1998 International Computer Symposium
Workshop on Algorithms
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

SOLVING THE ALL-PAIR-SHORTEST-LENGTH PROBLEM ON CHORDAL BIPARTITE GRAPHS

Chin-Wen Ho and Jou-Ming Chang*,†*

*Institute of Computer Science and Information Engineering,
National Central University, Chung-Li, Taiwan, R.O.C.
Email: hocw@csie.ncu.edu.tw

†Department of Information Management,
National Taipei College of Business, Taipei, Taiwan, R.O.C.
Email: spade@mail.ntcb.edu.tw

Abstract

The all-pairs-shortest-length (APSL) problem of a graph is to find the lengths of the shortest paths between all pairs of vertices. In this paper, we study the APSL problem on chordal bipartite graphs. By a simple reduction, we show that solving the APSL problem on chordal bipartite graphs can be transformed to solving the same problem on certain strongly chordal graphs. Consequently, there is an $O(n^2)$ time-optimal algorithm for this problem.

Keywords: shortest paths, chordal bipartite graphs, strongly chordal graphs.

1. Introduction

All graphs considered in this paper are undirected, loopless and without multiple edges. Let $G = (V, E)$ be a graph with vertex set V of size n and edge set E of size m. If G is associated with a weighted function $w : E \rightarrow \mathbf{R}$, the *length* of a path in G is the sum of the weights of its constituent edges. The *all-pairs-shortest-path* (APSP) problem is to find the shortest paths between all pairs of vertices. However, the *all-pairs-shortest-length* (APSL) problem is to find the "lengths" of the shortest paths between all pairs of vertices instead of providing explicit information about the shortest paths. One well-known algorithm for the APSP problem on general graphs without negative-length cycles is designed by Floyd [8] and it requires $O(n^3)$ time. As to the algorithms of solving the APSL problem, the most popular method is offered by Johnson [11] and it can be run in $O(nm + n^2 \log n)$ time.

Recently, Seidel [16] showed that the APSL problem on unweighted graphs (i.e., all the edges on the graph have the same weight) can be solved in $O(M(n) \log n)$ time with $O(n^2)$ space. $M(n)$ denotes the time complexity of multiplying two $n \times n$ matrices for small integers, and the time required is currently known to be $O(n^{2.376})$ [5]. The algorithm additionally builds a data structure that allows the shortest paths to be constructed in time proportional to their lengths. Besides, efficient algorithms have been developed for solving the APSL problem on some restricted classes of unweighted graphs. The $O(n^2)$ time-optimal algorithms are proposed for interval graphs [13, 15], bipartite permutation graphs [4], and strongly chordal graphs with a given strong elimination ordering [1, 6]. The optimal parallel algorithms of APSL problem for some certain classes of graphs can

be found in [4, 6]. All the above algorithms return an $n \times n$ distance matrix to record the lengths of all-pairs shortest paths.

In this paper, we consider the graphs for unweighted case and show that the APSL problem on chordal bipartite graphs can be solved in $O(n^2)$ time. The algorithm presented later uses a technique of transformation. We show that the algorithm of solving the APSL problem on strongly chordal graphs can be used for the same problem on chordal bipartite graphs. Moreover, a slight modification of a table provided in [1] can be used to construct the shortest paths in optimal time.

2. Background

For a graph $G = (V, E)$, the *open neighborhood* $N_G(u)$ of a vertex $u \in V$ is the set $\{v \in V : (u, v) \in E\}$; and the *closed neighborhood* $N_G[u]$ is $N_G(u) \cup \{u\}$. The *distance* $d_G(u, v)$ of two vertices $u, v \in V$ is the number of edges of the shortest paths between u and v in G. A *clique* in a graph is a subset of vertices which induce a complete subgraph. A *maximal clique* is a clique that is not properly contained in any other clique. A vertex $u \in V$ is called *simplicial* if $N_G[u]$ induces a clique of G. For a given cycle C of a graph, an edge (u, v) joining two vertices u and v that are nonadjacent along the cycle is called a *chord*. Moreover, a chord (u, v) is called an *odd chord* if the distance $d_C(u, v)$ is odd. A graph is *chordal* if every cycle of length at least four has a chord. A chordal graph is *strongly chordal* if every even cycle of length six or more has an odd chord.

