}$, either $first(\bar{u}_p) < first(\bar{u}_{p'})$ or $last(\bar{u}_{p'}) < last(\bar{u}_p)$. If $first(\bar{u}_p) < first(\bar{u}_{p'})$, we have the case as illustrated in Fig. 7(d). A submatrix isomorphic to M_2 exists:

$$M' = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

The case where $last(\bar{u}_{p'}) < last(\bar{u}_p)$ is similar.

Case 2: B is neither left-aligned nor right-aligned. This case is symmetric to case 1.

Case 3: A is not left-aligned and B is not left-aligned. Assume that A is right-aligned and B is right-aligned; otherwise, it falls into either Case 1 or Case 2. Then, $A \cup B \cup \{u_p\}$ is right-aligned. But this is impossible because condition (a) is false.

Case 4: A is not right-aligned and B is not right-aligned. The proof is similar to Case 3.

($3 \Rightarrow 1$): If condition (a) is satisfied (Fig. 8(a)), then the nodes of U can be relabeled

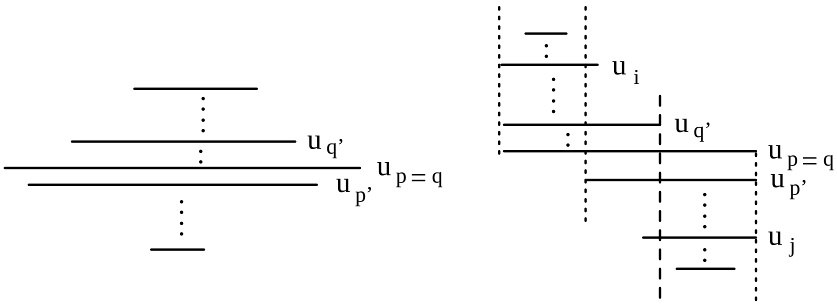


Figure 8: Two cases where G, with $p = q$, is forward-convex.

so that $\bar{u}_1 \subseteq \bar{u}_2 \subseteq \cdots \subseteq \bar{u}_m$. With this new labeling, Eq. 2 holds with $p = m$ and $A = \emptyset$. By Lemma 4, the nodes of V can be relabeled so that $\bar{u}_1 \preceq \bar{u}_2 \preceq \cdots \preceq \bar{u}_m$. This new labeling is forward-convex. If (b) holds (Fig. 8(b)), then by Lemma 4 a forward-convex labeling can be obtained by relabeling the nodes of V. Case (c) becomes case (b) if the nodes in U are relabeled in reverse order. □

4.3 Case $p < q$

Now consider the case $p < q$ in Eq. 2. Define $p'' > p$ to be the smallest index such that $\{\bar{u}_p, \bar{u}_{p''}\}$ is not right-aligned. Similarly, define $q'' < q$ to be the largest index such that $\{\bar{u}_q, \bar{u}_{q''}\}$ is not left-aligned. Note that $p < p' \leq p'' \leq q$ and $p \leq q'' \leq q' < q$. The next lemma completes the proof of Theorem 1 for the case $p < q$.

Lemma 6 *Let $G = (U, V, E)$ be a connected biconvex graph satisfying Eqs. 2, 4, 5. If $p < q$ in Eq. 2, then the following statements are equivalent.*

1. *G is forward-convex.*
2. *G contains no induced subgraph isomorphic to G_1 or G_2.*
3. *A and B satisfy the condition $(a1 \vee a2) \wedge (b1 \vee b2)$, where A and B are as defined in Eqs. 6 and 7 and a1, a2, b1, and b2 are the following statements:*

   (a1) A is left-aligned.

   (a2) A is right-aligned, $A = \{\bar{u}_1, \ldots, \bar{u}_p\}$, and $first(\bar{u}_1) \leq first(\bar{u}_{p''})$.

   (b1) B is right-aligned.

   (b2) B is left-aligned, $B = \{\bar{u}_q, \ldots, \bar{u}_m\}$, and $last(\bar{u}_m) \geq last(\bar{u}_{q''})$.

Proof. $(1 \Rightarrow 2)$: This has been proved in Section 2.

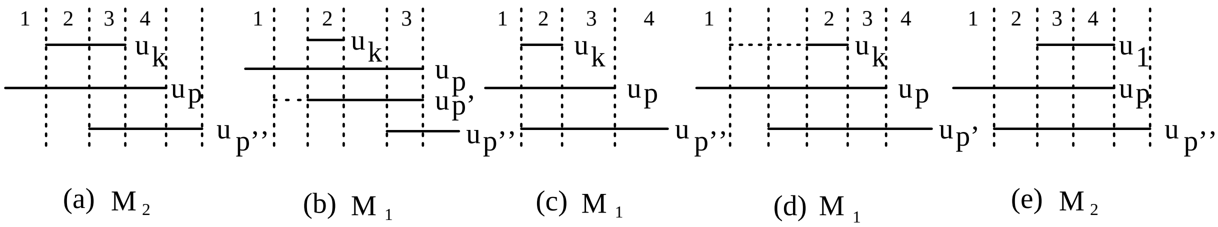


Figure 9: Cases where M contains a submatrix isomorphic to M_1 or M_2.

($2 \Rightarrow 3$): Let M be the adjacency matrix of G, and let M_1 and M_2 be the matrices defined in Eq. 1. We show that if a1 and a2 are both false, then M contains a submatrix isomorphic to M_1 or M_2. (The argument will be similar if b1 and b2 are both false.)

So, assume A is not left-aligned and at least one of these three conditions is satisfied: i) A is not right-aligned, ii) $A \neq \{\bar{u}_1, \ldots, \bar{u}_p\}$, iii) $first(\bar{u}_1) > first(\bar{u}_{p''})$. Let k be the smallest index such that $\bar{u}_k \in A$. Thus, $A = \{\bar{u}_k, \ldots, \bar{u}_p\}$ and $A' = \{\bar{u}_1, \ldots, \bar{u}_{k-1}\}$.

