

Bandwidth of Convex Bipartite Graphs and Related Graphs

Anish Man Singh Shrestha, Satoshi Tayu, and Shuichi Ueno

Department of Communications and Integrated Systems,
Tokyo Institute of Technology
`{anish,tayu,ueno}@lab.ss.titech.ac.jp`

Abstract. It is known that the bandwidth problem is NP-complete for chordal bipartite graphs, while the problem can be solved in polynomial time for bipartite permutation graphs, which is a subclass of chordal bipartite graphs. This paper shows that the problem is NP-complete even for convex bipartite graphs, a subclass of chordal bipartite graphs and a superclass of bipartite permutation graphs. We provide an $O(n)$ -time, 4-approximation algorithm and an $O(n \log^2 n)$ -time, 2-approximation algorithm for convex bipartite graphs with n vertices. For 2-directional orthogonal ray graphs, which is a subclass of chordal bipartite graphs and a superclass of convex bipartite graphs, we provide an $O(n^2 \log n)$ -time, 3-approximation algorithm, where n is the number of vertices.

1 Introduction

A linear layout of a graph G with vertex set $V(G)$ is a bijection $\pi : V(G) \rightarrow \{1, 2, \dots, |V(G)|\}$. The bandwidth of π is defined as $b_\pi(G) = \max\{|\pi(u) - \pi(v)| \mid (u, v) \in E(G)\}$. The *bandwidth* of G is defined as $b(G) = \min b_\pi(G)$ where π ranges over all linear layouts of G . A linear layout π of G is said to be optimal if $b_\pi(G) = b(G)$. Given a graph G and an integer k , the bandwidth problem asks whether the bandwidth of G is at most k . Since the bandwidth of a graph is the maximum bandwidth over all its connected components, we shall consider only connected graphs.

Let G be a bipartite graph with bipartition (X, Y) . The ordering $\prec$ of X is said to fulfill the adjacency property if for each $y \in Y$, the set of neighbors of y consists of vertices that are consecutive in the ordering $\prec$ of X . G is said to be *convex* if there is an ordering of X that fulfills the adjacency property. G is said to be *biconvex* if there is an ordering of X and an ordering of Y that fulfill the adjacency property. A graph G with vertex set $V(G) = \{v_1, v_2, \dots, v_n\}$ and edge set $E(G)$ is called a *permutation graph* if there exists a pair of permutations π_1 and π_2 on $N = \{1, 2, \dots, n\}$ such that for all $i, j \in N$, $(v_i, v_j) \in E(G)$ if and only if $(\pi_1^{-1}(i) - \pi_1^{-1}(j))(\pi_2^{-1}(i) - \pi_2^{-1}(j)) < 0$. A bipartite graph which is also a permutation graph is called a *bipartite permutation graph*. A bipartite graph G is said to be *chordal* if G contains no induced cycles of length greater than 4. A tree is a chordal bipartite graph by definition. A bipartite graph G with bipartition (X, Y) is called a *2-directional orthogonal ray graph* if, in

the xy -plane, there exist a family $\{R_a|a \in X\}$ of horizontal rays (half-lines) extending in the positive x -direction and a family $\{R_b|b \in Y\}$ of vertical rays extending in the positive y -direction, such that two rays R_a and R_b intersect if and only if a and b are adjacent in G . The following relationship between these classes of graphs is known [3,11]: $\{\text{Bipartite Permutation Graphs}\} \subset \{\text{Biconvex Bipartite Graphs}\} \subset \{\text{Convex Bipartite Graphs}\} \subset \{\text{2-directional Orthogonal Ray Graphs}\} \subset \{\text{Chordal Bipartite Graphs}\}$.

Papadimitriou [10] showed that the bandwidth problem is NP-complete for general graphs. Monien [9] showed that it is NP-complete even for generalized caterpillars of hair length at most 3, which are very special trees. This implies that it is also NP-complete for chordal bipartite graphs. On the other hand, Heggenes, Kratsch, and Meister [6] recently showed that the bandwidth of bipartite permutation graphs can be computed in polynomial time. Uehara [14] proposed a faster algorithm for the same problem. Polynomial-time algorithms are also known for chain graphs [7], interval graphs [13], and generalized caterpillars of hair length at most 2 [1]. To the best of our knowledge, there are no prior results that determine the complexities of the bandwidth problem for 2-directional orthogonal ray graphs, convex bipartite graphs, or biconvex bipartite graphs. We show in Section 2.1 that the bandwidth problem is NP-complete even for convex trees and therefore for 2-directional orthogonal ray graphs. In Section 4, we show that the problem can be solved in polynomial time for biconvex trees.