In algorithmic aspects, many efficient algorithms on some special graphs are designed by the linear structure of vertex ordering. Let π be an ordering of vertices in a graph. We write $v_i <_\pi v_j$ if v_i comes before v_j in π. Note that, if π is clear from the context, we will simply write $v_i < v_j$. For a graph $G = (V, E)$, a *perfect elimination ordering* is an ordering $v_1 < v_2 < \cdots < v_n$ of V with the property that for each $i = 1, \ldots, n$, v_i is a simplicial vertex of G_i, where G_i is the subgraph of G induced by the vertex set $\{v_i, v_{i+1}, \ldots, v_n\}$. Fulkerson and Gross [9] showed that a graph G is chordal if and only if it has a perfect elimination ordering. A perfect elimination ordering is a *strong elimination ordering* (SEO for short) if for any $i < j < k$ and $v_j, v_k \in N_{G_i}(v_i)$, $N_{G_i}[v_j] \subseteq N_{G_i}[v_k]$. Farber [7] showed that a graph is strongly chordal if and only if it admits an SEO.

In this paper, we use $G = (X, Y, E)$ to denote a bipartite graph which consists of two distinct sets of vertices $X = \{x_1, x_2, \ldots, x_p\}$ and $Y = \{y_1, y_2, \ldots, y_q\}$, where $p + q = n$. For a bipartite graph G, the *bipartite adjacency matrix* of G is a $p \times q$ (0,1)-matrix $M_G = (a_{i,j})$, while $a_{i,j} = 1$ if and only if $(x_i, y_j) \in E$. A bipartite graph is called *chordal bipartite* if every cycle of length at least six has a chord. Note that, chordal bipartite graphs form a large class of bipartite graphs containing, for example, convex and biconvex bipartite graphs, bipartite permutation graphs, and bipartite distance hereditary graphs (or (6,2)-chordal bipartite graphs). For an overview of these classes of graphs, please refer to [2].

A (0,1)-matrix is Γ-*free* if it contains no two pairs of rows and columns that induce the submatrix

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}.$$

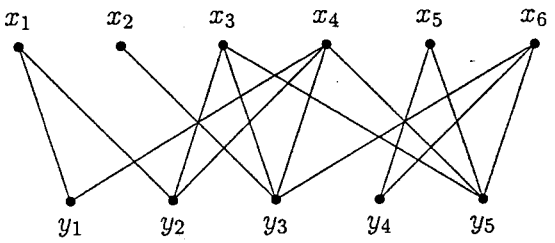
A matrix that has a permutation of the rows and columns that are Γ-free is called a *totally-balanced matrix*. The notion about totally-balanced matrix turns out to be extremely important in the study of strongly chordal graphs and chordal bipartite graphs. Hoffman et al. [10] used Γ-free matrices to characterize when a particular linear programming problem could be solved using a greedy algorithm. They also

showed that a graph G is chordal bipartite if and only if the bipartite adjacency matrix M_G is totally-balanced. Farber [7] showed that a graph G is strongly chordal if and only if the neighborhood matrix of G and the clique-vertex incidence matrix of G are totally-balanced.

A *doubly lexical ordering* of a matrix is an ordering of the rows and of the columns in which both the rows and the columns, as vectors, are lexically nondecreasing. Lubiw [12] proved that any doubly lexical ordering of a totally balanced matrix is Γ-free, which led to efficient recognition algorithms for chordal bipartite graphs and strongly chordal graphs. Let M be a $p \times q$ nonnegative matrix containing r nonzero entries. Pagie and Tarjan [14] gave an $O(p+q+r\log(p+q))$ time algorithm for finding a doubly lexical ordering of M. Besides, Spinrad [17] gave an $O(pq)$ time algorithm for finding this ordering if M is a (0,1)-matrix. Hence, the time complexities of the best known recognition algorithms for these cases of graphs are $O(m\log n)$ for sparse graphs and $O(n^2)$ for dense graphs, respectively.

