Maximum Weighted Edge Biclique Problem on Bipartite Graphs

Arti Pandey[1(✉)], Gopika Sharma[1], and Nivedit Jain[2]

[1] Department of Mathematics, Indian Institute Of Technology Ropar, Rupnagar 140001, Punjab, India
{arti,2017maz0007}@iitrpr.ac.in

[2] Department of Computer Science and Engineering, Indian Institute of Technology Jodhpur, Karwar 342037, Rajasthan, India
jain.22@iitj.ac.in

Abstract. For a graph G, a complete bipartite subgraph of G is called a biclique of G. For a weighted graph $G = (V, E, w)$, where each edge $e \in E$ has a weight $w(e) \in \mathbb{R}$, the Maximum Weighted Edge Biclique (MWEB) problem is to find a biclique H of G such that $\sum_{e \in E(H)} w(e)$ is maximum. The decision version of the MWEB problem is known to be NP-complete for bipartite graphs. In this paper, we show that the decision version of the MWEB problem remains NP-complete even if the input graph is a complete bipartite graph. On the positive side, if the weight of each edge is a positive real number in the input graph G, then we show that the MWEB problem is $O(n^2)$-time solvable for bipartite permutation graphs, and $O(m+n)$-time solvable for chain graphs, which is a subclass of bipartite permutation graphs.

Keywords: Maximum Weighted Edge Biclique · Bipartite permutation graphs · Chain graphs · NP-completeness · Graph algorithms

1 Introduction

Let $G = (V, E)$ be a graph. A *biclique* of G is a complete bipartite subgraph of G. The Maximum Vertex Biclique (MVB) problem is to find a biclique of G with maximum number of vertices. The decision version of the MVB problem is NP-complete for general graphs [1], but the MVB problem is polynomial time solvable for bipartite graphs [1]. The Maximum Edge Biclique (MEB) problem is to find a biclique in G with maximum number of edges. The decision version of the MEB problem is NP-complete for general graphs [1] and it also remains NP-complete for bipartite graphs [2]. Many researchers have also studied some other variations of these problems, see [1,3–5]. The Maximum Edge Biclique problem was first introduced in [1] and further studied in [2,6–9]. The MEB problem has applications in biclustering analysis techniques, where one is interested to capture the relationship between genes and conditions. The goal of biclustering algorithms is to find a subset of genes J and a subset of conditions C such that the change in the expression level of each $j \in J$ with respect to each $c \in C$ is significant. More details about the application of the MEB problem can be found in [3,8]. Since the MEB problem is also hard to approximate in bipartite graphs within n^{δ} for some $\delta > 0$ [10,11] under certain assumptions such as random 4-SAT or 3-SAT hardness hypothesis, researchers have also studied the problem for subclasses of bipartite graphs. The MEB problem is polynomial time solvable for the following subclasses of bipartite graphs: chordal bipartite graphs, convex bipartite graphs and bipartite permutation graphs [8,12–16]. Some other hardness results are also available for the MEB problem based on some assumptions [6,9,17,18]. In this paper, we study the weighted version of the MEB problem.

The weighted version of the MEB problem, namely Maximum Weighted Edge Biclique (MWEB) problem is also studied in literature, see [3–5,7]. Given a weighted graph $G = (V, E, w)$, where each edge $e \in E$ has a weight $w(e) \in \mathbb{R}$, the MWEB problem is to find a biclique C of G such that the sum of the weights of edges of C is maximum. Note that the Maximum Weighted Edge Biclique problem is the generalized version of the Maximum Edge Biclique problem. So, the hardness results for the Maximum Edge Biclique problem are also valid for the Maximum Weighted Edge Biclique problem. Given a graph G and an integer $k > 0$, the Weighted Edge Biclique Decision Problem (WEBDP) is to find a biclique C of G such that the sum of edge weights of C is at least k. In this paper, we show that WEBDP is NP-complete even for complete bipartite graphs.

There exists a restricted version of the MWEB problem, namely the S-MWEB problem, where S is a subset of real numbers from which edge weights are taken and the input graph is a bipartite graph. In 2008, Tan [7] proved that for a wide range of choices of S, no polynomial time algorithm can approximate the S-MWEB problem within a factor of n^{ϵ} for some $\epsilon > 0$ unless RP=NP. He also proved that the decision version of the S-MWEB problem is NP-complete even for $S = \{-1, 0, 1\}$. In this paper, we show that this problem remains NP-complete when $S = \{1, -M\}$ $(M > |E(G)|)$. On the positive side, we show that for a set S of positive real numbers, the S-MWEB problem is quadratic time solvable for bipartite permutation graphs and linear-time solvable for chain graphs.