Several results regarding approximation algorithms to compute bandwidth are known for general and special graph classes. Dubey, Feige, and Unger [5] showed that even for generalized caterpillars (and therefore for chordal bipartite graphs), it is NP-hard to approximate the bandwidth within any constant factor. Polynomial-time, constant-factor approximation algorithms are known for few special graph classes such as AT-free graphs and its subclasses as shown by Kloks, Kratsch, and Müller [8]. Convex graphs or 2-directional orthogonal ray graphs are not contained in any of the classes for which approximation algorithms have been reported. We provide in Section 2.2 an $O(n)$ -time, 4-approximation algorithm and an $O(n \log^2 n)$ -time, 2-approximation algorithm for convex bipartite graphs, and in Section 3 an $O(n^2 \log n)$ -time, 3-approximation algorithm for 2-directional orthogonal ray graphs, where n is the number of vertices of a graph.

2 Bandwidth of Convex Bipartite Graphs

2.1 NP-Completeness Result

A *caterpillar* is a tree in which all the vertices of degree greater than one are contained in a single path called a *body*. An edge incident to a vertex of degree one is called a *hair*. A *generalized caterpillar* is a tree obtained from a caterpillar by replacing each hair by a path. A path replacing a hair is also called a hair. Monien [9] showed the following:

Theorem I. *The bandwidth problem is NP-complete for generalized caterpillars of hair length at most 3.* $\square$

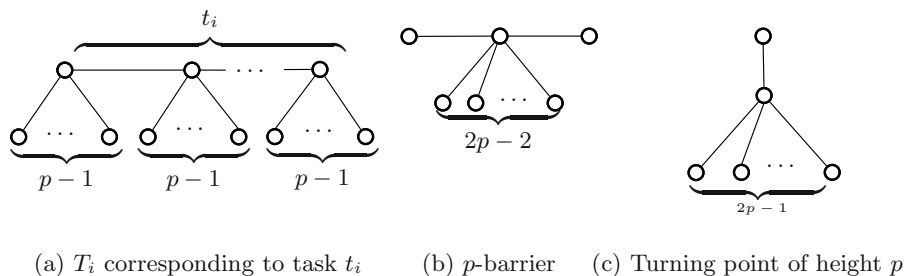


Fig. 1. Components of caterpillar C

We can show the following by a simple modification of the proof of Theorem I:

Theorem 1. *The bandwidth problem is NP-complete for convex trees.*

Proof. (Sketch.) As in the proof of Theorem I, we reduce the multiple processor scheduling problem, which is known to be strongly NP-complete, to our problem. Given a set $T = \{t_1, t_2, \dots, t_n\}$ of tasks (t_i being the execution time of task i), a deadline D , and the size m of a set $\{1, 2, \dots, m\}$ of processors, the multiple processor schedule problem asks whether the tasks in T can be scheduled on the m processors satisfying the deadline D . Corresponding to an instance of this problem, a convex tree C is constructed as follows.

Each task t_i is represented by a caterpillar T_i shown in Figure 1(a). Each processor i is represented by a chain P_i of length $D - 1$. Special components called “barrier” and “turning point” are constructed as shown in Figure 1(b) and Figure 1(c), respectively. C is constructed from these components as shown in Figure 2. Task caterpillars T_i and T_{i+1} are separated by a chain L_i of length Δ . Processor chains P_i and P_{i+1} are separated by a $(p + 1)$ -barrier B_i . A turning point of height $p + 2n + 1$ separates the upper task portion and the lower processor portion. A $(p + 2n + 1)$ -barrier B_0 is attached to the left of P_1 .

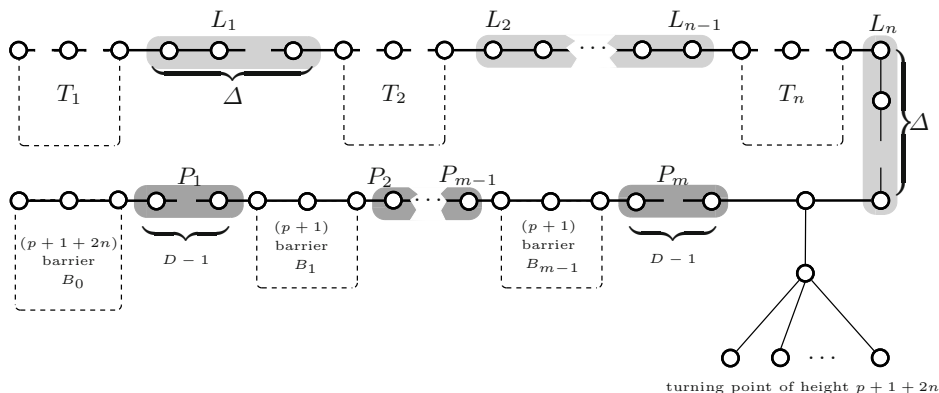


Fig. 2. Instance of bandwidth reduced from multiprocessor scheduling problem

If we remove from C the degree-1 vertices of the turning point, the remaining tree is a caterpillar. It is easy to see that a caterpillar is biconvex, and therefore both partitions of C have an ordering satisfying the adjacency property. If we restore the degree-1 vertices, irrespective of their positions in the ordering of the partition to which they belong, they do not disturb the adjacency property of the ordering of the other partition. Thus C is a convex tree.