For a bipartite graph $G=(X,Y,E)$, an ordering of vertices $\{x_1 < x_2 < \cdots < x_p\} \cup \{y_1 < y_2 < \cdots < y_q\}$ is a *strong Y-elimination ordering* if for every $1 \le i \le q$, $1 \le j < k \le p$ and $x_j, x_k \in N_G(y_i)$, $N_{G_{y_i}}(x_j) \subseteq N_{G_{y_i}}(x_k)$ where G_{y_i} is the subgraph of G induced by the vertices of $X \cup \{y_h \in Y : h \ge i\}$. Chang [3] showed that a bipartite graph is chordal bipartite if and only if it admits a strong Y-elimination ordering. Note that, for a chordal bipartite graph G, if the bipartite adjacency matrix M_G is Γ-free, then the ordering of the rows together with the ordering of the columns of M_G corresponds to a strong Y-elimination ordering of G. Consequently, a strong Y-elimination ordering of a chordal bipartite graph can be obtained from a doubly lexical ordering of its bipartite adjacency matrix. In the remainder of this paper, we assume that a strong Y-elimination ordering of a chordal bipartite graph is available.

Example. Figure 1 (a) gives a chordal bipartite graph G that its vertices have been numbered according to a strong Y-elimination ordering. In this Figure, $N_G(y_3) = \{x_2, x_3, x_4, x_6\}$, $N_{G_{y_3}}(x_2) = \{y_3\}$, $N_{G_{y_3}}(x_3) = N_{G_{y_3}}(x_4) = \{y_3, y_5\}$ and $N_{G_{y_3}}(x_6) = \{y_3, y_4, y_5\}$. Thus, $N_{G_{y_3}}(x_2) \subseteq N_{G_{y_3}}(x_3) \subseteq N_{G_{y_3}}(x_4) \subseteq N_{G_{y_3}}(x_6)$ satisfies the property of the strong Y-elimination ordering. Also, it is easy to see that the corresponding bipartite adjacency matrix of G shown in Figure 1 (b) is Γ-free.



(a)

	y_1	y_2	y_3	y_4	y_5
x_1	1	1	0	0	0
x_2	0	0	1	0	0
x_3	0	1	1	0	1
x_4	1	1	1	0	1
x_5	0	0	0	1	1
x_6	0	0	1	1	1

(b)

Figure 1: (a) A chordal bipartite graph. (b) A bipartite adjacency matrix.

3. The APSL problem on chordal bipartite graphs

In this section, we assume that all the considered graphs are connected. For a bipartite graph $G=(X,Y,E)$, we define $G_Y = (X \cup Y, E \cup E_Y)$ as an augmenting graph of G with respect to Y,

where

$$E_Y = \{(y, y') : y, y' \in Y \text{ and } N_G(y) \cap N_G(y') \neq \emptyset\}.$$

As a result, the distances between vertices of G_Y can be formulated by the following lemma.

Lemma 3.1 *For any bipartite graph $G = (X, Y, E)$, the following statements are true.*

(1) $d_{G_Y}(y, y') = \frac{1}{2} d_G(y, y')$ *for* $y, y' \in Y$;

(2) $d_{G_Y}(x, x') = \frac{1}{2} d_G(x, x') + 1$ *for* $x, x' \in X$;

(3) $d_{G_Y}(x, y) = \frac{1}{2}(d_G(x, y) + 1)$ *for* $x \in X$ *and* $y \in Y$.