The rest of the paper is organized as follows. In Sect. 2, we give some pertinent definitions and notations used in the paper. In Sect. 3, we show that the Weighted Edge Biclique Decision Problem is NP-complete even for complete bipartite graphs. In Sect. 4, we show that the S-MWEB is $O(n^2)$-time solvable for bipartite permutation graphs if S is as set of positive real numbers. In Sect. 5, we propose a linear-time algorithm to solve the S-MWEB problem in $O(m+n)$-time for chain graphs (under the assumption that S is a set of positive real numbers). Finally, Sect. 6 concludes the paper.

2 Preliminaries

We are considering undirected, simple and connected graphs throughout this paper. A graph G is called a *bipartite graph* if its vertex set can be partitioned into two sets, say V_1 and V_2 such that every edge of G has one end point in V_1 and other end point in V_2. The set $\{V_1, V_2\}$ is called a bipartition of G. We denote such a bipartite graph by $G = (V_1, V_2, E)$, where E is the edge set of G. A biclique of G is a complete bipartite subgraph of G. A biclique of a bipartite graph is called maximal if it is not a proper subgraph of any other biclique of G. Weight of a biclique is defined as the sum of weights of all edges belonging in it. For a complete bipartite graph with bipartition $\{X, Y\}$ such that $|X| = s$ and $|Y| = t$, we use the notation $K_{s,t}$. For a vertex v of a graph G, $d(v)$ denotes degree of v and $N(v)$ denotes the open neighborhood of v which is the set of vertices adjacent to v in G. For a set $S \subseteq V(G), N(S)$ denotes the union of open neighborhooods of all vertices in S. Throughout this paper, n denotes order (number of vertices) of the graph and m denotes the size (number of edges) of the graph under consideration.

A binary relation that is reflexive, symmetric and transitive on the same set, is called an *equivalence relation*. For an equivalence relation $\sim$ on a set S, the equivalence class of $x \in S$ is the set containing all elements which are related to x by $\sim$. We denote equivalence class of an element x by $[x]$. Equivalence classes of two elements are either disjoint or identical. Disjoint equivalence classes give a partition of the set on which the relation was defined.

3 NP-completeness

In this section, we show that the Weighted Edge Biclique Decision Problem (WEBDP) is NP-complete for complete bipartite graphs which is a very restricted subclass of bipartite graphs.

Theorem 1. *WEBDP is NP-complete for complete bipartite graphs.*

Proof. Clearly, WEBDP is in NP. To prove the NP-hardness of the WEBDP for complete bipartite graph, we make a polynomial reduction from the unweighted version of the same problem for bipartite graphs. So, we prove a construction of a weighted complete bipartite graph from an unweighted bipartite graph.

Let $G = (X, Y, E)$ be an unweighted bipartite graph with $|X| = n_1$ and $|Y| = n_2$. We construct a new graph H which is nothing but K_{n_1,n_2}. Now, for an edge e in H, we define its weight to be 1 if $e \in E$ and $-M$ otherwise, where $M > m = |E|$. So, H is a weighted complete bipartite graph with weights as any real number. Figure 1 illustrates the construction of H from G. The dashed edges in Fig. 1 are the edges with weight $-M$.

Now to complete the proof of the theorem, we only need to prove the following claim.

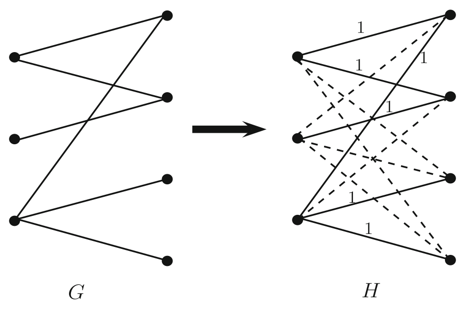

Fig. 1. An illustration to the construction of H from G.

Claim. G has a biclique of size at least $k > 0$ if and only if H has a biclique of weight at least $k > 0$.

Proof. The proof is easy and hence is omitted. □

Hence, the theorem is proved. □

We have observed that the WEBDP is NP-complete even for complete bipartite graphs. In next sections, we will discuss S-MWEB problem with S as the set of positive real numbers, which will be the restricted version of the MWEB problem. Throughout Sects. 4 and 5, by MWEB problem we mean S-MWEB problem, where $S = \mathbb{R}^+$.

4 Bipartite Permutation Graphs

A graph $G = (V, E)$ is called a *permutation graph* if there exists a one-to-one correspondence between its vertex set and a set of line segments between two parallel lines such that two vertices of G are adjacent if and only if their corresponding line segments intersect. A graph $G = (V, E)$ is called a *bipartite permutation graph* if it is both bipartite and permutation graph. We describe two characterizations of bipartite permutation graphs. A *strong ordering* $(<_X, <_Y)$ of a bipartite graph $G = (X, Y, E)$ consists of an ordering $<_X$ of X and an ordering $<_Y$ of Y, such that for all edges ab, $a'b'$, with $a, a' \in X$ and $b, b' \in Y$: if $a <_X a'$ and $b' <_Y b$, then ab' and $a'b$ are edges in G. An ordering $<_X$ of X has the *adjacency property* if, for every vertex in Y, its neighbors in X are consecutive in $<_X$. The ordering $<_X$ has the *enclosure property* if, for every pair of vertices y, y' of Y with $N(y) \subseteq N(y')$, the vertices of $N(y') \setminus N(y)$ appear consecutively in $<_X$. Strong ordering, adjacency property and enclosure property described above give rise to the following results which are already proven facts [19] providing two characterizations of bipartite permutation graphs.

