Vertex ordering with precedence constraints

Jeff Kinne[1], Akbar Rafiey[2], Arash Rafiey[1], and Mohammad Sorkhpar[1]

[1] Math and Computer Science, Indiana State University, Indiana, USA,
jkinne@cs.indstate.edu, arash.rafiey@indstate.edu,
msorkhpar@sycamores.indstate.edu
[2] University of California San Diego, CA, USA arafiey@ucsd.edu

Abstract. We study *bipartite graph ordering problem*, which arises in various domains such as production management, bioinformatics, and job scheduling with precedence constraints. In the bipartite vertex ordering problem, we are given a bipartite graph $H = (B, S, E)$ where each vertex in B has a cost and each vertex in S has a profit. The goal is to find a minimum K together with an ordering $<$ of the vertices of H, so that $i < j$ whenever $i \in B$ is adjacent to $j \in S$. Moreover, at each sub-order the difference between the costs and profits of the vertices in the sub-order does not exceed K.
The bipartite ordering problem is NP-complete when the weights are one, and the bipartite graph H is a bipartite circle graph. This restricted version was used in the study of the secondary structure of RNA in [11]. Thus, we seek exact algorithms for solving the bipartite ordering problem in classes with simpler structures than bipartite circle graphs. We give non-trivial polynomial time algorithms for finding the optimal solutions for bipartite permutation graphs, bipartite trivially perfect graphs, bipartite cographs, and trees. There are still several classes of bipartite graphs for which the ordering problem could be polynomial, such as bipartite interval graphs, bipartite convex graphs, bipartite chordal graphs, etc.
In addition, we formulate the problem as a linear programming (LP) model and conduct experiments on random instances. We did not find any example with an integrality gap of two or more when limited to bipartite circle graphs with unit weights. No example with an integrality gap of more than 5/2 was found for arbitrary bipartite graphs with random weights. It would be interesting to investigate the possibility of designing a constant approximation algorithm for this problem.

Keywords: Vertex ordering · Bipartite graph classes· Precedence constraints · Energy barrier

1 Introduction and Problem Definition

In this paper, we introduce the bipartite graph ordering problem, motivated by a studying energy barrier problem for transitioning from one DNA secondary structure (one folding) to another secondary DNA structure (with the same sequence and different folding) [14].

The authors of [11] looked at the energy barrier problem as a combinatorial problem on bipartite graphs, and they proved that the problem is NP-complete even on circle bipartite graphs [3] where the input weights are one.

Although the energy barrier problem is NP-complete, several algorithms have been developed to solve it. In [6, 7] heuristic methods were given. In [17, 18], the authors have focused on exact algorithms that take exponential time to solve the problem. The running time of the algorithm in [18] is $n^{O(K)}$, where K is the minimum energy required for this transformation. The worst-case time complexity of the algorithm in [17] is $O(|H|2^{|H|})$, where $|H|$ is the Hamming distance between the two input structures.

The bipartite ordering problem can be viewed as a variation of job scheduling problems with precedence constraints. The goal of our problem is to find the minimum initial budget required so that the vertices of the given bipartite graph are ordered, respecting the precedence and non-negative budget constraints. Job scheduling problems with precedence constraints have received much attention in theoretical computer science and applied mathematics due to their real-world applications in supply chain and production management. The aim of scheduling problems with precedence constraints is to order the jobs while respecting the precedence constraint. The objective function, however, can be different for different scenarios. Most of the work on job scheduling with precedence constraints has focused on minimizing the weighted completion time of the jobs in the single-processor or multi-processor setting [1, 2, 13, 19]. The general problem of finding an ordering of the jobs to schedule that respects the precedence constraints and minimizes the weighted completion time, or cost is NP-complete. Therefore, some approximation algorithms and the hardness of approximation results have been studied to solve the scheduling problem with precedence constraints [1, 2, 19]. There are also some special classes of scheduling problems with precedence constraints that one can find an exact solution in polynomial time [3, 10].

Our results : In this work, we develop algorithms for some special graph classes; trivially perfect bipartite graphs, bipartite cographs, and bipartite permutation graphs; that admit polynomial-time exact solutions for the bipartite graph ordering problem.

We briefly mention that these classes of bipartite graphs have been considered in other optimization problems. Trivially perfect bipartite graphs play an important role in studying the list homomorphism problem. The authors of [5] showed that the list homomorphism problem could be solved in logarithmic space for these bipartite graphs. They were also considered in the fixed parametrized version of the list homomorphism problem in [4]. The subclass of trivially perfect bipartite graphs called *laminar family bipartite graphs* was considered in [15] to

[3] A circle bipartite graph can be represented as two sets A, B where the vertices in A are a set of non-crossing arcs on a real line and the vertices in B are a set of non-crossing arcs from a real line; there is an edge between a vertex in A and a vertex in B if their arcs cross.

obtain a polynomial time approximation scheme (PTAS) for special instances of a job scheduling problem. Each problem instance in [15] is a bipartite graph $H = (J, M, E)$ where J is a set of jobs, and M is a set of machines. For every pair of jobs $u, v \in J$, the set of machines that can process u, v are either disjoint or one is a subset of the other. Bipartite permutation graphs, also known as proper interval bipartite graphs are of interest in graph homomorphism problems [9], and in energy production applications where resources (in our case B vertices) can be assigned (bought) and used (sold) within some successive time steps [12]. There are recognition algorithms for bipartite permutation graphs [9, 16].