Proof. Since the verifications of statements (2) and (3) are similar to statement (1), only statement (1) is proved here. Let $P = (y = z_0, z_1, \ldots, z_k = y')$ be any shortest path from y to y' in G. Since G is bipartite, k must be even, and for each vertex $z_i \in X$ we have $(z_{i-1}, z_{i+1}) \in E_Y$. Thus, $(y = z_0, z_2, z_4, \ldots, z_{k-2}, z_k = y')$ forms a path of G_Y and $d_{G_Y}(y, y') \leq \frac{k}{2} = \frac{1}{2} d_G(y, y')$. Furthermore, if there is a path joining y and y' with length less than $\frac{k}{2}$ in G_Y, then P cannot be a y-y' shortest path in G. Consequently, $d_{G_Y}(y, y') = \frac{1}{2} d_G(y, y')$. □

Let G be a chordal bipartite graph. The *last neighbor* of a vertex $y \in Y$, denoted by $LN(y)$, is the largest vertex in the strong Y-elimination ordering that is adjacent to y. The following lemma shows that if G is a chordal bipartite graph then there is a more efficient algorithm to construct G_Y without testing $N_G(y) \cap N_G(y')$ for every pair of vertices $y, y' \in Y$.

Lemma 3.2 *Let $G = (X, Y, E)$ be a chordal bipartite graph. Then, $E_Y = \{(y_i, y_j) : y_i \in Y$ and $y_j \in N_{G_{y_i}}(LN(y_i))$ for $i \neq j\}$.*

Proof. Assume that $y_i \in Y$ and $y_j \in N_{G_{y_i}}(LN(y_i))$ with $i \neq j$. By the definition of G_{y_i}, we obtain $y_i < y_j$ in the strong Y-elimination ordering of G. Since $LN(y_i) \in N_{G_{y_i}}(y_i) \cap N_{G_{y_i}}(y_j)$, $(y_i, y_j) \in E_Y$ directly follows from the definition that $N_G(y_i) \cap N_G(y_j) \neq \emptyset$. Conversely, we consider that $(y_i, y_j) \in E_Y$ for any two vertices $y_i, y_j \in Y$. Without loss of generality, we assume $y_i < y_j$ in the strong Y-elimination ordering. Since $(y_i, y_j) \in E_Y$, there is a vertex $x \in X$ such that $x \in N_G(y_i) \cap N_G(y_j)$. Since $LN(y_i)$ is the largest vertex of X that is adjacent to y_i, $N_{G_{y_i}}(x) \subseteq N_{G_{y_i}}(LN(y_i))$ follows from the property of strong Y-elimination ordering. Thus, $y_j \in N_{G_{y_i}}(x)$ implies $y_j \in N_{G_{y_i}}(LN(y_i))$. □

Lemma 3.3 *Let $G = (X, Y, E)$ be a chordal bipartite graph and $K \subseteq Y$. Then, K is a clique of G_Y if and only if there is a vertex $x \in X$ in G such that it is adjacent to all the vertices of K.*

Proof. The "if" part is obvious from the definition of augmenting graph G_Y. Conversely, the proof is produced by induction on k, the number of vertices of K. It is trivial for $k \in \{1, 2\}$. Assume that the lemma is true for every clique with no more than k vertices. Let $K = \{y_{h_1}, y_{h_2}, \ldots, y_{h_k}\}$ with $y_{h_1} < y_{h_2} < \cdots < y_{h_k}$ in the strong Y-elimination ordering. By the hypothesis, there are two vertices $x_i, x_j \in X$ in G_Y such that x_i is adjacent to all the vertices of $y_{h_1}, y_{h_2}, \ldots, y_{h_{k-1}}$, and x_j is adjacent to both y_{h_1} and y_{h_k}. Due to the property of strong Y-elimination ordering, if $x_i < x_j$ then x_j is adjacent to all the vertices of $y_{h_2}, \ldots, y_{h_{k-1}}$. On the other hand, if $x_j < x_i$ then x_i is adjacent to y_{h_k}. Thus, all the vertices of K have at least a common neighbor in X. □

From Lemma 3.3, it is easy to see that if G is chordal bipartite, then the set of all maximal cliques of G_Y is $\{N_{G_Y}[x] : x \in X\}$. The following procedure describes an algorithm to compute an ordering of vertices of G_Y. Theo-

rem 3.5 shows that the resulting ordering is an SEO. Hence, G_Y is a strongly chordal graph.