If we set the values of Δ and p such that $\Delta = 2 \times (m(D + 2) - 2)$ and $p > 2n(D + 4)$, C can be constructed in time polynomial in n , m , and D ; and it can be shown that the tasks in T can be scheduled on the m processors if and only if C has a bandwidth of $k = p + 1 + 2n$. In fact, this proof is exactly the same as the proof of Theorem I, except only for a slight difference in the structure of the turning point. Therefore, we shall only briefly describe the idea of the proof here. For a detailed treatment, we refer to Monien [9].

If there exists a scheduling of the tasks in T such that tasks $t_{i_1}, t_{i_2}, \dots, t_{i_j}$ are assigned to processor i , then C has bandwidth k and an optimal layout can be achieved by

- (a) laying out the vertices of the body of $T_{i_1}, T_{i_2}, \dots, T_{i_j}$ between barriers B_{i-1} and B_i (between B_{m-1} and turning point, for $i = m$) and
- (b) laying out the vertices of B_0 at the extreme left and those of the turning point at the extreme right.

Conversely, if C has bandwidth k , then in any optimal layout of C ,

- (a) the turning point must be laid out at one of the extreme ends, and barrier B_0 must be laid out at the other,
- (b) all the vertices of the body of each T_j must be laid out between two barriers B_i and B_{i+1} for some i (or B_{m-1} and the turning point for $i = m - 1$), and
- (c) for each i , if between B_i and B_{i+1} (or between B_{m-1} and turning point for $i = m - 1$), bodies of $T_{i_1}, T_{i_2}, \dots, T_{i_j}$ are laid out, then $t_{i_1} + t_{i_2} + \dots + t_{i_j} < D$.

This gives us a scheduling of the tasks in T . □

2.2 Approximation Algorithms for Convex Bipartite Graphs

In this section, we present two algorithms that approximate the bandwidth of convex graphs with worst-case performance ratios of 2 and 4.

Let G be a convex bipartite graph with bipartition (X, Y) and an ordering $\prec$ of X satisfying the adjacency property with $X = \{x_1, x_2, \dots, x_{|X|}\}$ and $x_1 \prec \dots \prec x_{|X|}$. Assume $Y = \{1, 2, \dots, |Y|\}$. Define mappings $s : Y \rightarrow \{1, 2, \dots, n\}$ and $l : Y \rightarrow \{1, 2, \dots, n\}$ such that for $y \in Y$, $x_{s(y)}$ and $x_{l(y)}$ are, respectively, the smallest and largest vertices in $\prec$ adjacent to y . For each vertex $y \in Y$, let $m(y) = \lceil (s(y) + l(y))/2 \rceil$.

2.2.1 Algorithm 1

Our first algorithm is described in Figure 3. Algorithm 1 takes as input G along with the mappings s and l and outputs a linear layout π of G . The idea of the

```

1 Compute  $m(i)$  for each vertex  $i \in Y$ . Add a dummy vertex  $|Y| + 1$  to  $Y$  with
 $m(|Y| + 1) = |X| + 1$ .
2 Let  $\sigma(1), \dots, \sigma(|Y| + 1)$  be the vertices of  $Y$  sorted in the non-decreasing order
of  $m(i)$  value, where  $\sigma$  is a permutation on  $\{1, \dots, |Y| + 1\}$ .
3 Initialize  $i \leftarrow 1, j \leftarrow 1, k \leftarrow 1$ .
4 while ( $j \leq |X|$ )
5     if  $j < m(\sigma(i))$ 
6          $\pi(x_j) = k; j \leftarrow j + 1; k \leftarrow k + 1$ .
7     else if  $j = m(\sigma(i))$ 
8          $\pi(\sigma(i)) = k; i \leftarrow i + 1; k \leftarrow k + 1$ .
9 return  $\pi$ 

```

Fig. 3. Algorithm 1

algorithm is to lay out the vertices of X in the same order as they appear in $\prec$ and insert the vertices of y between them, such that for each $y \in Y$, $\lfloor |N(y)|/2 \rfloor$ vertices of the set $N(y)$ of its neighbors are onto its left and the remaining to its right. Algorithm 1 starts by computing $m(y)$ for each vertex of Y and sorting the vertices according to their $m(i)$ values (Lines 1 and 2). It incrementally assigns labels to the vertices of X in the order in which they appear in $\prec$; stopping at each x_j to check whether there is a vertex in y with $m(y)$ value equal to j , in which case it assigns the current label to y . The process is repeated until all vertices have been labelled (Lines 3 through 8).