1.1 Problem Definition

We are given a bipartite graph $H = (B, S, E)$, where $B \cup S$ is the set of vertices, and E is the set of edges, a subset of $B \times S$. Each vertex $u \in B$ has a negative cost p_u, and each vertex $v \in S$ has a positive cost p_v. The goal of the bipartite graph ordering problem for H is to find a minimum value $bg(H)$ and an ordering $v_1 < v_2 < \cdots < v_n$ of the vertices of H that satisfies:

- *Precedence constraints*: if $(v_i, v_j) \in E$, with $v_i \in B$ and $v_j \in S$ then $v_i < v_j$.
- *Budget constraints:* for every sub-order of the vertices $v_1 < v_2 < \cdots < v_r$, $r < |B \cup S|$, we have $bg(H) + \sum_{k=1}^{k=r} p_{v_k} \geq 0$.

We often use the term *process first* (*process next*) for a subset of vertices of H, and we mean order them before (after) some other vertices of H in the final total ordering. Throughout the paper we denote the input instance by $H = (B, S, E)$ and we assume the cost of vertices in B are negative and the costs of vertices in S are positive. Figure 1 describes an example of the problem when the costs $p_v = 1$, $v \in S$ and $p_u = -1$, $u \in B$.

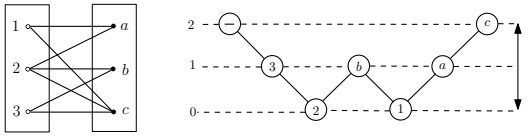

Fig. 1. The left graph is a bipartite circle graph with $B = \{1, 2, 3\}$, and $S = \{a, b, c\}$. Ordering $3, 2, b, 1, a, c$ and $bg(H) = 2$ give an optimal solution when the weights are 1 and -1.

The bipartite graph ordering problem is a natural variation of scheduling problems with precedence constraints. It can be used to model the purchase of supplies and production of goods when purchasing in bulk. Another way to view the problem is that the items in B are training sessions that employees must complete before employees (vertices in S) can begin to work.

No bound on the value of $bg(G)$ when the weights are one . In what follows, we introduce a class of bipartite graphs G with maximum degree at most $\sqrt{|G|}$ ($|G|$ number of vertices) while $bg(G)$ is greater than $|G|/2$. Let $\mathcal{P}$ be a projective plane of order p^2+p+1 with p prime. The projective plane of order $n = p^2+p+1$ consists of n lines, each consisting of precisely $p+1$ points, and n points which each are intersected by precisely $p+1$ lines. We construct a bipartite graph with each vertex in B corresponding to a line from the projective plane, each vertex in S corresponding to a point from the projective plane and a connection from $b \in B$ to $s \in S$ if the point corresponding to s is contained in the line corresponding to B. Vertices in B are given weight -1, and vertices in S are given weight 1. Note that the degree of each vertex in B is $p+1$. One can observe that the neighborhood of every set of $p+1$ vertices in S is at least $p^2 - \binom{p}{2}$. Therefore, to process the first $p+1$ vertices in S we need to process their neighborhood which decreases the budget by at least $p^2 - \binom{p}{2} + p > n/2$; implying that $bg(G) > n/2$.

1.2 Warm-up (simple cases)

In this subsection, we consider simple instances of the problem. This gives a better understanding of the problem and its difficulty. We provide this section to assist the reader in developing an intuition for the problem.

Let $H = (B, S, E)$ be a bipartite graph, and let X be a subset of vertices in H. $\|X\|$ refers to the mass of set X defined by $\sum_{x \in X} |p_x|$, where p_x is the cost of vertex x. We often consider X to be entirely in B or entirely in S.

Proposition 1. *Let $H = (B, S, E)$ be an instance of the bipartite ordering problem where H is a disjoint union of bicliques (bipartite cliques), and random weights. Then computing $bg(H)$ is a polynomial-time task.*

Proof. First, we note that if H is a biclique, then $bg(H) = \|B\|$. Now we consider the case where our graph is a disjoint union of bicliques $H_1, H_2, ..., H_m$ where each $H_i = (B_i, S_i, E_i)$ is a biclique. We start with value $K = \sum_{j=1}^{j=m} \|B_j\|$ as initial budget. Intuition suggests that we should first process those H_i with $\|S_i\| \geq \|B_i\|$, which we call positive sets. If multiple positive sets exist, we process the H_i with minimum $\|B_i\|$ and increase K by $\|S_i\| - \|B_i\|$. Then we are left with bicliques $H_i = (B_i, S_i)$ where $\|B_i\| > \|S_i\|$, which we call negative set.