Procedure Compute-SEO
Input: a chordal bipartite graph $G=(X,Y,E)$ and a strong Y-elimination ordering of G.
Output: an SEO of G_Y.
begin
  for $i = 1$ to $|X|$ **do**
    let $\{y_{h_1}, y_{h_2}, \ldots, y_{h_k}\}$ be the set consisting of all neighbors of x_i in G and assume that $y_{h_1} < y_{h_2} < \cdots < y_{h_k}$ in the strong Y-elimination ordering;
    put x_i in the SEO and remove x_i from G;
    for $j = 1$ to k **do**
      if y_{h_j} is an isolated vertex in the current graph
      then
        put y_{h_j} in the SEO and remove y_{h_j} from G;
    end for
  end for
end.

For convenience, throughout the rest of this paper we refer σ to the given strong Y-elimination ordering of G, and π to the resulting ordering obtained from the above procedure. It is obvious that for every two vertices $x_i, x_j \in X$, $x_i <_\sigma x_j$ if and only if $x_i <_\pi x_j$. The following lemma shows that with a restricted condition, the property is still satisfied for the vertices of Y.

Lemma 3.4 *Let y_i and y_j be any two vertices of Y with at least a common neighbor in X. Then, $y_i <_\sigma y_j$ if and only if $y_i <_\pi y_j$.*

Proof. Assume that $y_i <_\sigma y_j$ and let $x \in X$ be the largest vertex in σ that is adjacent to both y_i and y_j. We claim that $LN(y_i) = x$. Suppose not, i.e., $x <_\sigma LN(y_i)$. Based on the property of strong Y-elimination ordering, we have $N_{G_{y_i}}(x) \subseteq N_{G_{y_i}}(LN(y_i))$. Since $y_j \in N_{G_{y_i}}(x)$, y_j must be adjacent to $LN(y_i)$. This contradicts that x is the largest vertex adjacent to both y_i and y_j. By the assumption $y_i <_\sigma y_j$ and the result that $LN(y_i) = x$ is adjacent to y_j, we have $y_i <_\pi y_j$. Conversely, suppose that y_i does not precede y_j in σ. This means that $y_j <_\sigma y_i$ since the ordering of vertices of Y in σ is a total ordering. Thus, using the same argument mentioned above we can show that y_j comes before y_i in π. □

Theorem 3.5 *Given a chordal bipartite graph $G = (X,Y,E)$, the procedure Compute-SEO generates an SEO of G_Y in $O(n+m)$ time.*

Proof. The procedure Compute-SEO can obviously be implemented in $O(n+m)$ time. Let $z_1 <_\pi z_2 <_\pi \cdots <_\pi z_n$ be the ordering of vertices of G_Y that is generated from the above procedure. Let G_i be the subgraph of G_Y induced by the vertex set $\{z_i, z_{i+1}, \ldots, z_n\}$. In the following, we will show that each vertex z_i is a simplicial vertex of G_i. Moreover, we show that if $z_j, z_k \in N_{G_i}(z_i)$ for $z_i <_\pi z_j <_\pi z_k$, then $N_{G_i}[z_j] \subseteq N_{G_i}[z_k]$. The cases will be discussed respectively as follows.