We shall next analyze the performance of Algorithm 1. Consider a layout π output by Algorithm 1. The following lemma is easy to see.

Lemma 1. *Algorithm 1 preserves the ordering $\prec$ of X , i.e.,*

$$\pi(x_1) < \pi(x_2) < \dots < \pi(x_{|X|}).$$

□

For a vertex $y \in Y$, let G_y be the subgraph of G induced by the vertices in

$$V_y = \{v | \pi(x_{s(y)}) \leq \pi(v) \leq \pi(y)\} \cup \{v | \pi(y) \leq \pi(v) \leq \pi(x_{l(y)})\}.$$

The *diameter* of a graph is the least integer k such that a shortest path between any pair of vertices of the graph is at most k .

Lemma 2. *For any $y \in Y$, the diameter of G_y is at most 4.*

Proof. Consider G_y corresponding to some $y \in Y$. Let u, v be a pair of vertices of V_y .

Consider first the case that both $u, v \in V_y \cap X$. Since π preserves the ordering $\prec$ of X from Lemma 1, u must be x_i and v must be x_j for some $s(y) \leq i, j \leq l(y)$. Thus both u and v are adjacent to y . Hence the distance between u and v is 2.

Consider next the case that $u \in V_y \cap X$ and $v \in V_y \cap Y$. Vertex v must be adjacent to at least one vertex u' in $V_y \cap X$. If not, then it must be that $l(v) < s(y)$ or $s(v) > l(y)$, which means that $m(v) < s(y)$ or $m(v) > l(y)$,

contradicting the assumption that Algorithm 1 placed v between $x_{s(y)}$ and y or between $x_{l(y)}$ and y . If u' is u , then the distance between u and v is 1. Else, both u and u' are at distance 2 from the earlier case, and therefore the distance between u and v is 3.

Consider finally the case that both $u, v \in V_y \cap Y$. From the earlier case u must be adjacent to some vertex u' , v must be adjacent to some vertex v' . Also u' and v' are both adjacent to y . Hence the distance between u and v is at most 4.

In all the above cases, the shortest path between any pair of vertices does not exceed 4, and thus we have the lemma. $\square$

The following is a well-known lower bound for the bandwidth of a graph [1].

Lemma 3. *For a graph G , $b(G) \geq \max[(N' - 1)/D']$, where the maximum is taken over all connected subgraphs G' of G , N' is the number of vertices of G' , and D' is the diameter of G' .*

We are now ready to show the approximation ratio of Algorithm 1.

Lemma 4. *For layout π returned by Algorithm 1, $b_\pi(G) \leq 4 \times b(G)$.*

Proof. Let (x, y) , $x \in X, y \in Y$ be an edge of G such that $|\pi(x) - \pi(y)| = b_\pi(G)$. Let V_{xy} be the set of vertices v such that v lies between x and y in π . Then $b_\pi(G) = |V_{xy}| - 1$. On the other hand, from Lemmas 2 and 3, we get $b(G) \geq \lceil (|V_y| - 1)/4 \rceil$. Thus we have:

$$\frac{b_\pi(G)}{b(G)} \leq \frac{|V_{xy}| - 1}{(|V_y| - 1)/4}$$

Since the order of X in $\prec$ is preserved in π , x must be $x_{s(y)}$ or $x_{l(y)}$, and therefore $V_{xy} \subseteq V_y$. Thus we get:

$$\frac{b_\pi(G)}{b(G)} \leq 4. \quad \square$$

There exist convex bipartite graphs for which this ratio is asymptotically equal to 4. Figure 4(a) shows an example of such a convex bipartite graph. Let us assume that the mappings s and l provided to Algorithm 1 are based on the left to right ordering of the vertices of the upper partition as shown in Figure 4(a). The layout π returned by Algorithm 1 will lay out between y and x_{n+1} all the vertices $x_i, x'_i, y_{ij}, y'_{ij}$ ($1 \leq i \leq n, 1 \leq j \leq 2n - 1$). Thus $b_\pi(G) = 2n^2 + 2n + 1$. On the other hand, the diameter of this graph is 4, and so from Lemma 3, $b(G) \geq \lceil (2n^2 + 4n + 2)/4 \rceil$. In fact, for large values of n , there is a layout of bandwidth $\lceil (2n^2 + 4n + 2)/4 \rceil$ as shown in Figure 4(b). Thus the approximation ratio $b_\pi(G)/b(G)$ is asymptotically equal to 4.

