Computing the Cutwidth of Bipartite Permutation Graphs in Linear Time*

Pinar Heggernes[1], Pim van 't Hof[2], Daniel Lokshtanov[1], and Jesper Nederlof[1]

[1] Department of Informatics, University of Bergen,
P.O. Box 7803, N-5020 Bergen, Norway
{pinar.heggernes,daniel.lokshtanov,jesper.nederlof}@ii.uib.no
[2] School of Engineering and Computing Sciences, Durham University,
Science Laboratories, South Road, Durham DH1 3LE, United Kingdom
pimvanthof@gmail.com

Abstract. The problem of determining the cutwidth of a graph is a notoriously hard problem which remains NP-complete under severe restrictions on input graphs. Until recently, non-trivial polynomial-time cutwidth algorithms were known only for subclasses of graphs of bounded treewidth. In WG 2008, Heggernes et al. initiated the study of cutwidth on graph classes containing graphs of unbounded treewidth, and showed that a greedy algorithm computes the cutwidth of threshold graphs. We continue this line of research and present the first polynomial-time algorithm for computing the cutwidth of bipartite permutation graphs. Our algorithm runs in linear time. We stress that the cutwidth problem is NP-complete on bipartite graphs and its computational complexity is open even on small subclasses of permutation graphs, such as trivially perfect graphs.

1 Introduction

A large variety of problems in many different domains can be formulated as graph layout problems [8]. A well known problem of this type is *cutwidth*. Given a graph G and a positive integer k, the cutwidth problem is to decide whether there is an ordering of the vertices of G such that any line inserted between two consecutive vertices in the ordering cuts at most k edges of the graph. The cutwidth of the input graph is the smallest integer for which the question can be answered positively. This problem was first proposed as a model to minimize the number of channels in a circuit [1,14], and later it has found applications in areas like protein engineering [3], network reliability [12], automatic graph drawing [16], and as a subroutine in the cutting plane algorithm for TSP [11].

As most graph problems of practical interest, cutwidth is NP-complete [9], even when input graphs are restricted to planar graphs of maximum degree 3 [15], split graphs [10], unit disk graphs, partial grids [7], and consequently

* This work is supported by the Research Council of Norway and by EPSRC UK grant EP/D053633/1.

D.M. Thilikos (Ed.): WG 2010, LNCS 6410, pp. 75–87, 2010.

bipartite graphs. There is a polynomial-time $O(\log^2 n)$-approximation algorithm for general graphs [13], and a polynomial-time constant factor approximation algorithm for dense graphs [2].

The knowledge on polynomial-time algorithms for the exact computation of cutwidth on restricted inputs is very limited. Cutwidth of certain trivial graph classes, like meshes or complete p-partite graphs, can be computed easily as there exist closed formulas for their cutwidth [8]. Cutwidth of proper interval graphs has a trivial solution following an interval ordering of the vertices [21]. However, there are very few graph classes whose cutwidth is non-trivially computable in polynomial time. Until recently, polynomial-time cutwidth algorithms were known only for subclasses of graphs of bounded treewidth. In particular, Yannakakis [20] gave a sophisticated and technical algorithm for trees (see also [6]). Furthermore, Thilikos et al. gave an algorithm for computing the cutwidth of bounded cutwidth graphs [18], and extended this result to graphs of bounded treewidth and maximum degree [19]. As a recent development, in a WG 2008 paper the study of cutwidth on graph classes containing graphs of unbounded treewidth was initiated, resulting in a linear-time algorithm for computing the cutwidth of threshold graphs [10].

In this paper, we continue this line of research by showing that the cutwidth of a bipartite permutation graph can be computed in linear time. As mentioned above, the cutwidth problem is NP-complete on bipartite graphs, and its computational complexity is open on permutation graphs. Thus bipartite permutation graphs are natural candidates for studying the computational complexity of the cutwidth problem. Our algorithm relies heavily on a characterization of bipartite permutation graphs by strong orderings [17]. We would like to point out that bipartite permutation graphs and threshold graphs are two unrelated subclasses of permutation graphs; the intersection of these two graph classes is restricted to stars. We would also like to point out that bipartite permutation graphs form the first graph class of unbounded clique-width [5] whose cutwidth is shown to be computable in polynomial time.