Case 1: z_i is a vertex of X. We observe that all the vertices adjacent to z_i are the vertices of Y and $N_{G_i}[z_i]$ forms a clique of G_i. Thus, z_i is a simplicial vertex of G_i. Assume that $z_j, z_k \in N_{G_i}(z_i)$ with $z_j <_\pi z_k$. Note that, by Lemma 3.4 $z_j <_\sigma z_k$. To show that $N_{G_i}[z_j] \subseteq N_{G_i}[z_k]$, we consider that there is a vertex $z_l \in N_{G_i}(z_j) \backslash \{z_i, z_k\}$ for otherwise $N_{G_i}[z_j] = N_{G_i}[z_k] = \{z_i, z_j, z_k\}$. Clearly, $z_i <_\pi z_l$. Hence, if $z_l \in X$, then $z_i <_\sigma z_l$. In this case, it is easy to see that $(z_l, z_k) \in E$ because $N_{G_{z_j}}(z_i) \subseteq N_{G_{z_j}}(z_l)$ by the strong Y-elimination ordering of G. As a result, z_l and z_k are adjacent in G_i. On the other hand, if $z_l \in Y$, by definition there is a vertex $z_h \in X$ such that it is adjacent to both z_j and z_l in G.

In the following, we will show that z_l and z_k are adjacent in G_i. Clearly, if z_i and z_h are the same vertex, no further proof is necessary. If $z_h \neq z_i$, we proceed to the following subcases.

Case 1.1: $z_i <_\sigma z_h$.

By the strong Y-elimination ordering of G, $(z_i, z_j), (z_i, z_k), (z_h, z_j) \in E$ implies $(z_h, z_k) \in E$. Thus, $(z_l, z_k) \in E_Y$ follows from the fact that z_h is adjacent to both z_l and z_k in G.

Case 1.2: $z_h <_\sigma z_i$ and $z_j <_\sigma z_l$.

By the strong Y-elimination ordering of G, $(z_h, z_j), (z_h, z_l), (z_i, z_j) \in E$ implies $(z_i, z_l) \in E$. Therefore, $(z_l, z_k) \in E_Y$ follows from the fact that z_i is adjacent to both z_l and z_k in G.

Case 1.3: $z_h <_\sigma z_i$ and $z_l <_\sigma z_j$.

It is clear that there is at least a vertex $z_p \in X$ with $z_i <_\sigma z_p$ such that z_p and z_l are adjacent in G_i (otherwise, $z_l <_\pi z_i$). By the strong Y-elimination ordering of G, $(z_h, z_l), (z_h, z_j), (z_p, z_l) \in E$ implies $(z_p, z_j) \in E$, and $(z_i, z_j), (z_i, z_k), (z_p, z_j) \in E$ implies $(z_p, z_k) \in E$. Therefore, $(z_l, z_k) \in E_Y$ follows from the fact that z_p is adjacent to both z_l and z_k in G.