Algorithm 1 can be implemented to run in $O(|X| + |Y|)$ time. So it follows from Lemma 4 that:

Theorem 2. *Algorithm 1 computes a linear layout of a convex bipartite graph G with bipartition (X, Y) in $O(|X| + |Y|)$ time such that $b_\pi(G) \leq 4 \times b(G)$. $\square$*

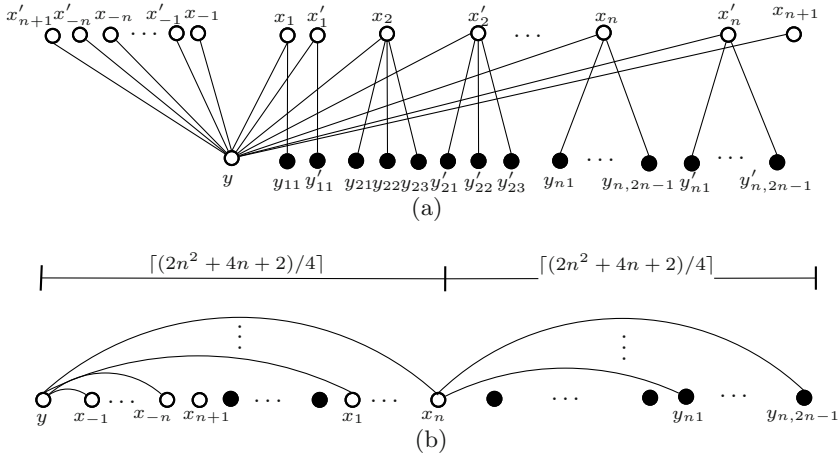


Fig. 4. (a) An example for which the approximation ratio of Algorithm 1 is asymptotically equal to 4. (b) A layout with bandwidth $\lceil (2n^2 + 4n + 2)/4 \rceil$. Only the half right of y is shown as the left half contains the primed counterparts in a symmetric layout. Vertices y_{ij} can be laid out in the region indicated by the black vertices.

If only G , and not s and l , is given, we can compute an ordering satisfying the adjacency property (and thus s and l) in time linear to the number of vertices and edges of the graph, as shown by Booth and Lueker [2]. In that case, the time complexity would be $O(|X| + |Y| + |E|)$, where E is the edge set of G . In the next subsection, we show a different algorithm that runs slower but improves the approximation ratio to 2.

2.2.2 Algorithm 2

Let G be a convex bipartite graph with bipartition (X, Y) and an ordering $\prec$ of X satisfying the adjacency property with $X = \{x_1, x_2, \dots, x_{|X|}\}$ and $x_1 \prec \dots \prec x_{|X|}$. Let s and l be mappings defined at the beginning of Section 2.2. Let G_I be the graph obtained from G by adding to it an edge (y_1, y_2) for each pair $y_1, y_2 \in Y$ having a common neighbor. A graph is said to be an *interval graph* if for every vertex of the graph, there exists an interval on the real line, such that two intervals intersect if and only if their corresponding vertices are adjacent.

Lemma 5. G_I is an interval graph.

Proof. We can see that G_I is an interval graph by defining interval $[i, i]$ for each vertex $x_i \in X$, and interval $[s(y), l(y)]$ for each vertex $y \in Y$. $\square$

Lemma 6. $b(G_I) \leq 2b(G)$

Proof. Let π be an optimal layout of G . Consider the same layout of G_I . For edge $(u, v) \in E(G_I) \cap E(G)$, $\pi(u) - \pi(v) \leq b(G)$. For edge $(u, v) \in E(G_I) \setminus E(G)$, there exists a common neighbor of u and v in G , and therefore $\pi(u) - \pi(v) \leq 2b(G)$. Thus $b_\pi(G_I) \leq 2b(G)$. Since $b(G_I) \leq b_\pi(G_I)$, we get $b(G_I) \leq 2b(G)$. $\square$

Sprague [13] showed that given an interval model of an n -vertex interval graph G and a positive integer k , a layout of bandwidth at most k , if one exists, can be constructed in $O(n \log n)$ time. Thus by doing a binary search between 1 and n , we can compute an optimal layout of G , and therefore we have the following lemma.

Lemma 7. *An optimal layout of an n -vertex interval graph can be computed in $O(n \log^2 n)$ time, if its interval model is given.*

Given a convex bipartite graph G and mappings s and l , Algorithm 2 simply constructs the interval model of G_I and applies the algorithm for interval graphs. The interval model of G_I can be constructed from s and l in time linear to the number of vertices in G , and therefore we have from Lemmas 6 and 7 the following theorem:

Theorem 3. *Algorithm 2 computes a linear layout π of a convex graph G with n vertices in $O(n \log^2 n)$ time such that $b_\pi(G) \leq 2 \times b(G)$.* $\square$

For a path of length 3, whose bandwidth is 1, Algorithm 2 may return a layout of bandwidth 2. Therefore this bound is tight.