2 Preliminaries

We consider undirected finite graphs with no loops or multiple edges. For a graph $G = (V, E)$, we denote its vertex set and edge set by V and E, respectively, with $n = |V|$ and $m = |E|$. Let $S \subseteq V$. The subgraph of G induced by S is denoted by $G[S]$. We write $G{-}S$ to denote the graph $G[V \setminus S]$, and we simply write $G{-}v$ instead of $G{-}\{v\}$ in case $S = \{v\}$. For two vertices $u, v \in V$ with $uv \notin E$, we write $G{+}uv$ to denote the graph $(V, E \cup \{uv\})$. The set of *neighbors* of a vertex x of G is $N(x) = \{v \mid xv \in E\}$. The *degree* of x is $d(x) = |N(x)|$. A graph is *connected* if there is a path between any pair of its vertices. A *connected component* of a disconnected graph is a maximal connected subgraph of it.

In a bipartite graph $G = (A, B, E)$, vertex sets A and B are called *color classes*. The partition of the vertex set into color classes of a connected bipartite graph is unique, up to symmetry. Vertices of A and of B are called *A-vertices* and *B-vertices*, respectively. We say that a vertex is *bipartite universal* if it is adjacent to all the vertices of the opposite color class.

An *ordering* of a set A is a one-to-one mapping $\sigma : A \rightarrow \{1, \ldots, |A|\}$. We also use the notation $\sigma = \langle a_1, a_2, \ldots, a_{|A|} \rangle$, meaning that $\sigma(a_i) < \sigma(a_j)$ when $i < j$, where each a_i is a distinct element of A, for $1 \leq i \leq |A|$. Integers $1, 2, \ldots, |A|$ are called the *positions* of σ, and $\sigma(a)$ is the *position* of a in σ. Intuitively, we will refer to the end of the ordering with a_1 as *the left* and the end of the ordering with $a_{|A|}$ as *the right*. For two elements a and a' of A, we say that *a appears before* (or *to the left of*) a' in σ, denoted $a \prec_\sigma a'$, if $\sigma(a) < \sigma(a')$. If $\sigma(a) > \sigma(a')$, then we say that *a appears after* (or *to the right of*) a' in σ and write $a \succ_\sigma a'$. We will also use the notion of a *leftmost*, *rightmost*, and *middle* vertex or neighbor, analogously and intuitively. A subset of k elements of A are *consecutive* in σ if they occupy positions $i+1, \ldots, i+k$, for some i between 0 and $|A| - k$. When we say that we *delete* an element a of A from σ, we get a new ordering in which all elements before a in σ keep their original positions, and the position of each element after a decreases by 1. We denote the new ordering by $\sigma - a$. For any subset of $A' \subseteq A$, we write $\sigma - A'$ to denote the ordering obtained from σ by consecutively deleting all the elements of A' from σ.

A *layout* of a graph $G = (V, E)$ is an ordering of V. We write $\Phi(G)$ to denote the set of all layouts of G. The *rank* of a vertex v with respect to a layout φ, denoted $rank_\varphi(v)$, is the number of neighbors of v appearing after v in φ minus the number of neighbors of v appearing before v in φ, i.e., $rank_\varphi(v) = |\{w \in N(v) \mid w \succ_\varphi v\}| - |\{w \in N(v) \mid w \prec_\varphi v\}|$. Note that the rank of a vertex can be negative. Given layout φ of a graph G and an integer $1 \leq i \leq n$, we define $L(i, \varphi, G) = \{u \in V \mid \varphi(u) \leq i\}$ and $R(i, \varphi, G) = \{u \in V \mid \varphi(u) > i\}$. The *ith gap* of φ is between $L(i, \varphi, G)$ and $R(i, \varphi, G)$, or equivalently, between positions i and $i+1$ of φ. For any set $S \subseteq V$, we define the *cut* of S to be $\theta(S, G) = \{uv \in E \mid u \in S, v \notin S\}$. The *cut of G at the ith gap* of φ is defined as $\theta(i, \varphi, G) = \{uv \in E \mid u \in L(i, \varphi, G) \wedge v \in R(i, \varphi, G)\}$. Note that by definition $\theta(i, \varphi, G) = \theta(L(i, \varphi, G), G)$. We call an edge set $\theta \subseteq E$ a *cut* of φ if $\theta = \theta(i, \varphi, G)$ for some $i \in \{1, 2, \ldots n-1\}$. The *size* of a cut θ is $|\theta|$. The *cutwidth of a layout* φ of G is $\mathrm{cw}\varphi(G) = \max_{1 \leq i \leq n} |\theta(i, \varphi, G)|$. A cut $\theta(i, \varphi, G)$ with $|\theta(i, \varphi, G)| = \mathrm{cw}_\varphi(G)$ is called a *worst cut* of φ. The *cutwidth* of G is $\mathrm{cw}(G) = \min_{\varphi \in \Phi(G)}\{\mathrm{cw}_\varphi(G)\}$, where the minimum is taken over all layouts of G. An *optimal layout* of G is a layout φ such that $\mathrm{cw}(G) = \mathrm{cw}_\varphi(G)$. The cutwidth of a graph G equals the maximum cutwidth over all connected components of G.