Theorem 2. *[19] The following statements are equivalent for a graph $G = (X, Y, E)$.*

1. *$G = (X, Y, E)$ is a bipartite permutation graph.*
2. *G has a strong ordering.*
3. *There exists an ordering of X which has the adjacency property and the enclosure property.*

For a connected graph, statements 2 and 3 of Theorem 2 can be combined as Lemma 1 which follows from the proof of Theorem 1 in [19].

Lemma 1. *[20] Let $(<_X, <_Y)$ be a strong ordering of a connected bipartite permutation graph $G = (X, Y, E)$. Then both $<_X$ and $<_Y$ have the adjacency property and the enclosure property.*

Throughout this section, $G = (X, Y, E)$ denotes a weighted bipartite permutation graph such that $|X| = k$ and $|Y| = k'$. Weights on the edges are some positive real numbers. We assume that the strong ordering $(<_X, <_Y)$ of vertices of G are already given for the input graph. This ordering is considered as $\{x_1, x_2, \ldots, x_k\}$ and $\{y_1, y_2, \ldots, y_{k'}\}$ for X and Y respectively. We write $u <_X v$ or $u <_Y v$ for vertices u, v of G if u appears before v in the strong ordering of vertices of G. We write $u < v$ when it is clear from the context that u, v are coming from which side of the bipartition. For any edge $x_i y_j$, its weight is denoted by w_{ij}.

Now, we define first and last neighbor of a vertex in G. Since both $<_X$ and $<_Y$ satisfy adjacency property, for a vertex v of G, its neighbor set has some consecutive vertices in $<_X$ or $<_Y$. *First neighbor* of v is defined as the vertex that appears first in the strong ordering of G in its neighbor set and *last neighbor* of v is defined as the vertex that appears last in the strong ordering of G in its neighbor set. For any vertex u of G, $f(u)$ denotes the first neighbor of u and $l(u)$ denotes the last neighbor of u. For u in X, we denote $f(u)$ by y_{α_u} and $l(u)$ by y_{β_u} where $1 \leq \alpha_u \leq \beta_u \leq k'$. Combining above results, it can be observed that for a bipartite permutation graph G with its strong ordering $(<_X, <_Y)$, it has the following properties which will be used in the further discussion (See [21]):

1. Given any vertex of G, its neighbor set consists of some consecutive vertices in $<_X$ or $<_Y$.
2. For a pair of vertices u, v from X or Y, if $u < v$ then $f(u) \leq f(v)$ and $l(u) \leq l(v)$.

Now, we will discuss about the structure of a maximal biclique of G which will be used in getting a maximum biclique of G.

4.1 Maximal Bicliques

Let $G' = (X', Y', E')$ denotes a maximal biclique of G with $X' = \{x_i, x_{i+1}, \ldots, x_j\}$ and $Y' = \{y_{i'}, y_{i'+1}, \ldots, y_{j'}\}$ then edge $x_i y_{i'}$ is called the *first edge* of G'. We call an edge uv of G as a *safe edge* if it is the first edge of some maximal biclique of G. We will see that one safe edge corresponds to exactly one maximal biclique of G and vice versa.

Lemma 2. *Let $G' = (X', Y', E')$ be a biclique of G with $X' = \{x_i, x_{i+1}, \ldots, x_j\}$ and $Y' = \{y_{i'}, y_{i'+1}, \ldots, y_{j'}\}$, then G' is a maximal biclique of G if and only if the following holds for the graph G.*

(a) $l(x_i) = y_{j'}$
(b) $f(x_j) = y_{i'}$
(c) $l(y_{i'}) = x_j$
(d) $f(y_{j'}) = x_i$

Proof. First, let us assume that G' is a maximal biclique. We need to show that conditions $(a), (b), (c)$ and (d) are true. For (a), it is clear that $l(x_i) \geq y_{j'}$ since G' is a biclique. If equality holds, we are done. So, let $l(x_i) > y_{j'}$, say $l(x_i) = y_t (> y_{j'})$. Since vertices are ordered according to the strong ordering, all vertices of X' are adjacent to the vertices $y_{j'+1}, y_{j'+2}, \ldots, y_t$ in G implying that G' is not a maximal biclique of G. Now for (b), suppose that $f(x_j) < y_{i'}$, say $f(x_j) = y_p (< y_{i'})$. All vertices of X' are adjacent to $y_p, y_{p+1}, \ldots, y_{i'-1}$ in G because of the strong ordering of the vertices of G, but G' was maximal. Similarly (c) and (d) can be proven.