3 Bandwidth of 2-Directional Orthogonal Ray Graphs

Since the set of convex bipartite graphs is a proper subset of the set of 2-directional orthogonal ray graphs, the bandwidth problem is NP-complete for 2-directional orthogonal ray graphs, by Theorem 1. In this section, we show a 3-approximation algorithm for 2-directional orthogonal ray graphs.

Let G be a bipartite graph with bipartition (X, Y) , and let $(\prec_X, \prec_Y)$ be a pair of orderings of X and Y , respectively. Two edges (x_1, y_1) and (x_2, y_2) of G are said to *cross* in $(\prec_X, \prec_Y)$ if $x_2 \prec_X x_1$ and $y_1 \prec_Y y_2$. If for every pair (x_1, y_1) and (x_2, y_2) that cross, (x_2, y_1) is also an edge of G , then $(\prec_X, \prec_Y)$ is said to be a *weak ordering* of G . If for every pair (x_1, y_1) and (x_2, y_2) of crossing edges, both (x_1, y_2) and (x_2, y_1) are edges of G , then $(\prec_X, \prec_Y)$ is said to be a *strong ordering* of G .

Spinrad, Brandstädt, and Stewart [12] gave the following characterization of bipartite permutation graphs.

Lemma 8. *A graph G is a bipartite permutation graph if and only if G has a strong ordering.* $\square$

In an earlier work [11], we showed the following characterization of 2-directional orthogonal ray graphs.

Lemma 9. *A graph G is a 2-directional orthogonal ray graph if and only if G has a weak ordering.* $\square$

Given a 2-directional orthogonal ray graph G with bipartition (X, Y) , edge set E , and a weak ordering $(\prec_X, \prec_Y)$ of G , we can construct a graph G_{BP} having vertex set $V_{BP} = X \cup Y$ and edge set $E_{BP} = E \cup E'$, where E' is the set consisting of an edge (x_1, y_2) for every pair of edges (x_1, y_1) and (x_2, y_2) that cross in $(\prec_X, \prec_Y)$.

Lemma 10. G_{BP} is a bipartite permutation graph.

Proof. We will show that G_{BP} is a bipartite permutation graph by showing that $(\prec_X, \prec_Y)$ is a strong ordering of G_{BP} .

Let $e_1 = (x_1, y_1)$ and $e_2 = (x_2, y_2)$ be two edges of G_{BP} that cross in $(\prec_X, \prec_Y)$. We distinguish three cases: **Case 1.** both $e_1, e_2 \in E$, **Case 2.** one each of e_1, e_2 is in $E' \setminus E$ and E , and **Case 3.** both $e_1, e_2 \in E' \setminus E$.

Case 1: Since $(\prec_X, \prec_Y)$ is a weak ordering of G , $(x_2, y_1) \in E$. By definition of E' , $(x_1, y_2) \in E'$. Hence both $(x_2, y_1), (x_1, y_2) \in E_{BP}$.

Case 2: Without loss of generality, assume $e_1 \in E' \setminus E$ and $e_2 \in E$. By definition of E' , $e_1 \in E' \setminus E$ implies that there exist $y'_1 \prec_Y y_1$ and $x'_1 \prec_X x_1$ such that $(x_1, y'_1), (x'_1, y_1) \in E$ and they cross. Since (x_1, y'_1) and (x_2, y_2) also cross, (x_1, y_2) must be in E' and therefore in E_{BP} . To see that $(x_2, y_1) \in E_{BP}$, we further distinguish three cases depending on the order of x'_1 and x_2 in $\prec_X$.

Case 2.1. $x'_1 = x_2$: $(x_2, y_1) = (x'_1, y_1)$ and hence $(x_2, y_1) \in E \subseteq E_{BP}$.

Case 2.2. $x_2 \prec_X x'_1$: since (x'_1, y_1) and (x_2, y_2) cross, $(x_2, y_1) \in E \subseteq E_{BP}$.

Case 2.3. $x'_1 \prec_X x_2$: since (x_1, y'_1) and (x_2, y_2) cross, $(x_2, y'_1) \in E$; and (x_2, y'_1) and (x'_1, y_1) cross, implying that $(x_2, y_1) \in E' \subseteq E_{BP}$.

In all the above subcases of Case 2, we have shown that $(x_2, y_1) \in E_{BP}$, and hence both $(x_2, y_1), (x_1, y_2) \in E_{BP}$.

Case 3: By definition of E' , $e_1 \in E' \setminus E$ implies that there exist $y'_1 \prec_Y y_1$ and $x'_1 \prec_X x_1$ such that $(x_1, y'_1), (x'_1, y_1) \in E$ and they cross. Again by definition of E' , $e_2 \in E' \setminus E$ implies that there exist $y'_2 \prec_Y y_2$ and $x'_2 \prec_X x_2$ such that $(x_2, y'_2), (x'_2, y_2) \in E$ and they cross. Since (x_1, y'_1) and (x'_2, y_2) also cross, (x_1, y_2) must be in E' and therefore in E_{BP} . To see that $(x_2, y_1) \in E_{BP}$, we further distinguish three cases depending on the order of x'_1 and x_2 in $\prec_X$.