In processing the remaining negative sets, the budget steadily goes down. As we shall see momentarily, we should process the H_i with the largest $\|S_i\|$ first and decrease K by $\|S_i\| - \|B_i\|$. Suppose on the contrary that $\|S_i\| > \|S_j\|$ but an optimal strategy *opt* processes H_j right before H_i. If K is the budget before this step we first have that $K - \|B_j\| + \|S_j\| \geq \|B_i\|$ because otherwise, there would not be sufficient budget after processing H_j to process H_i. Since we assumed that $\|S_i\| > \|S_j\|$ we have $K - \|B_i\| + \|S_i\| \geq \|B_j\|$. Thus, we could first process H_i and then H_j. We have thus given a method to compute an optimal strategy for a disjoint union of bicliques: first process positive sets in decreasing order of $\|B_i\|$, and then process negative bicliques in decreasing order of $\|S_i\|$. Suppose during this process K' is the minimum value of the current budget. Thus, $bg(H) = \sum_{j=1}^{j=m} \|B_j\| - K'$. ◻

Notice that when bipartite graph H consists of disjoint paths each of length 4 (path P_5) together with random weights, the approach in Proposition 1 does not work, giving some indication of the difficulty of the problem.

Next, we assume the input graph is a tree, and the costs are $p_v = 1$, $v \in S$ and $p_u = -1$, $u \in B$.

Proposition 2. *Let $T = (B, S, E)$ be a tree with weights one. Then $bg(T) = \|B\| - \|S\| + 1$ and finding an optimal ordering is a polynomial time task.*

Proof. Note that any leaf has a single neighbor (or none if it is an isolated vertex). We can thus immediately process any leaf $j \in S$ by processing its parent in the tree and then processing j. This requires an initial budget of only 1. After processing all leaves in S, we are left with a forest where all leaves are in B. We can first remove from consideration any disconnected vertices in B (these can, without loss of generality, be processed last). We are left with a forest H'. We next take a vertex $j_1 \in S$ (which is not a leaf because all leaves in S have already been processed) and process all of its neighbors. After processing j_1 we return 1 unit to the budget. Since H' is a forest, the neighborhood of j_1 has an intersection at most 1 with the neighborhood of any other sold vertex in S. Because we have already processed all leaves in S, we know that only j_1 can be processed after processing its neighbors.

After processing j_1, we may be left with some leaves in S. If so, we deal with these as above. We note that if removing the neighborhood of j_1 does create any leaves in S, then each of these has at least one vertex in B that is its neighbor and is not the neighbor of any of the other leaves in S. When no leaves remain, we pick a vertex $j_2 \in S$ and deal with it as we did j_1.

This process is repeated until all of H' is processed. We note that after initially dealing with all leaves in S, we gain at most a single leaf in S at a time. That is, the budget initially increases as we process vertices in S and process their parents in the tree, and then the budget goes down progressively, only ever temporarily going up by a single unit each time a vertex in S is processed. Note that the budget initially increases, and then once it is decreasing only a single vertex in S is processed at a time. This implies that the budget required for our strategy is $\|B\| - \|S\| + 1$, the best possible budget for T with weights 1 and -1.

2 Definitions and Concepts

In this section, we define key terms and concepts that are relevant to algorithms solving the bipartite graph ordering problem.

Let $H = (B, S, E)$ be a bipartite graph. For a subset, $I \subseteq B$, let $N^*(I)$ be the set of all vertices in S whose entire neighborhood lies in I. For example, in Figure 2, $N^*(J) = F$.

Definition 1 (Prime set). *We say a set $I \subseteq B$ is* prime *if there exists set $X \subset S$, where $N(X) = I$ and there is no X' with $N(X') \subset I$. Equivalently, I is prime if $N^*(I)$ is non-empty and for every $I' \subset I$, $N^*(I')$ is empty.*

In Figure 2, J and I are primes, but $I \cup I_1$ is not prime since $I_1 \subset I \cup I_1$, and $N^*(I_1) = O \neq \emptyset$. Other examples for prime sets in Figure 2 are $J_1 \cup J_2$, J, I, I_1 with $N^*(J_1 \cup J_2) = D$, $N^*(J) = F$, $N^*(I) = L$, and $N^*(I_1) = Q$. Note that the bipartite graph induced by a prime set I and $N^*(I)$ is a bipartite clique. For $X \subset B$, let $H[X \cup N^*(X)]$ be the induced subgraph of H by $X \cup N^*(X)$.

Definition 2 (Positive/Negative set). *A set $I \subseteq B$ is called* positive *if $\|I\| \leq \|N^*(I)\|$ and it is* negative *if $\|I\| > \|N^*(I)\|$.*

Definition 3 (Positive minimal set). *A set $I \subseteq B$ is called* positive minimal *if I is positive, and for every other positive subset I' of I we have $bg(H[I' \cup N^*(I')]) \geq bg(H[I \cup N^*(I)])$.*

For the given graph in Figure 2, I_1 is the only positive minimal set where $N^*(I_1) = O$ contains 7 vertices. Note that, in general, there can be more than one positive minimal set. Positive minimal sets are key in algorithms solving the general case of bipartite graph ordering because these are precisely the sets that we can process first, as can be seen from Lemma 2. In the graph of Figure 2, the positive set I_1 is the first to be processed.

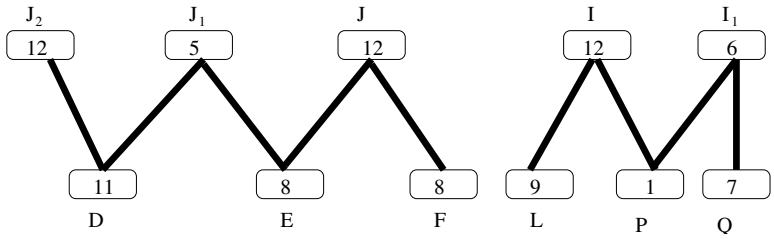

Fig. 2. Each bold line shows a complete connection, i.e. the induced sub-graph by $I \cup L$ is a biclique. The numbers in the boxes are the number of vertices. The sets J_1, J_2, J, I, I_1 are the vertices B, with each vertex having weight -1. The sets D, E, F, L, P, O are the vertices in S, with each vertex having weight 1.