Conversely, we assume that the conditions $(a), (b), (c)$ and (d) are true. Let, if possible, G' is not maximal. Then there exists a vertex v in G for which one of the following conditions must be satisfied: (i) $v < x_i$ and $vy_{j'} \in E(G), (ii)$ $v < y_{i'}$ and $vx_j \in E(G), (iii)$ $v > x_j$ and $vy_{i'} \in E(G)$, and (iv) $v > y_{j'}$ and $vx_i \in E(G)$. But none of the edges $vy_{j'}, vx_j, vy_{i'}, vx_i$ can be present in G because of our assumption that $(a), (b), (c)$ and (d) are true. So, G' is a maximal biclique. □

For any edge $e = uv$, the biclique corresponding to e, is the subgraph induced by the vertices $\{u, ..., l(v), v, ..., l(u)\}$. From Lemma 2, it can be observed that any maximal biclique of G can be identified from its first edge (safe edge). Given any edge uv ($u \in X$ and $v \in Y$) of G, one can easily check whether that is a safe edge or not as follows: If first neighbor of last neighbor of v is equal to v and first neighbor of last neighbor of u is equal to u, then uv qualifies as a safe edge. We observe from Lemma 2 that this condition is both necessary and sufficient for a biclique(corresponding to an edge uv) to be a maximal biclique. Hence, we can say that number of safe edges in G is equal to the number of maximal bicliques of G. We denote the maximal biclique corresponding to the safe edge e by G_e. For every vertex u of G, we define an array called *prefix sum array(psa)* of u of size $d(u)$ as an array in which each value equals the sum of weights of edges up to that position starting from $f(u)$. The *psa* of x_i(or y_j) is denoted by $A_i[\ ]$(or $B_j[\ ]$). Figure 2 represents a bipartite permutation graph. Next, we illustrate all the terminologies defined in this section using Fig. 2.

In bipartite permutation graph shown in Fig. 2, x_2y_2 is a safe edge since $f(l(x_2)) = f(y_5) = x_2$ and $f(l(y_2)) = f(x_4) = y_2$ but x_4y_4 is not as $f(l(x_4)) = f(y_5) = x_2 \neq x_4$. Prefix sum array of the vertex x_6 is $A_6 = \{27, 37, 48, 99\}$, where $A_6[1] = 27$, $A_6[2] = 27 + 10 = 37$, $A_6[3] = 27 + 10 + 11 = 48$ and $A_6[4] = 27 + 10 + 11 + 51 = 99$.

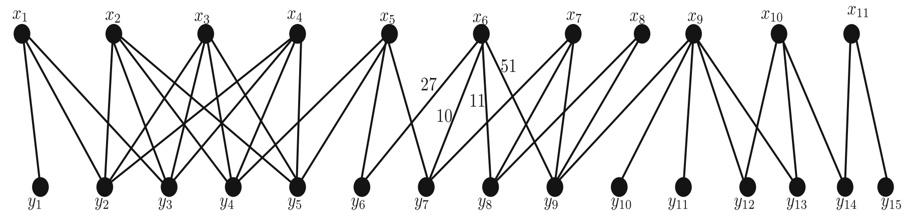

Fig. 2. An example of bipartite permutation graph.

4.2 Our Algorithm

Our idea for finding a maximum biclique is to look at all possible maximal bicliques of G and then return the one with the maximum weight. Since weights are positive real numbers any maximum biclique is some maximal biclique of G. The idea behind our algorithm is the following.

```
1. Find all safe edges of $G$.
2. Find psa of each vertex of $G$.
3. For each vertex $u \in X$,
       for each $v \in N(u)$ (choose vertex $v$ in the given ordering)
           if $e = uv$ is a safe edge
               find the maximal biclique $G_e$
               find $W_e$, the weight of biclique $G_e$ using psa of vertices.
4. Output the maximal biclique $G_{e^*}$ for which $W_{e^*}$ is maximum.
```

Note: we implement step 3 for each vertex in $O(n)$-time and hence overall complexity of step 3 is $O(n^2)$. The detailed algorithm is given in Algorithm 1.

Theorem 3. *Algorithm 1 outputs a maximum weighted edge biclique of the bipartite permutation graph G.*

Proof. The proof is omitted due to space constraints. □

Theorem 4. *Algorithm 1 runs in $O(n^2)$-time.*

Proof. For any edge e, it will take constant time to check whether an edge qualifies as a safe edge or not by Lemma 2. So, preprocessing all safe edges take $O(m)$-time as it scans all the edges one by one. For a vertex u of G, calculating its psa will take $d(u)$ amount of time. Hence, finding psa of each vertex will take $O(m)$-time. For a vertex $u \in X$, step 3 can be implemented in $O(n)$-time. This is possible because, for all the safe edges in which one of the end point is u, we can find the weights of the corresponding bicliques in $O(n)$-time altogether. So, overall step 3 takes $O(n^2)$-time. Therefore, the algorithm returns a maximum weighted edge biclique of G in $O(m) + O(m) + O(n^2) \approx O(n^2)$-time. □