Case 3.1. $x'_1 = x_2$: since $(x_2, y_1) = (x'_1, y_1)$, we have $(x_2, y_1) \in E \subseteq E_{BP}$.

Case 3.2. $x_2 \prec_X x'_1$: since $(x'_1, y_1) \in E$ and $(x_2, y_2) \in E' \setminus E$ cross, we have $(x_2, y_1) \in E_{BP}$ from Case 2.

Case 3.3. $x'_1 \prec_X x_2$: we further distinguish three cases, depending on the order of y'_2 and y_1 in $\prec_Y$.

Case 3.3.1. $y'_2 = y_1$: since $(x_2, y_1) = (x_2, y'_2)$, we have $(x_2, y_1) \in E \subseteq E_{BP}$.

Case 3.3.2. $y'_2 \prec_Y y_1$: since (x_2, y'_2) and (x'_1, y_1) cross, $(x_2, y_1) \in E' \subseteq E_{BP}$.

Case 3.3.3. $y_1 \prec_Y y'_2$: since $(x_1, y_1) \in E' \setminus E$ and $(x_2, y'_2) \in E$ cross, we have $(x_2, y_1) \in E_{BP}$ from Case 2.

In all the above subcases of Case 3, we have shown that $(x_1, y_1) \in E_{BP}$, and hence both $(x_2, y_1), (x_1, y_2) \in E_{BP}$.

Thus we have shown that for every $e_1 = (x_1, y_1)$ and $e_2 = (x_2, y_2)$ of G_{BP} that cross in $(\prec_X, \prec_Y)$, both (x_2, y_1) and (x_1, y_2) are also edges of G_{BP} ; and therefore from Lemma 8, G_{BP} is a bipartite permutation graph. $\square$

Lemma 11. $b(G_{BP}) \leq 3 \times b(G)$.

Proof. Let π be an optimal layout of G . Consider the same layout of G_{BP} . For an edge (x, y) of $E(G_{BP}) \cap E(G)$, $|\pi(x) - \pi(y)| \leq b(G)$. For an edge (x, y) of $E(G_{BP}) \setminus E(G)$, there exist vertices $x' \in X$ and $y' \in Y$ such that (y, x') , (x', y) , (y', x) are edges of G , and therefore $|\pi(x) - \pi(y)| \leq 3 \times b(G)$. Thus we have $b_\pi(G_{BP}) \leq 3b(G)$. Since $b(G_{BP}) \leq b_\pi(G_{BP})$, we get $b(G_{BP}) \leq 3 \times b(G)$. $\square$

We shall assume that along with a 2-directional orthogonal ray graph G , a weak ordering $(\prec_X, \prec_Y)$ is also provided as input. If not, then such an ordering can be computed in $O(n^2)$ time, where n is the number of vertices of G [11]. We can construct G_{BP} from G in $O(n^2)$ time. This can be done by first remembering for each $x \in X$, its smallest neighbor y_x in $\prec_Y$ and for each $y \in Y$, its smallest neighbor x_y in $\prec_X$, and then adding to G an edge (x, y) for each pair x, y for which $y_x \prec y$ and $x_y \prec x$. Uehara [14] showed that an optimal layout of an n -vertex bipartite permutation graph having bandwidth k can be computed in $O(n^2 \log k)$ time. Then it follows from Lemma 11 that:

Theorem 4. *There is an $O(n^2 \log n)$ -time algorithm which computes a linear layout π of an n -vertex 2-directional orthogonal ray graph G such that $b_\pi(G) \leq 3 \times b(G)$.* $\square$

Although we do not yet know of an instance for which the ratio is 3, we show in Figure 5(a), graph G for which the algorithm returns a layout of bandwidth 2.5 times the optimal. We can see that the ordering $(\prec_X, \prec_Y)$ of G such that both $\prec_X$ and $\prec_Y$ are the top-to-bottom order of the vertices in Figure 5(a) is a weak ordering. The corresponding bipartite permutation graph G_{BP} is shown in Figure 5(b). The bandwidth of G is 2. G_{BP} contains a complete bipartite graph $K_{4,3}$ induced by the round vertices. Since it is known [4] that $b(K_{m,n}) = \lfloor (m-1)/2 \rfloor + n$ for $m \geq n > 0$, we have $b(G_{BP}) \geq 4$. It can be quickly checked that if $b(G_{BP}) = 4$, then in any optimal layout, the black vertices must be laid out as one contiguous block with two of the remaining round vertices on either side of the block. The square vertex, which is adjacent to three round vertices, cannot be placed anywhere without increasing the bandwidth of the layout. Thus $b(G) > 4$. On the other hand, a layout of bandwidth 5 can be easily obtained.