As the name already indicates, bipartite permutation graphs are permutation graphs that are bipartite. For the definition and properties of permutation graphs, we refer to [4]. The study of bipartite permutation graphs was initiated by Spinrad et al. in [17]. They present two characterizations of bipartite permutation graphs, leading to a linear-time recognition algorithm of this class as well as polynomial-time algorithms for some NP-complete problems restricted to bipartite permutation input graphs.

A *strong ordering* (σ_A, σ_B) of a bipartite permutation graph $G = (A, B, E)$ consists of an ordering σ_A of A and an ordering σ_B of B such that for all

$ab, a'b' \in E$, where $a, a' \in A$ and $b, b' \in B$, $a \prec_{\sigma_A} a'$ and $b' \prec_{\sigma_B} b$ implies that $ab' \in E$ and $a'b \in E$. An ordering σ_A of A has the *adjacency property* if, for every $b \in B$, $N(b)$ consists of vertices that are consecutive in σ_A. The ordering σ_A has the *enclosure property* if, for every pair b, b' of vertices of B with $N(b) \subseteq N(b')$, the vertices of $N(b') \setminus N(b)$ appear consecutively in σ_A, implying that b is adjacent to the leftmost or the rightmost neighbor of b' in σ_A.

Theorem 1 ([17]). *The following statements are equivalent for a bipartite graph* $G = (A, B, E)$.

1. *G is a bipartite permutation graph.*
2. *G has a strong ordering.*
3. *There exists an ordering of A which has the adjacency and enclosure properties.*

A strong ordering of a bipartite permutation graph can be computed in linear time [17]. If the graph G in Theorem 1 is connected, then it follows from the proof of Theorem 1 in [17] that we can combine statements 2 and 3 in Theorem 1 as follows.

Lemma 1 ([17]). *Let (σ_A, σ_B) be a strong ordering of a connected bipartite permutation graph $G = (A, B, E)$. Then both σ_A and σ_B have the adjacency and enclosure properties.*

3 Cutwidth of Bipartite Permutation Graphs

In this section we prove that the cutwidth of bipartite permutation graphs can be computed in linear time. The complete algorithm is given in the proof of Theorem 2. The main ingredient is an algorithm that we call MinCutBPG. This algorithm takes as input a connected bipartite permutation graph G and a strong ordering of G, and it outputs an optimal layout of G. We will spend most of this section describing and proving the correctness of Algorithm MinCutBPG. Before we give the algorithm, we define an operation to modify a given layout in an intuitive way. Given a layout φ of a graph, when we *move* a vertex v from position i to position j, with $i < j$, only vertices in positions from i to j are affected. We get a new layout φ' in which v gets position $\varphi'(v) = j$, the vertex x with $\varphi(x) = j$ gets position $\varphi'(x) = j - 1$, and each of the other affected vertices decrease their positions by 1, similarly. All other vertices have the same position in φ' as they had in φ. What we described is a *move toward the right*. A *move toward the left* is defined symmetrically.