Fixing an order for B in instance $H = (B, S, E)$: Let $\prec$ be an arbitrary order of the vertices in B. We order the positive minimal subsets of B, in the lexicographic order $\prec_L$. This means for two sets $A_1 \subset B$ and $A_2 \subseteq B$, $A_1 \prec_L A_2$ (A_1 is before A_2) if the smallest element of A_1 (according to $\prec$) say a_1 is before the smallest element A_2, say a_2. If $a_1 = a_2$ then $A_1 \prec_L A_2$ if $A_1 \setminus \{a_1\} \prec_L A_2 \setminus \{a_2\}$.

Definition 4 (Closure). *For $I \subseteq B$ of instance $H = (B, S, E)$, let $c\ell(I) = \cup_{i=1}^{r} I_i \cup I$ where each $I_i \subseteq B$, $1 \leq i \leq r$ is the lexicographically first positive minimal subset in $H_i = H \setminus (\cup_{j=0}^{i-1} I_j \cup N^*(\cup_{j=0}^{i-1} I_j))$ ($I_0 = I$) such that in H_i we have $bg(I_i) \leq bg(H) - \cup_{j=0}^{i-1} \|I_j\| + \cup_{j=0}^{i-1} \|N^*(I_j)\|$. Here r is the number of times the process of ordering a positive minimal set can be repeated after I.*

Note that $c\ell(I)$ could be only I, in this case $r = 0$. For instance, consider Figure 2. In the graph induced by $\{J, J_1, J_2, I, D, E, F, L\}$ we have $c\ell(J) = J \cup J_1$.

In what follows, we define trivially perfect bipartite graphs, bipartite cographs, and bipartite permutation graphs. We discuss the key properties that are used in our algorithm for solving the bipartite graph ordering problem on these graph classes.

Definition 5 (Trivially perfect bipartite graph). *A bipartite graph $H = (B, S, E)$ is called* trivially perfect *if it can be constructed using the following operations. If H_1 and H_2 are trivially perfect bipartite graphs, then the disjoint union of H_1 and H_2 is trivially perfect. If $H_1 = (B_1, S_1, E_1)$ and $H_2 = (B_2, S_2, E_2)$ are trivially perfect bipartite graphs then by joining every vertex in S_1 to every vertex in B_2, the resulting bipartite graph is trivially perfect. Notice that a bipartite graph with one vertex is trivially perfect.*

Bipartite graph H is trivially perfect if and only if it does not contain any of the following as an induced sub-graph: C_6, P_6 [5].

Definition 6 (Bipartite cograph). *A bipartite graph $H = (B, S, E)$ is called* cograph *if it can be constructed using the following operations. If H_1 and H_2 are bipartite cographs then the disjoint union of H_1 and H_2 is a bipartite cograph. If $H_1 = (B_1, S_1, E_1)$ and $H_2 = (B_2, S_2, E_2)$ are bipartite cographs, their* complete *join—where every S_1 is joined to every vertex in B_2 and every vertex in B_1 is joined to every vertex in S_2—is a cograph. A bipartite graph with one vertex is a cograph.*

The bipartite cographs studied in [8], and in terms of forbidden obstruction characterization, H is a bipartite cograph if and only if it does not have any of the following graphs depicted in Figure 3 as an induced sub-graph.

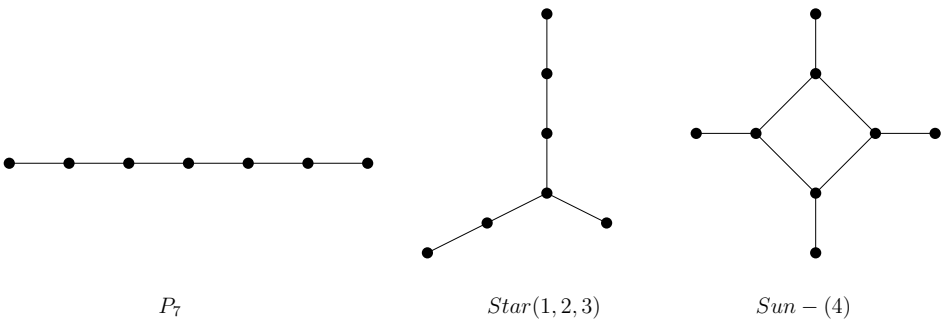

Fig. 3. Forbidden constructions for bipartite cographs.

An example of each type of graph is given in Figure 4. In the left figure (trivially perfect) $I = \{I_1, I_2\}$ and $J = \{I_2, I_3\}$ are prime sets. On the right figure

(bipartite cograph) prime sets are $R_1 = \{J_1, J_2, J_3\}, R_2 = \{J_1, J_2, J_4\}, R_3 = \{J_3, J_4, J_1\}, R_4 = \{J_3, J_4, J_2\}$ are prime sets.