4 Bandwidth of Biconvex Trees

The *2-claw* is a graph obtained from the complete bipartite graph $K_{1,3}$ by replacing each edge by a path of length 2. The following lemma can be quickly verified.

Lemma 12. *The 2-claw is not a biconvex tree.* $\square$

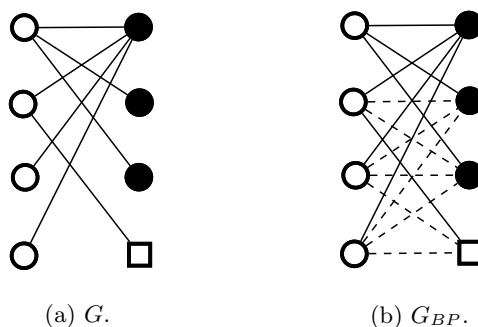


Fig. 5. 2-directional orthogonal ray graph G for which the approximation ratio is 2.5

Biconvex trees can be characterized as follows:

Lemma 13. *A tree T is biconvex if and only if T is a caterpillar.*

Proof. The sufficiency is easy. To prove the necessity, suppose T is a biconvex tree. Let P be a longest path in T . If the length of P is less than five, T is trivially a caterpillar, and so we assume that it is greater than five. Suppose there exists a vertex not in P having degree greater than 1. This implies that T contains the 2-claw as a subtree, contradicting the assumption that T is biconvex graph. Therefore T is a caterpillar.

Assmann, Peck, Sysłó, and Zak showed the following:

Lemma 14. *The bandwidth of an n -vertex generalized caterpillars of hair length at most two can be computed in $O(n \log n)$ time.* $\square$

From Lemma 13 and Lemma 14, we have:

Theorem 5. *The bandwidth of an n -vertex biconvex tree can be computed in $O(n \log n)$ time.* $\square$

We conclude this paper by noting that the complexity of bandwidth problem for biconvex graphs is open.

References

1. Assmann, S.F., Peck, G.W., Sysłó, M.M., Zak, J.: The Bandwidth of Caterpillars with Hairs of Length 1 and 2. *SIAM Journal on Algebraic and Discrete Methods* 2(4), 387–393 (1981)
2. Booth, K.S., Lueker, G.S.: Testing for the Consecutive Ones Property, Interval Graphs, and Graph Planarity Using PQ-tree Algorithms. *Journal of Computer and System Sciences* 13(3), 335–379 (1976)
3. Brandstädt, A., Le, V.B., Spinrad, J.P.: *Graph Classes: A Survey*. Society for Industrial and Applied Mathematics, Philadelphia (1999)
4. Chvátal, V.: A Remark on a Problem of Harary. *Czechoslovak Mathematical Journal* 20(1), 109–111 (1970)

5. Dubey, C., Feige, U., Unger, W.: Hardness Results for Approximating the Bandwidth. *J. Comput. Syst. Sci.* 77, 62–90 (2011)
6. Heggenes, P., Kratsch, D., Meister, D.: Bandwidth of Bipartite Permutation Graphs in Polynomial Time. *J. of Discrete Algorithms* 7(4), 533–544 (2009)
7. Kloks, T., Kratsch, D., Müller, H.: Bandwidth of Chain Graphs. *Inf. Process. Lett.* 68(6), 313–315 (1998)
8. Kloks, T., Kratsch, D., Müller, H.: Approximating the Bandwidth for Asteroidal Triple-Free Graphs. *Journal of Algorithms* 32, 41–57 (1999)
9. Monien, B.: The Bandwidth Minimization Problem for Caterpillars with Hair Length 3 is NP-complete. *SIAM J. Algebraic Discrete Methods* 7(4), 505–512 (1986)
10. Papadimitriou, C.: The NP-Completeness of the Bandwidth Minimization Problem. *Computing* 16, 263–270 (1976)
11. Shrestha, A.M.S., Tayu, S., Ueno, S.: On Orthogonal Ray Graphs. *Discrete Appl. Math.* 158, 1650–1659 (2010)
12. Spinrad, J., Brandstädt, A., Stewart, L.: Bipartite Permutation Graphs. *Discrete Appl. Math.* 18(3), 279–292 (1987)
13. Sprague, A.P.: An $O(n \log n)$ Algorithm for Bandwidth of Interval Graphs. *SIAM J. Discrete Math.* 7(2), 213–220 (1994)
14. Uehara, R.: Bandwidth of Bipartite Permutation Graphs. In: Hong, S.-H., Nagamochi, H., Fukunaga, T. (eds.) *ISAAC 2008*. LNCS, vol. 5369, pp. 824–835. Springer, Heidelberg (2008)