3.1 Description of Algorithm MinCutBPG

We now give an outline of Algorithm MinCutBPG, which takes as input a connected bipartite permutation graph $G = (A, B, E)$ and a strong ordering (σ_A, σ_B) of G. It outputs an optimal layout φ of G. Let $A = \{a_1, \dots, a_s\}$ where $a_1 \prec_{\sigma_A}$

$\cdots \prec_{\sigma_A} a_s$, and let $B = \{b_1, \ldots, b_t\}$ where $b_1 \prec_{\sigma_B} \cdots \prec_{\sigma_B} b_t$. The vertices of A will appear in the final layout φ in the same order as they appear in σ_A. Similarly, the order in which the vertices of B appear in φ corresponds to the order in which they appear in σ_B.

Before deciding where the vertices of A will appear in φ with respect to the vertices of B, the algorithm first assigns the vertices of B to "boxes". There are two types of boxes: a box X_i for every vertex $a_i \in A$, and a box $X_{i,i+1}$ for every pair of consecutive vertices $a_i, a_{i+1} \in A$. Recall that the neighbors of any vertex $b \in B$ appear consecutively in σ_A by Lemma 1. If b has even degree and its two middle neighbors are a_i and a_{i+1}, then b is assigned to box $X_{i,i+1}$. If b has odd degree and its middle neighbor is a_i, then b is assigned to box X_i. For convenience, we also define the boxes $X_{0,1} = \emptyset$ and $X_{s,s+1} = \emptyset$. Observe that some boxes might be empty and the collection of non-empty boxes is a partition of B. The following observation is a direct consequence of Lemma 1, the properties of a strong ordering, and the definition of boxes.

Observation 1. *Given a connected bipartite permutation graph $G = (A, B, E)$ with $|A| = s$ and a strong ordering (σ_A, σ_B), where $\sigma_A = \langle a_1, a_2, \ldots, a_s \rangle$, let boxes $X_{0,1}, X_1, X_{1,2}, \ldots, X_s, X_{s,s+1}$ be defined as above. Then we have the following:*

1. *every vertex of X_i appears before every vertex of $X_{i,i+1}$ in σ_B, and every vertex of $X_{i,i+1}$ appears before every vertex of X_{i+1} in σ_B, for $1 \le i \le s$;*
2. *$N(b) = N(b')$ for any two vertices b and b' appearing in the same box.*

We start with an initial layout of G in which a_1 is placed first, vertices of X_1 are placed in the immediately following positions, vertices of $X_{1,2}$ are placed in the next positions, then a_2 is placed, followed by vertices of X_2, $X_{2,3}$, $\{a_3\}$, X_3, …, $\{a_{s-1}\}$, X_{s-1}, $X_{s-1,s}$, $\{a_s\}$, and X_s. Within each box, the vertices of B belonging to that box are ordered according to σ_B. For $1 \le i \le s$, a_i appears just before the vertices of box X_i. To define and obtain the final layout φ, we just need to move each a_i to its final position. This will be one of the initial positions of $\{a_i\} \cup X_i$. As a consequence, we can observe already now that, for every $b \in B$, $rank_\varphi(b) \in \{-1, 0.1\}$. The ranks of the A-vertices might have a larger range of values. Let i be any index satisfying $1 \le i \le s$. Recall that $rank_\varphi(a_i)$ depends on the position where a_i is placed: the further to the left a_i appears, the higher its rank. The algorithm moves a_i in such a way that $rank_\varphi(a_i)$ is as close to 0 as possible, i.e., the value of $|rank_\varphi(a_i)|$ is as small as possible, subject to the condition that the position of a_i is one of the initial positions of $\{a_i\} \cup X_i$. This is done in the following way. Note first that the set of possible positions for a_i does not intersect with the set of possible positions for any other A-vertex a_j with $i \neq j$. Furthermore, $rank_\varphi(a_i)$ is only dependent on the neighbors of a_i and no two A-vertices are adjacent. Therefore, the placement of each a_i among the positions of $\{a_i\} \cup X_i$ can be decided independently of the placements of the other A-vertices. By Lemma 1, the neighbors of a_i appear consecutively in σ_B. If a_i has odd degree then let b be the middle neighbor of a_i in σ_B. If a_i has even degree then let b be the right one of the two middle neighbors of a_i. If $b \in X_i$, then we move a_i to the position just before the position of b. If b appears in a box to the left of X_i then we do not move a_i. If b appears in a box to the right of X_i then we move a_i to the last position among the positions of X_i. Thus, if a_i is placed between two vertices of X_i then its rank is 0 or 1. If a_i is placed before or after all vertices of X_i then its rank can be higher or lower. This completes the definition and computation of φ.