Case 1: A is not right-aligned In this case, A is neither left- nor right-aligned and so $\bar{u}_k$ and $\bar{u}_p$ are neither left- nor right-aligned. There are four possibilities:

Subcase 1: $\{\bar{u}_p, \bar{u}_{p'}\}$ is right-aligned and $first(\bar{u}_k) < first(\bar{u}_{p'})$. The three segments $\bar{u}_k$, $\bar{u}_p$, $\bar{u}_{p'}$ have same relationships as the three segments $\bar{u}_i$, $\bar{u}_p$ and $\bar{u}_j$ of Fig. 7(a). Hence, M contains a submatrix isomorphic to M_2.

Subcase 2: $\{\bar{u}_p, \bar{u}_{p'}\}$ is right-aligned and $first(\bar{u}_{p'}) \leq first(\bar{u}_k) < first(\bar{u}_{p''})$. Consider the case $first(\bar{u}_{p''}) \leq last(\bar{u}_k)$. Fig. 9(a) illustrates the case. Obviously, M contains a submatrix isomorphic to M_2.

On the other hand, assume that $first(\bar{u}_{p''}) > last(\bar{u}_k)$. Fig. 9(b) illustrates the case. It can be checked that M contains a submatrix isomorphic to M_1.

Subcase 3: $\{\bar{u}_p, \bar{u}_{p'}\}$ is right-aligned and $first(\bar{u}_{p'}) \leq first(\bar{u}_{p''}) \leq first(\bar{u}_k)$. Since $\bar{u}_p \neq \bar{u}_{p'}$, $first(\bar{u}_p) < first(\bar{u}_{p'})$. Out of the three segments u_k, u_p, $u_{p''}$ (see Fig. 9(c) for illustration), a submatrix isomorphic to M_1 can be easily constructed.

Subcase 4: $\{\bar{u}_p, \bar{u}_{p'}\}$ is not right-aligned. Fig. 9(d) illustrates the relationship among the three segments $\bar{u}_k$, $\bar{u}_p$ and $\bar{u}_{p'}$. One readily sees that M contains a submatrix isomorphic to M_1.

Case 2: $A \neq \{\bar{u}_1, \ldots, \bar{u}_p\}$ (i.e., $1 < k < p$). Assume A is right-aligned (or case 1 applies). The four segments u_1, u_k, u_p and $u_{p''}$ have the same relationships as the four segments u_k, $u_{p'}$, u_p and $u_{p''}$ in Fig. 9(b). Therefore, a submatrix isomorphic to M_1 can be obtained.

Case 3: $first(\bar{u}_1) > first(\bar{u}_{p''})$. Assume A is right-aligned and $A = \{\bar{u}_1, \ldots, \bar{u}_p\}$ (or we have case 1 or 2). The three segments u_1, u_p, and $u_{p''}$, as illustrated in Fig. 9(e), indicates the existence of a submatrix isomorphic to M_1.

($3 \Rightarrow 1$): Conditions a1 and a2 are illustrated in Fig. 10. Assume $(a1 \vee a2) \wedge (b1 \vee b2)$ is true. There are four possible cases: $a1 \wedge b1$, $a2 \wedge b1$, $a1 \wedge b2$, $a2 \wedge b2$.

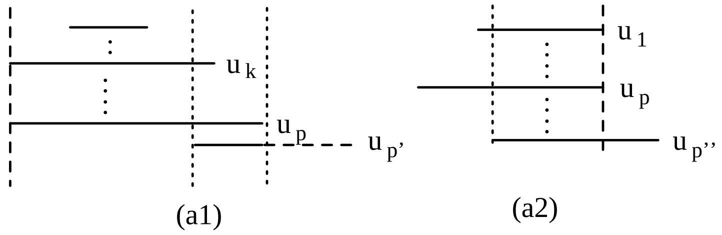


Figure 10: Two cases where G, with $p < q$, is isomorphically forward-convex.

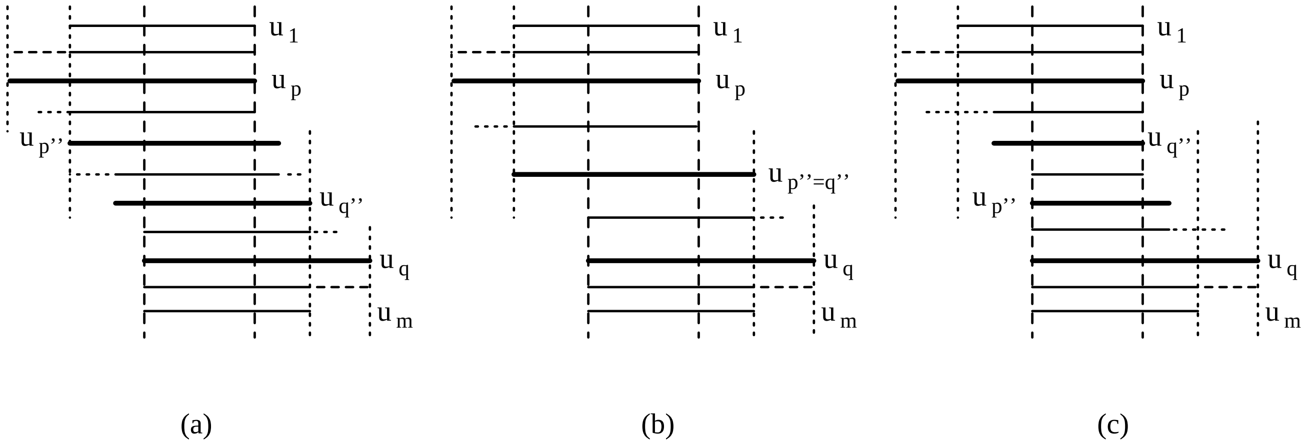


Figure 11: Three cases that A satisfies a2 and B satisfies b2.