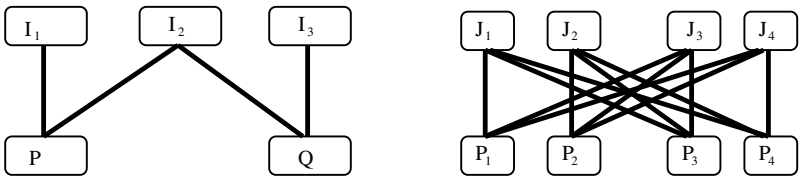

Fig. 4. Each bold line shows a complete connection, i.e. the induced sub-graph by $I_1 \cup P$ is a biclique (complete bipartite graph)

Definition 7 (Bipartite permutation graph). *A bipartite graph $H = (B, S, E)$ is called permutation graph (proper interval bipartite graph) if there exists an ordering $b_1, b_2, \ldots, b_p$ of the vertices in B, and an ordering $s_1, s_2, \ldots, s_q$ of the vertices in S such that if $s_i b_j$ and $s_{i'} b_{j'}$ are edges of H and $j' < j$ and $i < i'$ then $s_i b_{j'}, s_{i'} b_j \in E(H)$. This ordering is called* min-max *ordering [9].*

3 Polynomial Time Cases

We mention the following lemma which is correct for solving the general case of bipartite graphs. Therefore, identifying the prime subsets of B, would lead us to an optimal solution according to the following lemma.

Lemma 1. *Let $H = (B, S, E)$ be a bipartite graph without isolated vertices. Then, there is an optimal strategy to compute $bg(H)$ starting by a prime set.*

Proof. Let $u_1, u_2, \ldots, u_n$ be an optimal ordering that does not start with a prime set. Suppose u_i, $2 \leq i \leq n$, is the first vertex in S. Let $M = \{u_1, u_2, \ldots, u_{i-1}\}$. Let $I \subseteq M$ be the smallest set with $N^*(I) \neq \emptyset$. Note that such I exists since all the adjacent vertices to u_i are among vertices in M. Observe that changing the processing order on vertices in M does not harm optimality. Therefore, we can modify the order, by placing the vertices in I first, without changing the budget. In addition, we can order $N^*(I)$ immediately after the vertices in I.

We continue this section by two lemmas about the positive minimal subsets which are used in designing our polynomial time algorithms, and they can also be used in designing a heuristic.

Lemma 2. *Let $H = (B, S, E)$ be a bipartite graph that can be processed with $bg(H) = K$. If H contains a positive minimal set I with $bg(I) \leq K$ then there is a strategy for H with budget K that begins by processing a positive minimal subset of H.*

Lemma 3. *Suppose that I^+ is a positive subset with $bg(H[I^+ \cup N^*(I^+)]) > K$ and I^- is a negative subset where $bg(H[I^- \cup N^*(I^-)]) \leq K$ and $I^+ \cap I^- \neq \emptyset$. If $bg(H[I^+ \cup I^- \cup N^*(I^+ \cup I^-)]) \leq K$ then $I^+ \cup I^-$ forms a positive subset.*

3.1 Polynomial Time Algorithm for Trivially Perfect Bipartite graph, Bipartite cographs, and Bipartite Permutation Graphs

We continue this section by designing algorithms to solve the bipartite graph ordering for Trivially perfect bipartite graphs and bipartite cograph.

Algorithm 1 BUDGETTRIVIALLYPERFECT (H, K)

Input: Trivially perfect bipartite graph $H = (B, S, E)$, integer K, and decomposition tree T for H
Output: "True" if we can process H with budget at most K, otherwise "False".
if $S = \emptyset$ and $K \geq 0$ **then return** True
if H is a bipartite clique and $\|B\| \leq K$ **then** process H by ordering vertices in B first and then ordering vertices in S after and **return** True
if H is constructed by join operation between $H_1 = (B_1, S_1)$ and $H_2 = (B_2, S_2)$ **then**
    ▷ $bg(H_1), bg(H_2)$ already computed and B_1 and S_2 induce a bipartite clique.
    if $bg(H_1) > K$ **then return** False;
    else if $bg(H_2) > K - \|B_1\| + \|S_1\|$ **then return** False;
    else first process H_1 then process H_2 and **return** True,
if H is constructed by union of H_1 and H_2 **then**
    return COMBINE(H_1, H_2, K)

Our algorithm to solve $bg(H)$ for trivially perfect bipartite graphs and bipartite cographs centers around constructing H as in Definition 5. We view this construction as a tree of operations that are performed to build up the final bipartite graph, and where the leaves of the tree of operations are *bicliques*. If H is not connected, then the root operation in the tree is a disjoint union, and each of its connected components is a trivially perfect bipartite graph (respectively, bipartite cograph). If H is connected, then the root operation is a join. It is easy to construct a decomposition tree for a given trivially perfect bipartite graph. We traverse the decomposition tree in a bottom-up manner. Algorithm 1, takes the input bipartite graph H with weights and a decomposition tree for $H = (B, S, E)$, together with integer K, and it returns yes together with an ordering if $bg(H) \leq K$. To guess the right value for $bg(H)$, we do a binary search between 1 and $\|B\|$.