Case 2: z_i is a vertex of Y. From the procedure Compute-SEO, we can see that all the vertices of X adjacent to z_i have been removed when z_i is added into the ordering π. Thus $N_{G_i}[z_i] \subseteq Y$. To show that z_i is a simplicial vertex of G_i, we first prove that for any two vertices $z_j, z_k \in N_{G_i}(z_i)$, there is at least a vertex of X such that it is adjacent to both z_j and z_k in G; i.e., $(z_j, z_k) \in E_Y$. Without loss of generality, we assume $z_j <_\sigma z_k$. By definition, if $z_j, z_k \in N_{G_i}(z_i)$, then there are two vertices $z_{j'}, z_{k'} \in X$ such that $z_{j'}$ (resp. $z_{k'}$) is adjacent to both z_i and z_j (resp. z_i and z_k) in G. Clearly, if $z_{j'}$ and $z_{k'}$ are the same vertex, no further proof is necessary. Due to the strong Y-elimination ordering of G, if $z_{j'} <_\sigma z_{k'}$, then $(z_{k'}, z_j) \in E$ because $N_{G_{z_i}}(z_{j'}) \subseteq N_{G_{z_i}}(z_{k'})$. On the other hand, if $z_{k'} <_\sigma z_{j'}$, a similar argument can show that $(z_{j'}, z_k) \in E$. Thus, $(z_j, z_k) \in E_Y$. Based on the result that every two vertices $z_j, z_k \in N_{G_i}(z_i)$ are adjacent in G_i, $N_{G_i}[z_i]$ forms a clique of G_i. Besides, since every two vertices $z_j, z_k \in N_{G_i}(z_i)$ have a common neighbor in X, if $z_j <_\sigma z_k$ then $z_j <_\pi z_k$ by Lemma 3.4. We now show that $N_{G_i}[z_j] \subseteq N_{G_i}[z_k]$ in this case. We may assume that there is a vertex $z_l \in N_{G_i}(z_j) \backslash \{z_i, z_k\}$ for otherwise $N_{G_i}[z_j] = N_{G_i}[z_k] = \{z_i, z_j, z_k\}$. Clearly, $z_i <_\pi z_l$. Since $z_i, z_j, z_k \in Y$ and $\{z_i, z_j, z_k\}$ forms a clique in G_Y, by Lemma 3.3 there is a vertex $z_{i'} \in X$ such that it is adjacent to all the vertices of z_i, z_j and z_k. Since the two vertices $z_{i'} \in X$ and $z_i \in Y$ are adjacent, $z_{i'} <_\pi z_i$. As a result, we have $z_{i'} <_\pi z_l$. Now, if $z_l \in X$ then $z_{i'} <_\sigma z_l$. Thus, $(z_{i'}, z_j), (z_{i'}, z_k), (z_l, z_j) \in E$ implies $(z_l, z_k) \in E$ by the strong Y-elimination ordering of G. Next, we consider the case $z_l \in Y$. Since $(z_l, z_j) \in E_Y$, there must exist a vertex (say) $z_h \in X$ that is adjacent to both z_l and z_j in G. Clearly, if $z_{i'}$ and z_h are the same vertex, then $(z_l, z_k) \in E_Y$. On the other hand, we may consider the following three subcases: (i) $z_{i'} <_\sigma z_h$; (ii) $z_h <_\sigma z_{i'}$ and $z_j <_\sigma z_l$; and (iii) $z_h <_\sigma z_{i'}$ and $z_l <_\sigma z_j$. Since the arguments are similar to those proofs of Cases 1.1, 1.2 and 1.3, we can obtain that $(z_l, z_k) \in E_Y$ in the above three subcases without doubt. □

To solve the APSL problem on a chordal bipartite graph G, we consider that G is given by a strong Y-elimination ordering. Note that, this ordering can be achieved in $O(n^2)$ time using the algorithm of [17]. We then construct the augmenting graph G_Y. By Lemma 3.2, this work can easily be done in $O(n^2)$ time because that E_Y contains at most $O(n^2)$ edges. Theorem 3.5 shows that G_Y is strongly chordal and an SEO can be computed in $O(n + m)$ time. Hence, we can use the all-pairs-shortest-length algorithm proposed by Balachandhran and Rangan [1] to obtain the distance matrix of G_Y. In addition, Lemma 3.1 shows that the distance

between each pair of vertices in G can immediately be determined from the distance matrix of G_Y. Since there are n^2 entries in the matrix, we conclude the following:

Theorem 3.6 *The APSL problem on chordal bipartite graphs can be solved in $O(n^2)$ time.*

The algorithm provided in [1] for solving the APSL problem on a strongly chordal graph G additionally establishes an $n \times n$ table S which can be used to construct the shortest paths in optimal time. In that table, the entry $S[u, v]$ reports the vertex next to u in a shortest path from u to v in G; i.e., $S[u, v] = w$ if and only if $d_G(u, v) = d_G(u, w) + d_G(w, v) = 1 + d_G(w, v)$. Therefore, to find the u-v shortest path, we begin by retrieving the entry $S[u, v]$, and then retrieve $S[w, v]$ in the next step. The previous procedure is carried out repeatedly until we find an entry $S[w', v] = v$ in the table. Consequently, the shortest path between u and v in G can be obtained. The detail about how to build the table S is not mentioned here. Please refer to [1] for reference.