Case 1: $a1 \wedge b1$. It follows from Lemma 4 that the nodes in $\bar{u}_p - \bar{u}_q$ and the nodes in $\bar{u}_q - \bar{u}_p$ can be respectively relabeled so that $\bar{u}_1 \preceq \cdots \preceq \bar{u}_p$ and $\bar{u}_q \preceq \cdots \preceq \bar{u}_m$.

Case 2: $a2 \wedge b1$. It follows from Eq 1, condition a2 and the definition of p'' that $\{\bar{u}_1, \ldots, \bar{u}_{p''-1}\}$ is right-aligned and, therefore, the nodes $u_1, \ldots, u_{p''-1}$ can be relabeled so that $\bar{u}_1 \preceq \cdots \preceq \bar{u}_{p''-1} \preceq \bar{u}_{p''}$. This relabeling does not affect condition b2 because $p'' \leq q$. It is thus possible to relabel the nodes in $\bar{u}_q - \bar{u}_p$ to satisfy $\bar{u}_q \preceq \cdots \preceq \bar{u}_m$.

Case 3: $a1 \wedge b2$. This case is symmetric to case 2.

Case 4: $a2 \wedge b2$. In this case, $\{\bar{u}_1, \ldots, \bar{u}_{p''-1}\}$ is right-aligned and $\{\bar{u}_{q''+1}, \ldots, \bar{u}_m\}$ left-aligned. If $p'' \leq q''$ (as illustrated in Fig. 11(a)-(b)), it is easy to relabel $u_1, \ldots, u_{p''-1}$ as well as $u_{q''+1}, \ldots, u_m$ to make the graph forward-convex. If $p'' > q''$ (see Fig. 11(c)), then the segments $\bar{u}_{q''+1}, \ldots, \bar{u}_{p''-1}$ are both left- and right-aligned. It is possible to relabel the nodes $u_1, \ldots, u_{q''}$ and the nodes $u_{p''}, \ldots, u_m$ to make G forward-convex. $\square$

4.4 The Algorithm and Its Complexity

Lemmas 5 and 6 together prove the "(c) $\Rightarrow$ (b)" part of Theorem 1. The proofs of these lemmas also provide an algorithm for recognizing forward-convex graphs and generating forward-convex labelings. For clarity, we summarize the algorithm in the following.

Algorithm Recognizing and Labeling Forward-Convex Graphs

Input: A connected bipartite graph $G = (U, V, E)$ with labeling $U = \{u_1, \dots, u_m\}$ and $V = \{v_1, \dots, v_n\}$.

Output: A forward-convex labeling for G if it is forward-convex.

Begin

1. Apply Booth and Lueker's algorithm twice, once on U and once on V, and generate a biconvex labeling. If this cannot be done then print "G is not forward-convex" and return.

2. Check if the following condition (i.e., Eq. 2) is satisfied:

$$\emptyset = \bar{u}_0 \subseteq \bar{u}_1 \subseteq \cdots \subseteq \bar{u}_p \preceq \cdots \preceq \bar{u}_q \supseteq \cdots \supseteq \bar{u}_m \supseteq \bar{u}_{m+1} = \emptyset$$

   If not, relabel the nodes of U in reversed order so that the condition holds.

3. Choose the p in Eq. 2 to be as large as possible, and choose q to be as small as possible.

4. Let $p' > p$ be the smallest index such that $\bar{u}_{p'} \neq \bar{u}_p$, and let $q' < q$ be the largest index such that $\bar{u}_{q'} \neq \bar{u}_q$.

5. Compute A and B as defined in Eqs. 6 and 7, and suppose this yields

$$A = \{\bar{u}_k, \dots, \bar{u}_p\} \quad \text{and} \quad B = \{\bar{u}_q, \dots, \bar{u}_l\}.$$

6. If $p = q$, then test if any of the three conditions as described in statement 3 of Lemma 5 is satisfied:

   case

   - for all i and j $(1 \leq i, j \leq m)$ it holds either $\bar{u}_i \subseteq \bar{u}_j$ or $\bar{u}_i \supseteq \bar{u}_j$:
     relabel the nodes of U so that $\bar{u}_1 \subseteq \bar{u}_2 \subseteq \cdots \subseteq \bar{u}_m$;
     apply Lemma 4 and relabel V so that $\bar{u}_1 \preceq \bar{u}_2 \preceq \cdots \preceq \bar{u}_m$;
   - A is left-aligned and B is right-aligned:
     apply Lemma 4 and relabel V so that $\bar{u}_1 \preceq \bar{u}_2 \preceq \cdots \preceq \bar{u}_m$;
   - A is right-aligned and B is left-aligned:
     reverse the order of U, and then
     apply Lemma 4 and relabel V so that $\bar{u}_1 \preceq \bar{u}_2 \preceq \cdots \preceq \bar{u}_m$
   - else:
     print "G is not forward-convex" and return

   end case

7. If $p < q$, then let $p'' > p$ be the smallest index such that $\{\bar{u}_p, \bar{u}_{p''}\}$ is not right-aligned, and let $q'' < q$ be the largest index such that $\{\bar{u}_q, \bar{u}_{q''}\}$ is not left-aligned. Then check if A satisfies (a1) or (a2) and B satisfies (b1) or (b2) of Lemma 6:

case

- $a1 \wedge b1$:
  apply Lemma 4 and relabel the nodes in $\bar{u}_p - \bar{u}_q$ so that $\bar{u}_1 \preceq \cdots \preceq \bar{u}_p$;
  apply Lemma 4 and relabel the nodes in $\bar{u}_q - \bar{u}_p$ so that $\bar{u}_q \preceq \cdots \preceq \bar{u}_m$;
- $a2 \wedge b1$:
  relabel nodes $u_1, \ldots, u_{p''-1}$ so that $\bar{u}_1 \preceq \cdots \preceq \bar{u}_{p''-1} \preceq \bar{u}_{p''}$;
  relabel the nodes in $\bar{u}_q - \bar{u}_p \subseteq V$ so that $\bar{u}_q \preceq \cdots \preceq \bar{u}_m$.
- $a1 \wedge b2$:
  relabel nodes $u_{q''+1}, \ldots, u_m$ so that $\bar{u}_{q''} \preceq \bar{u}_{q''+1} \preceq \cdots \preceq \bar{u}_{p_m}$;
  relabel the nodes in $\bar{u}_p - \bar{u}_q \subseteq V$ so that $\bar{u}_1 \preceq \cdots \preceq \bar{u}_p$;
- $a2 \wedge b2$:
  if $p'' \leq q''$ (as illustrated in Fig. 11(a)-(b)) **then**
    relabel $u_1, \ldots, u_{p''-1}$ and $u_{q''+1}, \ldots, u_m$ so that
    $\bar{u}_1 \preceq \bar{u}_2 \preceq \cdots \preceq \bar{u}_m$;
  else /* $p'' > q''$ as illustrated in Fig. 11(c) */
    relabel nodes $u_1, \ldots, u_{q''}$ and nodes $u_{p''}, \ldots, u_m$ so that
    $\bar{u}_1 \preceq \bar{u}_2 \preceq \cdots \preceq \bar{u}_m$;
- else:
  print "G is not forward-convex"

end case

End

Assuming that the given graph G is represented by adjacency lists (i.e., nonzero entries of the adjacency matrix), we analyze the time complexity of the above algorithm as follows. Step 1 needs $O(|U| + |V| + |E|)$ time as required by the Booth-Lueker algorithm. Step 2 can be done in $O(|U|)$ time if one first obtains the boundary of the 1's in the matrix (i.e., the left margin l_i and right margin r_i of the 1's on each row i); these margins can be computed in $O(|E|)$ time. It takes only $O(|U|)$ time to identify p, q, p', and q' in steps 3 and 4. With l_i's and r_i's known, computing A and B in step 5 takes no more than $O(|U|)$ time. Checking condition (a) of Lemma 5 in step 6 can be done in $O(|U|)$ time; here we take the advantage of Eq. 2 and do something similar to merging; it's not necessary to compare every pair $(\bar{u}_i, \bar{u}_j)$. Conditions (b) and (c) can also be tested in $O(|U|)$ time. The relabeling process takes at most $O(|U| + |V|)$ time for condition (a) and $O(|V|)$ time for conditions (b) and (c). In step 7, checking conditions (a1), (a2), (b1) and (b2) of Lemma 6 can be done in $O(|U|)$ time. The relabeling process takes no more than $O(\max\{|U|,|V|\})$ time. So, the overall time complexity is $O(|U|+|V|+|E|)$.

5 Minimizing the Buffer Size: An Application

Consider a join operation of two relations in a relational database system. Each relation is a collection of *tuples* and is stored in the disk as a number of *pages*. The *page connectivity graph* of a join operation is a graph in which each node represents a data page and there is an edge between two nodes if and only if their corresponding data pages possibly contain tuples that need to be joined [2, 6, 5]. Only those pages (nodes) with at least an edge in the graph need to be accessed, and two data pages need to be in the main memory simultaneously if they are connected by an edge. Given a page connectivity graph G, the minimum buffer size problem is to determine the minimum amount of buffer required to perform the join operation represented by G. This problem is NP-hard if G is an arbitrary graph [2]. As was mentioned in the introduction, the page connectivity graphs for many join operations are forward-convex [11]. This motivates us to look for an efficient algorithm for the minimum buffer size problem with problem instances restricted to forward-convex graphs. We report a linear time algorithm here. Several other NP-complete problems are known to be polynomially solvable for bipartite permutation graphs [8].

We start our discussion by formally formulating the problem. Let $G=(U,V,E)$ be the page connectivity graph of a join operation. Without loss of generality, assume that G contains no isolated nodes as such nodes can be deleted. A *retrieved-once sequence* S of G is a permutation of $U \cup V$, in which each node appears exactly once. In order to perform the join operation, suppose that the pages of the relations are fetched, in the order of S, to a buffer in main memory. Once a page is fetched, it is retained in the buffer until it is no longer needed. Thus, each page is retrieved exactly once. The *cost* of a retrieved-once sequence S, denoted $Cost(S)$, is the minimum buffer size (in terms of number of pages) required to perform the operation in question if pages are read into the main memory buffer according to the sequence S. The *minimum buffer size problem* is to find a retrieved-once sequence of least cost. The problem is more formally formulated in the following.

Let $S=(w_1,\ldots,w_{m+n})$, where $m=|U|$ and $n=|V|$, be a retrieved-once sequence of G. Define $S_i=(w_1,\ldots,w_i)$ and $\bar{S}_i=(w_{i+1},\ldots,w_{m+n})$ for $0 \le i \le m+n$. Define $Rem(S_i)=\{w : w \in S_i \text{ and } \exists w' \in \bar{S}_i \text{ such that } (w,w') \in E\}$. After the pages in S_i are retrieved and the join among these pages performed, the pages in $Rem(S_i)$ must *remain* in the buffer as they still need to join with some pages in $\bar{S}_i$ which are yet to be fetched. Note that $Rem(S_0) = \emptyset$. Define $Mem(S_i) = Rem(S_{i-1}) \cup \{w_i\}$ for $1 \leq i \leq m+n$. Thus, $Mem(S_i)$ is the set of nodes (pages) residing in the buffer at the time page w_i is just read into the buffer. With these notations, the minimum buffer size problem can be formally defined as follows.