At each node of the decomposition tree, we assume the optimal budgets for its children have been computed and stored for the graph associated with a particular tree node. If H is constructed by union operation, it requires a merging procedure, which is given in Algorithm 2 called **Combine**. **Combine** takes optimal solutions of two trivially perfect (respectively co-bipartite) graphs

and return an optimal strategy for their union. We give the description of our algorithm and prove its correctness. Recall that we assume every vertex in B has at least one neighbor.

Algorithm 2 COMBINE (H_1, H_2, K)

Input: K and optimal strategies for $H_1 = (B_1, S_1, E_1), H_2 = (B_2, S_2, E_2)$
Output: "True" if we can process $H = H_1 \cup H_2$ with budget at most K, otherwise "False"
if H_1 is empty **then** process H_2 if $bg(H_2) \leq K$ and return True, else return False.
if H_2 is empty **then** process H_1 if $bg(H_1) \leq K$ and return True, else return False.
while $\exists$ positive minimal set I in $H_1 \cup H_2$ with $bg(I) \leq K$ **do**
    Process $c\ell(I)$ and $N^*(I)$.
    if $I \subset H_1$ **then** Set $H_1 \setminus (c\ell(I) \cup N^*(c\ell(N^*(I))$
    else Set $H_2 \setminus (c\ell(I) \cup N^*(c\ell(N^*(I)))$
    Set $K \leftarrow K - \|c\ell(I)\| + \|N^*(c\ell(I))\|$.
Let J_1 be the first prime set in an optimal solution for H_1 and J_2 be the first prime set in optimal solution for H_2.
if $\|J_1\| > K$ OR $bg(H_2) > K - \|c\ell(J_1)\| + \|N^*(c\ell(J_1))\|$ **then**
    Process $c\ell(J_2)$ and $N^*(c\ell(J_2))$
    Call COMBINE$(H_1, H_2 \setminus (c\ell(J_2) \cup N^*(c\ell(J_2))), K - \|c\ell(J_2)\| + \|N^*(c\ell(J_2))\|)$.
else
    Process $c\ell(J_1)$ and $N^*(c\ell(J_1))$
    Call COMBINE $(H_1 \setminus (c\ell(J_1) \cup N^*(c\ell(J_1))), H_2, K - \|c\ell(J_1)\| + \|N^*(c\ell(J_1))\|)$.

Theorem 1. *For trivially perfect bipartite graph H with n vertices the* BUDGETTRIVIALLYPERFECT *algorithm runs in $O(n^2)$ and correctly decides if H can be processed with budget K (Algorithm 1 and Algorithm 2).*

Our algorithm for computing $bg(H)$ when H is bipartite cograph is similar to Algorithm 1. The main difference is in the way we deal with bipartite cograph $H = (B, S, E)$ when it is constructed from two bipartite cographs $H_1 = (B_1, S_1, E_1)$ and $H_2 = (B_2, S_2, E_2)$ by join operation. Recall that in the join operation for bipartite cographs, $H[B_1 \cup S_2]$ and $H[B_2 \cup S_1]$ are bipartite cliques. Observe that, in this case, there are two possibilities for processing H:

- first process entire B_2 then solve the problem for H_1 with budget $K - \|B_2\|$, and at the end process S_2, or
- first process entire B_1 then solve the problem for H_2 with budget $K - \|B_1\|$, and at the end process S_1.

For the case when H is constructed from H_1 and H_2 by union operation, we call COMBINE Algorithm 2. The proof of correctness is almost identical to the proof of Theorem 1.

Theorem 2. *$bg(H)$ can be found in polynomial time for bipartite cograph H.*

Let $H = (B, S, , E)$ be a bipartite permutation graph. Notice that by definition 7, the neighborhood of each vertex in S and B form an interval. Note that the class of circle bipartite graphs $G = (X, Y)$, for which obtaining the optimal budget is NP-complete, contains the class of bipartite permutation graphs. Let $B[i, j]$ denote the interval of vertices $b_i, b_{i+1}, \ldots, b_j$ in B. We compute the optimal budget for every $B[i, j]$. In order to compute $bg(H[B[i, j] \cup N^*(B[i, j])])$ we assume that the optimal strategy starts with some sub-interval J of $B[i, j]$ and it processes $c\ell(J)$, which is indeed an interval. This is because of the property of the min-max ordering. We are left with two disjoint instances, B_1 and B_2 possiblity with some vertices in S with neighbors in $B_1 \cup c\ell(J)$ or in $B_2 \cup c\ell(J)$.

We then argue how to combine the optimal solutions of B_1 and B_2 and obtain an optimal strategy for $B[i, j] \setminus c\ell(J)$. We must consider every possible prime interval J in the range $B[i, j]$ and take the minimum budget. For details, see Algorithm 3.