By modifying the table S of the strongly chordal graph G_Y, we are able to show how to construct the shortest paths for a chordal bipartite graph G. By Lemma 3.2, $(y, y') \in E_Y$ for $y <_\sigma y'$ if and only if $LN(y) \in N_G(y) \cap N_G(y')$. Thus, for each entry $S[y, z] = y'$ where $y, y' \in Y$ and $z \in X \cup Y$, we reset $S[y, z]$ to be $LN(y)$ and $S[LN(y), z]$ to be y'. This modification totally takes $O(n^2)$ time. Due to the structural properties of G_Y, the internal vertices of any shortest path with length ≥ 2 in G_Y are the vertices of Y. Also, from Lemma 1 we have known that $d_G(y, y') = 2 \cdot d_{G_Y}(y, y')$ for $y, y' \in Y$. Therefore, the new table S can preserve the information for the shortest paths of G. Furthermore, constructing a shortest path between vertices in G can be performed by retrieving the new table as the previous process.

References

[1] V. Balachandhran and C. P. Rangan, All-pairs-shortest-length on strongly chordal graphs, *Discrete Appl. Math.*, **69** (1996) 169-182.

[2] A. Brandstädt, Special graph classes — a survey, Schriftenreihe des Fachbereichs Mathematik, SM-DU-199, Universität Duisburg, 1991.

[3] M. S. Chang, Algorithms for maximum matching and minimum fill-in on chordal bipartite graphs, Proceedings of the 7th Annual International Symposium on Algorithms and Computation (ISAAC'96), Lecture Notes in Computer Science, Vol. 1178, pp. 146-155, 1996.

[4] L. Chen, Solving the shortest-paths problem on bipartite permutation graphs efficiently, *Inform. Process. Lett.*, **55** (1995) 259-264.

[5] D. Coppersmith and S. Winograd, Matrix multiplication via arithmetic progressions, *J. Symbolic Comput.*, **9** (1990) 251-280.

[6] E. Dahlhaus, Optimal (parallel) algorithms for the all-to-all vertices distance problem for certain graph classes, Lecture Notes in Computer Science, Vol. 657, pp. 60-69, 1993.

[7] M. Farber, Characterizations of strongly chordal graphs, *Discrete Math.* **43** (1983) 173-189.

[8] R. W. Floyd, Algorithm 97 (SHORTEST PATH), *Communication ACM*, **5** (1962) 345.

[9] D. R. Fulkerson and O. A. Gross, Incidence matrices and interval graphs, *Pacific J. Math.*, **15** (1965) 835-855.

[10] A. J. Hoffman, A. W. J. Kolen and M. Sakarovitch, Totally-balanced and greedy matrices, *SIAM J. Alg. Disc. Meth.*, **6** (1985) 721-730.

[11] D. B. Johnson, Efficient algorithms for shortest paths in sparse networks, *J. ACM* **24** (1977) 1-13.

[12] A. Lubiw, Doubly lexical orderings of matrices, *SIAM J. Comput.*, **16** (1987) 854-879.

[13] P. Mirchandani, A simple $O(n^2)$ algorithm for the all-pairs shortest path problem on an interval graph, *Networks*, **27** (1996) 215-217.

[14] R. Paige and R. E. Tarjan, Three partition refinement algorithms, *SIAM J. Comput.*, **16** (1987) 973-989.

[15] R. Ravi, M. V. Marathe and C. P. Rangan, An Optimal algorithm to solve the all-pair shortest path problem on interval graphs, *Networks*, **22** (1992) 21-35.

[16] R. Seidel, On the all-pairs-shortest-path problem in unweighted undirected graphs, *J. Comput. System Sci.*, **51** (1995) 400-403.

[17] J. P. Spinrad, Doubly lexical orderings of dense 0-1 matrices, *Inform. Process. Lett.*, **45** (1993) 229-235.