Problem Definition. Let $G = (U, V, E)$ be a bipartite graph, where $m = |U|$ and $n = |V|$. The *minimum buffer size problem* is to sort the nodes in $U \cup V$ into a sequence $S = (w_1, \ldots, w_{m+n})$ so as to minimize

$$Cost(S) = max\{|Mem(S_i)| : 1 \leq i \leq m+n\}.$$

As remarked earlier, this problem is NP-hard if G is an arbitrary graph [2]. It can be solved efficiently if the page connectivity graph is forward-convex, which is the main result of this section.

We shall assume that the given page connectivity graph is not only forward-convex but also connected. This assumption does not lose any generality because the connected components of a forward-convex graph are each forward-convex; individually generating a least-cost retrieved-once sequence for each connected component and then putting them together, one after another, immediately yields a least-cost retrieved-once sequence for the whole graph. Thus, we focus on connected forward-convex graphs.

5.1 Algorithm

Let $G = (U, V, E)$ be a connected forward-convex graph, with forward-convex labeling $U = \{u_1, \ldots, u_m\}$ and $V = \{v_1, \ldots, v_n\}$, and adjacency matrix $M(1..m, 1..n)$. Let $S = (w_1, w_2, \ldots, w_{m+n})$ be a retrieved-once sequence.

Definition 2 A retrieved-once sequence $S = (w_1, \ldots, w_{m+n})$ is said to be *regular* if the following three conditions are satisfied:

1. S starts with u_1 and v_1; i.e., $(w_1, w_2) = (u_1, v_1)$.
2. The nodes of U appear in S in ascending order, and so do the nodes of V; i.e., $(u_1, u_2, \ldots, u_m)$ and $(v_1, v_2, \ldots, v_n)$ are each a subsequence of S.
3. For each i, $1 < i \leq m+n$, there is at least one edge between w_i and S_{i-1}. (Thus, after page w_i is read into the buffer, it is immediately used to join with some other pages already in the buffer.)

The importance of regular sequences is that in search for a least-cost retrieved-once sequence, we can confine ourselves to regular sequences, due to the following theorem, which we choose to prove later in Section 5.2 in order not to interrupt the development of the algorithm.

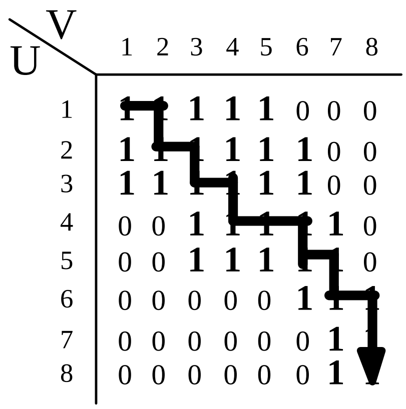


Figure 12: Example regular path.

Theorem 3 *For any retrieved-once sequence S', there exists a regular sequence S such that $Cost(S) \leq Cost(S')$.*

It is convenient to think of a regular sequence as a path on the adjacency matrix M. For instance, the sequence $S = (u_1, v_1, v_2, u_2, v_3, u_3, v_4, u_4, v_5,\ v_6, u_5, v_7, u_6, v_8, u_7, u_8)$ may be represented by the path of Fig. 12. The path has a ladder shape stretching from the left-top corner to the right-bottom. Such paths are formally defined in the following.

Definition 3 A *regular path* on M is a sequence of entries $P = ((x_1, y_1), \ldots, (x_{m+n-1}, y_{m+n-1}))$ satisfying the following conditions:

1. $M(x_i, y_i) = 1$ for each i, $1 \leq i \leq m+n-1$.

2. $\{x_1, \ldots, x_{m+n-1}\} = \{1, \ldots, m\}$ and $\{y_1, \ldots, y_{m+n-1}\} = \{1, \ldots, n\}$.

3. $1 = x_1 \leq \ldots \leq x_{m+n-1} = m$ and $1 = y_1 \leq \ldots \leq y_{m+n-1} = n$.

It's not hard to see that each regular path corresponds to a regular sequence and vice versa. Define the cost of a regular path to be the same as its corresponding sequence. Then a least-cost regular path translates to a least-cost retrieved-once sequence.

Now let's see how to compute the cost of a regular path. Let $S = (w_1, w_2, \ldots, w_{m+n})$ be a regular sequence and $P = ((x_1, y_1), \ldots, (x_{m+n-1}, y_{m+n-1}))$ its corresponding path. Each subpath $P_i = ((x_1, y_1), \ldots, (x_i, y_i))$ of P corresponds to a subsequence $S_{i+1} = (w_1, \ldots, w_{i+1})$ of S. For each point $p_i = (x_i, y_i)$ on P, define $R(p_i) = |Rem(S_{i+1})|$. If $m + n > 2$ (i.e., the given graph has more than one edge), then

$$\begin{aligned} Cost(P) &= Cost(S) \\ &= \max\{|Mem(S_i)| : 1 \leq i \leq m+n\} \end{aligned}$$

$$
\begin{aligned}
&= 1 + \max\{|Rem(S_i)| : 0 \le i \le m+n-1\} \\
&= 1 + \max\{R(p_i) : 1 \le i \le m+n-2\} \\
&= 1 + \max\{R(p) : p \text{ is a point on } P\} \qquad (11)
\end{aligned}
$$

Although we have defined $R(p_i)$ through a path, its value actually is determined solely by the point p_i and can be easily computed without referring to any particular path. For any point (x, y) on M such that $M(x, y) = 1$, $R(x, y)$ is the number of pages among $u_1, \ldots, u_x$ and $v_1, \ldots, v_y$ that still need to be retained in the buffer. Thus, $R(x, y) = |X'| + |Y'|$ where $X' = \{x' : M[x', y+1] = 1\}$ and $Y' = \{y' : M[x+1, y'] = 1\}$. Or, equivalently,

$$R(x, y) = \max\{0, x - t_{y+1} + 1\} + \max\{0, y - l_{x+1} + 1\} \qquad (12)$$

where l_i, $1 \le i \le m$, is the position of the leftmost 1 on row i (i.e., the smallest integer j such that $M(i, j) = 1$) and, similarly, t_j, $1 \le j \le n$, is the position of the topmost 1 on column j (i.e., the smallest integer i such that $M(i, j) = 1$); $l_{m+1} = t_{n+1} = \infty$. For instance, the point $(3, 2)$ in Fig. 12 has $R(3, 2) = (3 - 1 + 1) + 0 = 3$.