Algorithm 3 BudgetPermutation (H, K)

Input: Bipartite permutation graph $G = (B, S, E)$ with ordering $<$ on vertices in B, S i.e. $b_1 < b_2 < \cdots < b_n, s_1 < s_2 < \cdots < s_m$ which is a min-max ordering
Output: Computing the budget for G and optimal strategy
for $i = 1$ to $i = n - 1$ **do**
  for $j = 1$ to $j \leq n - i$ **do**
    Let $H' = (B[j, j + i], N^*(B[j, j + i]))$
    Let K' be the minimum number s.t. Optimal-Budget(H', K') is True
    Set $bg(H') = K'$ and process H' be according to Optimal-Budget(H', K')
    Let S_r be the set of vertices with neighbors in both $B[i, i+j], B[i+j+1, n]$)
    Set $H_r = H' \cup S_r$
    Let K' be the minimum number s.t. Optimal-Budget(H_r, K') is True
    Set $bg(H_r) = K'$ and process H_r be according to Optimal-Budget(H_r, K')
    Let S_l be the set of vertices with neighbors in both $B[1, i - 1], B[i, j + i]$
    Set $H_l = H' \cup S_l$
    Let K' be the minimum number s.t. Optimal-Budget(H_l, K') is True
    Set $bg(H_l) = K'$ and process H_l be according to Optimal-Budget(H_l, K')

Theorem 3. Bipartite Ordering Problem *on a bipartite permutation graph with n vertices is solved in time $O(n^6 \log \|B\|)$.*

We heavily use the min-max ordering property to find $bg(H)$ when $H = (B, S, E)$ is a bipartite permutation graph. The next natural superclass of bipartite permutation graphs is the class of convex bipartite graphs. A bipartite graph H is convex if the vertices are ordered in B so that the neighborhood of each vertex in S is an interval.

Problem 1. Let H be a convex bipartite graph. Is it polynomial to decide the optimal value of $bg(H)$?

function OPTIMAL-BUDGET($H = (B, S)$, K)
  Input: Bipartite permutation graph $H = (B, S, E)$ with ordering $<$ on vertices in B, S
  Output: Process H with budget at most K, otherwise "False"
  if $S = \emptyset$ and $K \geq 0$ OR H is a bipartite clique and $\|B\| \leq K$ **then** process H **return** True
  if for every prime $I \subseteq B$, $\|I\| > K$ **then return** False
  while $\exists$ positive prime set I with $bg(I) \leq K$ **do**
    Process $c\ell(I) \cup N^*(c\ell(I))$ and set $H \leftarrow H \setminus c\ell(I) \cup N^*(c\ell(I))$.
    Set $K \leftarrow K - \|c\ell(I)\| + \|N^*(c\ell(I))\|$
  for every prime interval I of H **do**
    Set $B_1 = \{b_1, b_2, \ldots, b_i\}$ and $B_2 = \{b_j, \ldots, b_n\}$ where b_{i+1} is the first vertex of $c\ell(I)$ and b_{j-1} is the last vertex of $c\ell(I)$ in the ordering $<$
    Let S_i, $i = 1, 2$ be the set of vertices in S that have neighbors in B_i ▷ $S_1 \cap S_2 = \emptyset$
    Let $H_1 = H[B_1 \cup S_1]$ and $H_2 = H[B_2 \cup S_2]$.
    Set Flag=Combine($H_1, H_2, K - \|c\ell(I)\| + \|N^*(c\ell(I))\|$)
    if Flag=True **then**
      return Process of $c\ell(I)$ together with process of $H \setminus (c\ell(I) \cup N^*(c\ell(I)))$ by Combine Algorithm.

When the instance $T = (B, S, E)$ is a tree with arbitrary weights. It is easy to see that every prime and positive set form a sub-tree in T; hence, we can find all the positive sets in polynomial time. Moreover, once a prime set I and $N^*(I)$ is removed from T, the remaining becomes a forest. Suppose we have $bg(T_1), bg(T_2), \ldots, bg(T_r)$, where $T_1, T_2, \ldots, T_r$ are disjoint sub-trees in $T \setminus (I \cup N^*(I))$. Now we can use the COMBINE algorithm 2 to combine the optimal strategy of T_1 and T_2 and obtain an optimal strategy for $T_1 \cup T_2$, and then the optimal strategy for $T_1 \cup T_2 \cup T_3$, and eventually an optimal strategy for T and $bg(T)$. Therefore, we have the following proposition.

Proposition 3. *Let $T = (B, S, E)$ together with arbitrary weights be an instance of the bipartite graph ordering problem. If T is a tree, then $bg(T)$ can be computed in polynomial time.*

4 Linear Program Formulation of the problem

Let $H = (B, S, E)$ together with the weight be an instance of the bipartite ordering problem. Suppose there is an ordering $<$ on the vertices of a given bipartite graph $H = (B, S, E)$ in which for every edge uv of H ($u \in B$ and $v \in S$) u is before v in $<$. Then we obtain a strategy for solving the budget minimization on H and decide whether the budget would be K or smaller. With a given value K for $bg(H)$, we translate this ordering process into a linear program as follows. For every pair of vertices $u, v \in B \cup S$, we define variable $0 \leq X_{u,v} \leq 1$. We interpret $X_{u,v} = 1$ (in integral solution) as placing u before

v in a total ordering. The linear program defines as follows. Minimize K such that:

$$\forall u \in B, v \in S, \ \ with \ \ uv \in E, \quad X_{u,v} = 1$$
$$\forall u, v \in B \cup S, u \neq v, \quad X_{u,v} + X_{v,u} = 1$$
$$\forall u, v, w \in B \cup S, \ u \neq v, \quad X_{u,v} + X_{v,w} + X_{w,u} \geq 1$$
$$\forall y \in B \cup S, \quad K + \sum_{u \in B} p_u X_{u,y} + \sum_{v \in S} p_v X_{v,y} \geq 0$$
$$K \geq \min_{v \in S} \{ \sum_{u \in N(v)} |p_u| \}$$
$$\forall u, v \in S \quad if \ \ N(u) \subset N(v) \ \ then \quad X_{u,v} = 1$$
$$\forall u, v \in B, \quad if \ \ N(u) \subset N(v) \ \ then \quad X_{v,u} = 1$$
$$\forall u, v, w \in B \cup S \ \ with \ \ w \notin \{u, v\} \quad if \ \ N(u) = N(v) \ \ then \quad X_{u,w} = X_{v,w}$$

There is a one-to-one correspondence between the optimal solutions of the $bg(H)$ and integral solutions of the above LP. The following table shows the result of our experiment. We have run the LP on random graphs and integer LP on those samples (each having 50 vertices) and taken the maximum ratio of the integral LP by optimal fractional LP.