Algorithm 1. Algorithm for finding a maximum weighted edge biclique of a bipartite permutation graph

```
Input: A bipartite permutation graph $G = (X, Y, E)$ with the strong ordering of its
       vertices
Output: A maximum weighted edge biclique of $G$
/* identifying safe edges */
for each edge e=uv in E do
    if f(l(v))==v and f(l(u))==u then
        mark e as a safe edge
/* finding prefix sum arrays of vertices of $G$ */
/* If $N(x_i) = \{y_{s_1}, y_{s_2}, \ldots, y_{s_{d(x_i)}}\}, S_{x_i}$ denotes the set $\{1, 2, \ldots, d(x_i)\}$ */
for each vertex $x_i$ from $X$ do
    for every $j$ from $S_{x_i}$ do
        $A_i[j] = A_i[j-1] + w_{is_j}$
/* similarly we can find psa of vertices of $Y$ */
max:=0
for each vertex $x = x_i$ from $X$ do
    sum:=0
        /* $S'_x$ denotes the set $\{y_{a_1}, y_{a_2}, \ldots, y_{a_t}\}$ where $a_1 < a_2 < \ldots < a_t$ such
        that $xy_{a_1}, xy_{a_2}, \ldots, xy_{a_t}$ are safe edges */
    for $j = 1$ to $t$ do
        /* finding maximal biclique corresponding to the safe edge $e = xy_{a_j}$
           */
        $X_e := \{x_i, x_{i+1}, \ldots, x_p\}$                       // $x_p = l(y_{a_j})$
        $Y_e := \{y_{a_j}, y_{a_j+1}, \ldots, y_q\}$                  // $y_q = l(x_i)$
        $E_e := \{uv | u \in X_e, v \in Y_e\}$, $G_e := (X_e, Y_e, E_e)$
        if sum==0 then
            for each vertex $x' = x_b$ from $X_e$ do
                $sum := sum + A_b[q - \alpha_{x'} + 1] - A_b[a_j - \alpha_{x'}]$
            $W_e := sum$
        else
            $W_e := sum + W_1 - W_2$, $sum := W_e$
                /* $W_1$ and $W_2$ are the weights of the subgraphs
                induced by the vertices $\{x_{c+1}, \ldots l(y_{a_j}), y_{a_j}, \ldots, y_q\}$ and
                $\{x_i, \ldots x_c, y_{a_{j-1}}, \ldots, y_{a_j - 1}\}$ respectively, where $x_c = l(y_{a_{j-1}})$.
                $W_1$ and $W_2$ are obtained using psa of vertices */
        if $W_e > max$ then
            $max := W_e$, $e^* := e$
return $G_{e^*}$ and $max$
```

5 Chain Graphs

A bipartite graph $G = (X, Y, E)$ is called a chain graph if there exists an ordering of vertices of $X = \{x_1, x_2, \ldots, x_{n_1}\}$ and an ordering of vertices of $Y = \{y_1, y_2, \ldots, y_{n_2}\}$ such that $N(x_1) \subseteq N(x_2) \subseteq \ldots \subseteq N(x_{n_1})$ and

$N(y_1) \supseteq N(y_2) \supseteq \ldots \supseteq N(y_{n_2})$. Throughout this section, $G = (X, Y, E)$ denotes a weighted chain graph with $|X| = n_1$ and $|Y| = n_2$. Weights on the edges are some positive real numbers. We assume that this ordering is given with the input graph.

For u, v in G, we define u and v to be *similar* vertices if $N(u) = N(v)$. For a set $S \subseteq V(G)$, we define S to be a *similar neighborhood set* if every two vertices from S are similar.

Now, we define a relation $\sim$ on X as for $u, v \in X$, $u \sim v$ if and only if vertices u and v are similar. One can easily observe that $\sim$ is an equivalence relation so it provides a partition P of the set X. If we define the same relation on the set Y, we will get a partition P' for the set Y. For any set $S \in P$, we keep the order of the vertices in S as it was given in the input chain graph. Order of the sets in P is also considered in such a way that taking union of all sets in that order gives the actual ordering of the vertices. We write $P = \{X_1, X_2, \ldots, X_{k_1}\}$ and $P' = \{Y_1, Y_2, \ldots, Y_{k_2}\}$, the partitions obtained for X and Y respectively from the relation $\sim$. Recall that $[x]$ denotes the equivalence class of the element x from X.