A path from (x, y) to (m, n) on M is said to be a *regular sub-path* if it is a sub-path of a regular path. The cost of a regular sub-path P is defined as $Cost(P) = 1 + \max\{R(p) : p \text{ a point on } P\}$. This definition is consistent with Eq. 11. For any entry (x, y) such that $M(x, y) = 1$, let $mcost(x, y)$ denote the cost of a minimum regular sub-path from (x, y) to (m, n), whose cost is minimum among all regular sub-path from (x, y) to (m, n). Note that $mcost(1, 1)$ gives the cost of a least-cost regular path. One readily sees that the function $mcost(x, y)$ satisfies the following equation:

$$
mcost(x, y) = \begin{cases}
1 & \text{if } (x, y) = (m, n) \\
\infty & \text{if } x > m \text{ or } y > n \text{ or } M[x, y] = 0 \\
\max\{R(x, y) + 1, \min\{mcost(x, y+1), mcost(x+1, y)\}\} & \text{otherwise.}
\end{cases}
$$

The following procedure computes a least-cost regular path based on the above recurrence equation.

Procedure Least-cost Regular Path

Input: A forward-convex graph $G = (U, V, E)$ with forward-convex labeling $U = \{u_1, \ldots, u_m\}$ and $V = \{v_1, \ldots, v_n\}$.

Output: a least-cost regular path.

1. Find the boundary of the 1's in M. Specifically, for each row i, find the left margin l_i and right margin r_i of the 1's on row i. Similarly, for each column j, find the top margin t_j and bottom margin b_j of the 1's on column j.

2. Compute $R(x, y)$ for all (x, y) such that $(u_x, v_y) \in E$, using Eq. 12.

3. Compute the function $mcost(x, y)$ as follows.

   for $i := 1$ **to** m **do** $mcost(i, r_i + 1) := \infty$;
   for $j := 1$ **to** n **do** $mcost(b_j + 1, j) := \infty$;
   for $i := m$ **to** 1 **do**
       for $j := r_i$ **to** l_i **do**
           if $(i, j) = (m, n)$ **then**
               $mcost(i, j) := 1$
           else if $mcost(i, j + 1) \leq mcost(i + 1, j)$ **then**
               $mcost(i, j) := \max\{R(i, j) + 1, mcost(i, j + 1)\}$
               $next(i, j) := (i, j + 1)$
           else
               $mcost(i, j) := \max\{R(i, j) + 1, mcost(i + 1, j)\}$
               $next(i, j) := (i + 1, j)$

4. Print $(1, 1)$, $next(1, 1)$, $next(next(1, 1))$, $\ldots$, until (m, n) is reached.

End.

The given graph may be represented by an adjacency matrix or by adjacency lists. In both cases, the time complexity of this algorithm is $O(|E|)$, where $|E|$ is the number of edges in the graph (or the number of 1's in M). This is obtained as follows. In step 1 there is no need to search the entire matrix should G be represented by its adjacency matrix; starting with entry (1,1), it suffices to visit the 1's in the matrix and the 0's that are adjacent to at least a 1; this takes $O(|E|)$ time. Step 2 evidently needs no more than $O(|E|)$ time as each computation of Eq. 12 takes only $O(1)$ time. Step 3 requires $O(|U| + |V| + |E|) = O(|E|)$ time, since $|U| \leq |E|$ and $|V| \leq |E|$. Step 4 obviously can be done in $O(|E|)$ time. So the overall time complexity is $O(|E|)$.

The above algorithm always produces optimal solutions for forward-convex graphs. In [2] there are two heuristic algorithms that produce near-optimal solutions for forward-convex as well non-forward-convex graphs.

5.2 Proof of Theorem 2

We now prove Theorem 3, which was stated and used in Section 5.1. To this aim, we establish some properties of forward-convex graphs and retrieved-once sequences.

Definition 4 A bipartite graph $G' = (U', V', E')$ is said to be *complete* if $(u, v) \in E'$ for all $u \in U'$ and $v \in V'$.

In the following lemmas, $G = (U, V, E)$ is a connected forward-convex graph with forward-convex labeling $U = \{u_1, \ldots, u_m\}$ and $V = \{v_1, \ldots, v_n\}$.

Lemma 7 *If* $(u_a, v_b) \in E$*, then the subgraph of* G *induced by* $U' \cup V'$ *is a complete graph, where* $U' = \{u_{a'} : a' \geq a$ *and* $(u_{a'}, v_b) \in E\}$ *and* $V' = \{v_{b'} : b' \geq b$ *and* $(u_a, v_{b'}) \in E\}$.

Proof. Since the labeling is forward-convex, both U' and V' are each a segment. Let $U' = \{u_a, \ldots, u_c\}$ and $V' = \{v_b, \ldots, v_d\}$. For each pair of nodes $u_i \in U'$ and $v_j \in V'$, since $(u_a, v_j) \in E$ and $(u_i, v_b) \in E$, it follows from Corollary 1 that $(u_i, v_j) \in E$. The lemma is proved. □

For convenience, define $\hat{S}_i = Rem(S_i) \cup \bar{S}_i$. The following lemma shows that at any point of a regular sequence, the nodes (pages) remaining in the buffer together with the pages yet to be fetched form a forward-convex subgraph.