Graph type	**Circle bipartite graph**	**Circle bipartite graph with weight**	**General bipartite graph**	**General bipartite graph with weight**
Max Ratio	1.818181818	2.813949433	2.179503945	4.311947725
Number of samples	1943	677	1687	932

We pose the following problem.

Problem 2. Does the bipartite ordering problem admit a constant factor approximation ?

Conclusion and Future works The bipartite ordering problem has several applications, there are several open problems which leaves the door open for future research.

References

1. Christoph Ambühl, Monaldo Mastrolilli, Nikolaus Mutsanas, and Ola Svensson. On the approximability of single-machine scheduling with precedence constraints. *Mathematics of Operations Research*, pages 653–669, 2011.
2. Christoph Ambuhl, Monaldo Mastrolilli, and Ola Svensson. Inapproximability results for sparsest cut, optimal linear arrangement, and precedence constrained scheduling. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 329–337. IEEE, 2007.

3. André Berger, Alexander Grigoriev, Pinar Heggernes, and Ruben van der Zwaan. Scheduling unit-length jobs with precedence constraints of small height. *Oper. Res. Lett.*, 42(2):166–172, 2014.
4. Rajesh Chitnis, László Egri, and Dániel Marx. List h-coloring a graph by removing few vertices. In *European Symposium on Algorithms*, pages 313–324. Springer, 2013.
5. László Egri, Andrei Krokhin, Benoit Larose, and Pascal Tesson. The complexity of the list homomorphism problem for graphs. *Theory of Computing Systems*, pages 143–178, 2012.
6. Christoph Flamm, Ivo L Hofacker, Sebastian Maurer-Stroh, Peter F Stadler, and Martin Zehl. Design of multistable rna molecules. *Rna*, pages 254–265, 2001.
7. Michael Geis, Christoph Flamm, Michael T Wolfinger, Andrea Tanzer, Ivo L Hofacker, Martin Middendorf, Christian Mandl, Peter F Stadler, and Caroline Thurner. Folding kinetics of large rnas. *Journal of molecular biology*, pages 160–173, 2008.
8. Vassilis Giakoumakis and Jean-Marie Vanherpe. Bi-complement reducible graphs. *Advances in Applied Mathematics*, pages 389–402, 1997.
9. Gregory Gutin, Pavol Hell, Arash Rafiey, and Anders Yeo. A dichotomy for minimum cost graph homomorphisms. *European Journal of Combinatorics*, pages 900–911, 2008.
10. Berit Johannes. On the complexity of scheduling unit-time jobs with or-precedence constraints. *Oper. Res. Lett.*, 33(6):587–596, 2005.
11. Ján Maňuch, Chris Thachuk, Ladislav Stacho, and Anne Condon. Np-completeness of the direct energy barrier problem without pseudoknots. In *International Workshop on DNA-Based Computers*, pages 106–115. Springer, 2009.
12. Monaldo Mastrolilli and Georgios Stamoulis. Restricted max-min fair allocations with inclusion-free intervals. In *International Computing and Combinatorics Conference*, pages 98–108. Springer, 2012.
13. Rolf H Möhring, Martin Skutella, and Frederik Stork. Scheduling with and/or precedence constraints. *SIAM Journal on Computing*, pages 393–415, 2004.
14. Steven R Morgan and Paul G Higgs. Barrier heights between ground states in a model of rna secondary structure. *Journal of Physics A: Mathematical and General*, page 3153, 1998.
15. Gabriella Muratore, Ulrich M Schwarz, and Gerhard J Woeginger. Parallel machine scheduling with nested job assignment restrictions. *Operations Research Letters*, pages 47–50, 2010.
16. Jeremy Spinrad, Andreas Brandstädt, and Lorna Stewart. Bipartite permutation graphs. *Discrete Applied Mathematics*, pages 279–292, 1987.
17. Hiroki Takizawa, Junichi Iwakiri, Goro Terai, and Kiyoshi Asai. Finding the direct optimal rna barrier energy and improving pathways with an arbitrary energy model. *Bioinformatics*, 36:227–235, 07 2020.
18. Chris Thachuk, Ján Maňuch, Arash Rafiey, Leigh-Anne Mathieson, Ladislav Stacho, and Anne Condon. An algorithm for the energy barrier problem without pseudoknots and temporary arcs. In *Biocomputing 2010*, pages 108–119. World Scientific, 2010.
19. Gerhard J Woeginger. On the approximability of average completion time scheduling under precedence constraints. *Discrete Applied Mathematics*, pages 237–252, 2003.