Lemma 3. *Let $\sim$ be the relation defined on X and Y as discussed above, then partitions P and P' are of same size, i.e. $|P| = |P'|$.*

Proof. We have defined the relation in such a way that vertices in one set of these partitions are similar to each other, so $N(X_1) \subset N(X_2) \subset \ldots \subset N(X_{k_1})$ and $N(Y_1) \supset N(Y_2) \supset \ldots \supset N(Y_{k_2})$ holds true. For any $i < j$, $N(X_i)$ is a proper subset of $N(X_j)$, so, say, $y \in N(X_j)$ such that $y \notin N(X_i)$. Since the graph is connected, this give rise to atleast two sets in P'. Hence, we get that $k_2 \geq k_1$. Similarly, $N(Y_i) \supset N(Y_j)$ gives $k_1 \geq k_2$ implying that $|P| = k_1 = k_2 = |P'|$. $\square$

Now, we define the representative vertex for each set of P. For a set $S \in P$, a vertex from S is called the *representative vetex* of the set S, if it is the least indexed vertex among all vertices of S. We denote representative vertex of a set S by r_S. Next we state some observations related to maximal bicliques of a chain graph which leads to a maximum weighted edge biclique of G.

5.1 Maximal Bicliques

Lemma 4. *Let $G' = (X', Y', E')$ be a maximal biclique of G, then the following holds:*

(a) If $x \in X'$, then $[x] \subseteq X'$.
(b) If, $y \in Y'$, then $[y] \subseteq Y'$.

Proof. (a) Here, we will show that $[x] \subseteq X'$ for any $x \in X'$. Let $x_0 \in [x]$, as x_0 and x are similar vertices, $N(x_0) = N(x)$. Now, $Y' \subseteq N(x) = N(x_0)$ implies that x_0 is adjacent to all vertices of Y' in G. We must have these edges in G' as it is a maximal biclique. So, $[x] \subseteq X'$ is true.

Proof of the part (b) is similar. $\square$

Below, we give a result which describes the detailed structure of a maximal biclique of a chain graph.

Lemma 5. *Let $G' = (X', Y', E')$ be a maximal biclique of G. Then there exists an index $1 \leq i \leq k$ such that $X' = X_i \cup X_{i+1} \cup \ldots \cup X_k$ and $Y' = N(r_{X_i})$.*

Proof. We know that vertices of G have an ordering as $\{x_1, x_2, \ldots, x_{n_1}\}$ and $\{y_1, y_2, \ldots, y_{n_2}\}$ for X and Y respectively. Let j be the minimum index from $\{1, 2, \ldots, n_1\}$ such that $x_j \in X'$ and there is some t such that $x_j \in X_t$. Now Lemma 4 tells that $[x_j] = X_t \subseteq X'$ implying that $x_j = r_{X_t}$. Since j is the smallest index, we get that $\{X_1 \cup X_2 \cup \ldots \cup X_{t-1}\} \cap X' = \phi$. Now, as $Y' \subseteq N(x_j)$ and G is a chain graph, $X' = X_t \cup X_{t+1} \cup \ldots \cup X_k$. Hence, for $i = t$, one part of the lemma holds. For the remaining part, it is enough to show that $N(x_j) \subseteq Y'$. So, let y be a neighbor of x_j, then y is adjacent to all vertices in the set $\{x_{j+1}, x_{j+2}, \ldots, x_{n_1}\}$ implying that $y \in Y'$. Hence, $Y' = N(r_{X_i})$ and $X' = X_i \cup X_{i+1} \cup \ldots \cup X_k$. $\square$

It can be identified from Lemma 5 that a chain graph has exactly k maximal bicliques, where k is the number of distinct equivalence classes corresponding to the relation $\sim$. Now, we define an array called *partition sum array (ptsa)* of size k for each $y \in Y$. In a partition sum array of a vertex y, each value contains the sum of weights of the edges incident on the vertex y coming from one set of P. We denote the *ptsa* of y_i by $A_i[\ ]$. Figure 3 represents a chain graph. We illustrate all the terminologies defined in this section using Fig. 3.

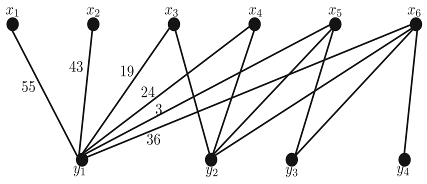

Fig. 3. An example of chain graph.

In the chain graph shown in Fig. 3, the partition $P = \{X_1, X_2, X_3, X_4\}$, where $X_1 = \{x_1, x_2\}, X_2 = \{x_3, x_4\}, X_3 = \{x_5\}$ and $X_4 = \{x_6\}$. Vertices x_3, x_4 are similar but x_1, x_3 are not and all the sets in P are similar neighborhood sets. Partition sum array of the vertex y_1 is $A_1 = \{98, 43, 3, 36\}$, where $A_1[1] = 55 + 43 = 98$, $A_1[2] = 19 + 24 = 43$, $A_1[3] = 3$ and $A_1[4] = 36$.

From, Lemma 5, we know the structure of maximal bicliques of G. One can easily see that each maximal biclique can be identified from the representative vertex of one of the X_i's from P. We use the notation G_x for the maximal biclique corresponding to the representative vertex x and, W_x for the weight of the maximal biclique G_x, where x is the representative vertex of some set in P.