Lemma 8 *If* S *is a regular sequence, then for each* i*,* $1 \leq i \leq |U| + |V|$*, the subgraph induced by* $\hat{S}_i$ *is connected and forward-convex; furthermore, the induced labeling on* $\hat{S}_i$ *has the forward-convex property.*

Proof. Since the labeling of G has the forward-convex property and since S retrieves the nodes of U (V) in ascending order, it is not hard to see that the nodes of U (V) are released from the buffer also in ascending order. Thus, for each i, $\hat{S}_i = \{u_j, \ldots, u_m\} \cup \{v_k, \ldots, v_n\}$ for some j and k. The subgraph induced by these nodes is evidently connected and forward-convex. □

For any non-empty set $W \subseteq U \cup V$ and a retrieved-once sequence S, we write $W \sqsubseteq S$ if there exists an i, $1 \leq i \leq m + n$, such that $W \subseteq Mem(S_i)$. The following two lemmas further show some important properties regarding complete subgraphs and retrieved-once sequence for any graph.

Lemma 9 *If* $G' = (U', V', E')$ *is a complete subgraph of* G*, then every retrieved-once sequence* S *of* G *satisfies either* $\{u'\} \cup V' \sqsubseteq S$ *for every* $u' \in U'$ *or* $\{v'\} \cup U' \sqsubseteq S$ *for every* $v' \in V'$.

Proof. Since S is a retrieved-once sequence, each retrieved node (page) cannot be released from the buffer until all of its adjacent nodes have been accessed. Let x be the first node in $U' \cup V'$ to be released from the buffer.

Case 1. $x \in U'$. Since $G' = (U', V', E')$ is complete and x is the first-released node in $U' \cup V'$, there exists an i, $0 < i \leq |U| + |V|$, such that $\{x\} \cup V' \subseteq Mem(S_i)$. After that, no node in V' can be released until every other node in U' has been retrieved. That means, $\{u'\} \cup V' \sqsubseteq S$ for every $u' \in U'$.

Case 2. $x \in V'$. In this case, a similar argument will show $\{v'\} \cup U' \sqsubseteq S$ for every $v' \in V'$. □

Lemma 10 *Let $G' = (U', V', E')$ be a complete subgraph of G and S a retrieved-once sequence. If there exist two subsets $U'' \subseteq U'$ and $V'' \subseteq V'$ such that $U'' \sqsubseteq S$ and $V'' \sqsubseteq S$, then $U'' \cup V'' \sqsubseteq S$.*

Proof. Assume $U'' \sqsubseteq S$ and $V'' \sqsubseteq S$, where $U'' \subseteq U'$ and $V'' \subseteq V'$. Let $U'' \subseteq Mem(S_i)$ and $V'' \subseteq Mem(S_j)$. Assume $i \neq j$, or $U'' \cup V'' \subseteq Mem(S_i)$ and the lemma is proved. Without lose of generality, we further assume $i < j$. Since G' is complete and S is retrieved-once, it must be that $U'' \subseteq Rem(S_j)$. Hence, $U'' \cup V'' \subseteq Mem(S_j) \sqsubseteq S$. □

We are now ready to prove the theorem.

Theorem 3 *Let $G = (U, V, E)$ be a connected forward-convex graph with forward-convex labeling $U = \{u_1, \ldots, u_m\}$ and $V = \{v_1, \ldots, v_n\}$. For any retrieved-once sequence S of G, there exists a regular sequence R of G such that $Cost(R) \leq Cost(S)$.*

Proof. Let S be any retrieved-once sequence. We shall construct a regular sequence R such that $Cost(R) \leq Cost(S)$. Starting with $R^0 = (u_1, v_1)$, we shall extend it to R^1, R^2, and so on, until all nodes are included. The resulting sequence, say R^k, will be our R.

Let $R^0 = (u_1, v_1)$. For $i = 1, 2, \ldots$, as long as $|R^{i-1}| \neq m + n$, construct R^i as follows.

1. If $Rem(R^{i-1})$ is a subset of U (or V), then let $R^i = R^{i-1} \circ y$, where y is the first node of V (or U, respectively) that is not in R^{i-1}. (The basic idea is that if the pages currently in the buffer are all from the same relation then fetch the next page from the other relation.)

2. Otherwise, let u_i^f (v_i^f, respectively) be the first node in $Rem(R^{i-1})$ that belongs to U (V, respectively). Let x_i^f be either u_i^f or v_i^f such that $\{x_i^f\} \cup (Adj(x_i^f) \cap \hat{R}^{i-1}) \sqsubseteq S$, and let X be the node set containing x_i^f. Define

$$R^i = R^{i-1} \circ A_i \circ B_i$$

where

$$A_i = Adj(x_i^f) - R^{i-1}$$

and

$$B_i = \{x' \in X : Adj(x') \subseteq Adj(x_i^f)\} - R^{i-1}.$$

(Comments: The pages in A_i are those needed by page x_i^f but not yet fetched. So we go ahead to get them. After these pages are read into the buffer and the join performed, page x_i^f can be released from the buffer. At this point, the pages needed by any $x' \in B_i$ are all available in the buffer. Thus, the pages in B_i can be read into the buffer one by one and immediately released from the buffer. As a result,

$Mem(R^{i-1} \circ A_i) = Rem(R^{i-1}) \cup A_i$ and if $B_i = \{b_1, \ldots, b_q\}$ then $Mem(R^{i-1} \circ A_i \circ b_1 \circ \cdots \circ b_j) \subseteq Rem(R^{i-1}) \cup A_i \cup \{b_j\} - \{x_i^f\}$ for each j, $1 \leq j \leq q$.

Now we show by induction that R^i, as defined above, is meaningful whenever $|R^{i-1}| < m + n$. R^0 is obviously well-defined, and it is a regular subsequence (i.e., a prefix of some regular sequence). Suppose that R^{i-1} has been well-defined and is a regular subsequence with $|R^{i-1}| < m + n$.

Case 1: R^i is defined in step 1. In this case, since G is connected, $Rem(R^{i-1}) \neq \emptyset$ and y exists. Thus R^i is well-defined. Since the labeling of G is forward-convex, by Corollary 1 there must be an edge between y and $Rem(R^{i-1})$ and, hence, R^i is a regular subsequence.

Case 2: R^i is defined in step 2. By Lemma 8, the subgraph of G induced by the nodes of $\hat{R}^{i-1}$ is a connected forward-convex graph. Therefore, $(u_i^f, v_i^f) \in E$ and, as a result of Lemma 7, the subgraph of G induced by $(Adj(u_i^f) \cup Adj(v_i^f)) \cap \hat{R}^{i-1}$ is a complete graph. Hence, by Lemma 9, x_i^f exists and R^i is well defined. That R^i is a regular subsequence can be readily checked.

Assume that the above procedure yields $R^0 \subseteq R^1 \subseteq \cdots \subseteq R^k$. Write R^k as R. As a regular subsequence containing $m + n$ elements, R is a regular sequence.

Next, we show by induction that for each prefix R' of R, $|Mem(R')| \leq Cost(S)$, from which it follows that $Cost(R) \leq Cost(S)$.

It's evident that $Mem(R^0) \sqsubseteq S$, and each prefix R' of R^0 satisfies $|Mem(R')| \leq Cost(S)$. As the induction hypothesis, assume $Mem(R^{i-1}) \sqsubseteq S$ and each of its prefix R' satisfies $|Mem(R')| \leq Cost(S)$. We have to show that $Mem(R^i) \sqsubseteq S$ and each of its prefix R' satisfies $|Mem(R')| \leq Cost(S)$.

Case 1: $R^i = R^{i-1} \circ y$. It follows directly from Lemma 10 and the induction hypothesis $Mem(R^{i-1}) \sqsubseteq S$ that $Mem(R^i) = Rem(R^{i-1}) \cup \{y\} \subseteq Mem(R^{i-1}) \cup \{y\} \sqsubseteq S$.

Case 2: $R^i = R^{i-1} \circ A_i \circ B_i$. By the induction hypothesis, $Rem(R^{i-1}) \subseteq Mem(R^{i-1}) \sqsubseteq S$. From definitions of x_i^f and A_i, one readily sees $A_i = Adj(x_i^f) \cap \bar{R}^{i-1} \subseteq Adj(x_i^f) \cap \hat{R}^{i-1} \sqsubseteq S$. It can be checked that $Rem(R^{i-1})$ and A_i are contained in a complete subgraph of G. Thus, by Lemma 10, we have

$$Rem(R^{i-1}) \cup A_i \sqsubseteq S \tag{13}$$

which implies that $|Mem(R')| \leq Cost(S)$ for each prefix R' of $R^{i-1} \circ A_i$. Let $B_i = \{b_1, \ldots, b_q\}$. Observe that for $1 \leq j \leq q$,

$$Mem(R^{i-1} \circ A_i \circ b_1 \circ \cdots \circ b_j) \subseteq Rem(R^{i-1}) \cup A_i \cup \{b_j\} - \{x_i^f\}. \tag{14}$$

Eqs. 13 and 14 imply $Mem(R^i) \sqsubseteq S$ (by Lemma 10). Furthermore, for each j, $1 \leq j \leq q$, $|Mem(R^{i-1}) \circ A_i \circ b_1 \circ \cdots \circ b_j)| \leq |Rem(R^{i-1} \cup A_i)| \leq Cost(S)$. This completes the induction step. $\square$

6 Conclusion

We have defined the class of forward-convex graphs and showed it to be identical to the class of bipartite permutation graphs. We also developed an algorithm that tests whether a bipartite graph is forward-convex and, if so, generates a forward-convex labeling for the graph. Forward-convex graphs are interesting to us because the page connectivity graphs of many frequently encountered join operations are forward-convex. A number of NP-complete problems can be solved in polynomial time when the input instances are restricted to forward-convex graphs. We have reported one such result here. Other results of this type can be found in [4, 7, 8, 9].

References

[1] K. S. Booth and G. S. Lueker, "Testing for the Consecutive Ones Property, Interval Graphs, and Graph Planarity Using PQ-Tree Algorithms," *J. of Computer and System Sciences* 13 (1976), 335–379.

[2] F. Fotouhi and S. Pramanik, "Optimal Secondary Storage Access Sequence for Performing Relational Join," *IEEE Trans. on Knowledge and Data Engineering* 1, 3 (Sept. 1989), 318–328.

[3] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, 1980.

[4] T. H. Lai and S. S. Wei, "Algorithms for Page Retrieval and Hamiltonian Paths on Forward-Convex Line Graphs," *Journal of Algorithms*, in press.

[5] E. R. Omiecinski, "Heuristics for Join Processing Using Nonclustered Indexes," *IEEE Trans. on Software Engineering* 15, 1 (Jan. 1989), 18–25.

[6] S. Pramanik and D. Ittner, "Use of Graph-Theoretic Models for Optimal Relational Database Accesses to Perform Join," *ACM Trans. on Database Systems* 10, 1 (March 1985), 57–74.

[7] J. Spinrad, Two Dimensional Partial Orders, Ph.D. Thesis, Princeton University, Dept. of EECS, 1982.

[8] J. Spinrad, A. Brandstadt, L. Stewart, "Bipartite Permutation Graphs," *Discrete Applied Mathematics* 18 (1987), 279–292.

[9] L. K. Stewart, Permutation Graph Structure and Algorithms, Ph.D. Thesis, University of Toronto, Dept. of Computer Science, 1985.

[10] A. Tucker, “A Structure Theorem for the Consecutive 1’s Property,” *Journal of Combinatorial Theory* 12(B) (1972), 153–162.

[11] S. S. Wei, *Design and Analysis of Algorithms for Processing Queries in Database Systems*, Ph.D. Thesis, Department of CIS, The Ohio State University, 1992.