5.2 Our Algorithm

Our basic idea for finding a maximum biclique in chain graphs is to find weight of each maximal biclique of G and output the one with the maximum weight. Since G has only k maximal bicliques, so, in order to get the desired biclique, we need to find out the weights of these k bicliques. Since chain graph is a subclass of bipartite permutation graph, we may also use Algorithm 1 to compute a maximum weighted edge biclique of G. The ordering of vertices of G as given in chain graph will also work for bipartite permutation graph. In this way, we will get our desired output in $O(n^2)$-time. Here, we propose an algorithm in which we use a different method to find out the sum of each maximal biclique of G which results in overall running time $O(m+n)$. The difference here is to use partition sum array instead of prefix sum array. The idea behind our algorithm is the following.

```
ALGORITHM 2

1. Find the partitions $P = \{X_1, X_2, \ldots, X_k\}$ and $P' = \{Y_1, Y_2, \ldots, Y_k\}$ from the
   equivalence relation $\sim$, say $R =< r_{X_k}, r_{X_{k-1}}, \ldots, r_{X_1} >$.
2. Calculate the ptsa for each vertex of $Y$.
3. For each vertex $u$ according to the order in which it appears in $R$,
      find the maximal biclique $G_u$ corresponding to the vertex $u$.
      find $W_u$, the weight of biclique $G_u$ using ptsa of vertices from $Y \cap V(G_u)$.
4. Output the maximal biclique $G_{u^*}$ for which $W_{u^*}$ is maximum.
```

Note that we implement step 3 for each vertex $u \in R$ such that $W_{r_{X_j}}$ is calculated using ptsa of vertices of $N(r_{X_j})$. The implementation details are omitted due to space constraints. Proof of correctness of Algorithm 2 follows from the fact that it considers weights of all maximal bicliques of G and any maximum biclique is one of the maximal bicliques. So, we can directly state the following theorem.

Theorem 5. *Algorithm 2 outputs a maximum weighted edge biclique of a chain graph G.*

To analyse the running time of Algorithm 2, we need to bring some notations into consideration. We denote the cardinalities of sets in the partition P and P' by p_i, q_j for X_i, Y_j respectively, i.e. $|X_i| = p_i$ and $|Y_j| = q_j$. Now, we give a result which will be used in analyzing the running time of Algorithm 2.

Lemma 6. *Let G be a chain graph with a partition obtained from the $\sim$ relation defined on X as well as on Y. Then $m \geq kq_1 + (k-1)q_2 + \ldots + q_k$.*

Proof. We know that the relation $\sim$ made the sets from the partitions P and P' to follow the strict inclusion as $N(X_1) \subset N(X_2) \subset \ldots \subset N(X_k)$ and $N(Y_1) \supset N(Y_2) \supset \ldots \supset N(Y_k)$. Since $Y_1 \cup Y_2 \cup \ldots \cup Y_k = Y$ and for $i \neq j$, $Y_i \cap Y_j = \phi$, we can write that $m = \sum_{y \in Y_1} d(y) + \sum_{y \in Y_2} d(y) + \ldots + \sum_{y \in Y_k} d(y) = q_1 \sum_{i=1}^{k} p_i + q_2 \sum_{i=2}^{k} p_i + \ldots + q_k p_k \geq kq_1 + (k-1)q_2 + \ldots + q_k$. The last inequality follows since $|X_i| \geq 1$ for $1 \leq i \leq k$. $\square$

Theorem 6. *Algorithm 2 runs in $O(m+n)$-time.*

Proof. Step 1 will take $O(n)$-time as we have to go through all the vertices of G. To find out the time taken by step 2, we see that we are doing some number of additions during Algorithm 2. For each vertex y of Y, we are doing $d(y)$ number of additions, so overall step 2 takes $\sum_{y \in Y} d(y) = O(m)$ time. Now, to analyse step 3, we see that in our proposed algorithm, we are finding weights of maximal bicliques in the order $W_{r_{X_k}}, W_{r_{X_{k-1}}}, \ldots, W_{r_{X_1}}$. For calculating $W_{r_{X_j}}$, we are doing $\sum_{i=1}^{j} q_i + (j-1)$ number of additions, where j varies from k downto 1. Hence, step 3 performs $\sum_{i=1}^{k} q_i + \sum_{i=1}^{k-1} q_i + \ldots + q_1 + (k-1) + (k-2) + \ldots + 2 + 1 + 0 \leq kq_1 + (k-1)q_2 + \ldots + q_k + \frac{k(k+1)}{2}$ number of additions. Now we know that $m \geq \frac{k(k+1)}{2}$ since $N(X_1) \subset N(X_2) \subset \ldots \subset N(X_{k_1})$ and $N(Y_1) \supset N(Y_2) \supset \ldots \supset N(Y_{k_2})$. Now using Lemma 6, we can say that step 3 will take $O(m)$-time to execute. Clearly, choosing maximum among all the W_u's will take $O(k)$-time. Therefore, the Algorithm 2 returns a maximum weighted edge biclique of G in $O(n) + O(m) + O(m) + O(k) \approx O(m+n)$ time. □

6 Conclusion

Our paper deals with the Maximum Weighted Edge Biclique problem. In this paper, we show that the decision version of the Maximum Weighted Edge Biclique problem remains NP-complete even for complete bipartite graphs, which is a subclass of bipartite graphs. On the positive side, we show that for the input graph G, if the weight of each edge is a positive real number, then the MWEB problem is $O(n^2)$-time solvable for bipartite permutation graphs and $O(m+n)$-time solvable for chain graphs. It will be interesting to try linear-time algorithm for bipartite permutation graphs, as for the unweighted graph this problem is linear-time solvable. One may also try linear-time algorithm for the Maximum Edge Biclique problem in convex bipartite graphs.

References

1. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness, pp. 641–650 (1979)
2. Peeters, R.: The maximum edge biclique problem is NP-complete. Discrete Appl. Math. **131**(3), 651–654 (2003)
3. Dawande, M., Keskinocak, P., Tayur, S.: On the biclique problem in bipartite graphs. Technical report, Carnegie-Mellon University (1997)
4. Dawande, M., Keskinocak, P., Swaminathan, J.M., Tayur, S.: On bipartite and multipartite clique problems. J. Algorithms **41**(2), 388–403 (2001)
5. Hochbaum, D.S.: Approximating clique and biclique problems. J. Algorithms **29**(1), 174–200 (1998)

6. Manurangsi, P.: Inapproximability of maximum biclique problems, minimum k-cut and densest at-least-k-subgraph from the small set expansion hypothesis. Algorithms **11**(1), 10 (2018)
7. Tan, J.: Inapproximability of maximum weighted edge biclique and its applications. In: Agrawal, M., Du, D., Duan, Z., Li, A. (eds.) TAMC 2008. LNCS, vol. 4978, pp. 282–293. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-79228-4_25
8. Nussbaum, D., Pu, S., Sack, J.R., Uno, T., Zarrabi-Zadeh, H.: Finding maximum edge bicliques in convex bipartite graphs. Algorithmica **64**(2), 311–325 (2012)
9. Ambühl, C., Mastrolilli, M., Svensson, O.: Inapproximability results for maximum edge biclique, minimum linear arrangement, and sparsest cut. SIAM J. Comput. **40**(2), 567–596 (2011)
10. Feige, U.: Relations between average case complexity and approximation complexity. In: Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing, pp. 534–543. ACM (2002)
11. Goerdt, A., Lanka, A.: An approximation hardness result for bipartite clique. Electronic Colloquium on Computational Complexity, Report vol. 48 (2004)
12. Alexe, G., Alexe, S., Crama, Y., Foldes, S., Hammer, P.L., Simeone, B.: Consensus algorithms for the generation of all maximal bicliques. Discrete Appl. Math. **145**(1), 11–21 (2004)
13. Dias, V.M., De Figueiredo, C.M., Szwarcfiter, J.L.: Generating bicliques of a graph in lexicographic order. Theoret. Comput. Sci. **337**(1–3), 240–248 (2005)
14. Dias, V.M., de Figueiredo, C.M., Szwarcfiter, J.L.: On the generation of bicliques of a graph. Discrete Appl. Math. **155**(14), 1826–1832 (2007)
15. Gély, A., Nourine, L., Sadi, B.: Enumeration aspects of maximal cliques and bicliques. Discrete Appl. Math. **157**(7), 1447–1459 (2009)
16. Kloks, T., Kratsch, D.: Computing a perfect edge without vertex elimination ordering of a chordal bipartite graph. Inf. Process. Lett. **55**(1), 11–16 (1995)
17. Feige, U., Kogan, S.: Hardness of approximation of the balanced complete bipartite subgraph problem. Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel, Technical report MCS04-04 (2004)
18. Ambuhl, C., Mastrolilli, M., Svensson, O.: Inapproximability results for sparsest cut, optimal linear arrangement, and precedence constrained scheduling. In: 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), pp. 329–337. IEEE (2007)
19. Spinrad, J., Brandstädt, A., Stewart, L.: Bipartite permutation graphs. Discrete Appl. Math. **18**(3), 279–292 (1987)
20. Heggernes, P., van't Hof, P., Lokshtanov, D., Nederlof, J.: Computing the cutwidth of bipartite permutation graphs in linear time. In: Thilikos, D.M. (ed.) WG 2010. LNCS, vol. 6410, pp. 75–87. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16926-7_9
21. Lai, T.H., Wei, S.S.: Bipartite permutation graphs with application to the minimum buffer size problem. Discrete Appl. Math. **74**(1), 33–55 (1997)