We make the following observations about the layout φ generated by Algorithm MinCutBPG, which are direct consequences of Lemma 1.

Observation 2. *Let $G = (A, B, E)$ be a connected bipartite permutation graph and let (σ_A, σ_B) be a strong ordering of G, where $\sigma_A = \langle a_1, a_2, \ldots, a_s \rangle$. Let φ be the layout of G generated by Algorithm* MinCutBPG *on input G and (σ_A, σ_B). Then, for $1 \leq i \leq s$, we have the following:*

1. *for any $b \in X_{i,i+1}$, $rank_\varphi(b) = 0$;*
2. *for any $b \in X_i$, $rank_\varphi(b) = 1$ if $b \prec_\varphi a_i$ and $rank_\varphi(b) = -1$ if $a_i \prec_\varphi b$;*
3. *every $b \in X_{i-1,i} \cup X_i \cup X_{i,i+1}$ is adjacent to a_i;*

3.2 Correctness of Algorithm MinCutBPG

We show that Algorithm MinCutBPG produces an optimal layout when the input is a connected bipartite permutation graph and a strong ordering of that graph. We assume for contradiction that there is a connected bipartite permutation graph G for which the algorithm outputs a layout φ such that $\mathrm{cw}_\varphi(G) > \mathrm{cw}(G)$. Such a graph is called a *counterexample*, and we write $\mathcal{G}$ to denote the set of all counterexamples. Let $\mathcal{G}' \subseteq \mathcal{G}$ be the set of counterexamples having the minimum number of vertices among all counterexamples, and let $\mathcal{G}'' \subseteq \mathcal{G}'$ be the set of graphs in $\mathcal{G}'$ having the maximum number of edges among all graphs in $\mathcal{G}'$. A graph in $\mathcal{G}''$ is called a *tight counterexample.* If there exists a counterexample, then there also exists a tight counterexample.

For the statements and the proofs of the following lemmas, let $G = (A, B, E)$ with $E \neq \emptyset$ be a connected bipartite permutation graph that is a tight counterexample, and let (σ_A, σ_B) be a strong ordering of G such that $\sigma_A = \langle a_1, \ldots, a_s \rangle$ and $\sigma_B = \langle b_1, \ldots, b_t \rangle$. Furthermore, let $\varphi = \langle v_1, \ldots, v_n \rangle$ be the layout of G generated by Algorithm MinCutBPG on input G and (σ_A, σ_B).

Lemma 2. *Let $\theta(j, \varphi, G)$ be a worst cut of φ. Then we have the following:*

1. *a_1 is adjacent to the rightmost B-vertex of $L(j, \varphi, G)$;*
2. *b_1 is adjacent to the rightmost A-vertex of $L(j, \varphi, G)$;*
3. *a_s is adjacent to the leftmost B-vertex of $R(j, \varphi, G)$;*
4. *b_t is adjacent to the leftmost A-vertex of $R(j, \varphi, G)$.*

Proof. We only prove claim 1; the proofs of claims 2, 3, and 4 are very similar and have therefore been omitted. Let $\theta = \theta(j, \varphi, G)$ and let b be the rightmost B-vertex of $L = L(j, \varphi, G)$. If $b \prec_\varphi a_1$ then all B-vertices in L appear before a_1 in φ, and b is the vertex just before a_1 in φ, implying that $\varphi(b) = \varphi(a_1) - 1$.

Hence $b \in X_1$, and by Observation 2, $a_1 b \in E$. Now assume that $a_1 \prec_\varphi b$, and suppose for contradiction that a_1 is not adjacent to b. Note that this means that $b \notin X_1$, since every vertex in box X_1 is adjacent to a_1 by Observation 2. We claim that $G' = G - (\{a_1\} \cup X_1)$ is a counterexample, contradicting the assumption that G is a tight counterexample. Observe that G' is a connected bipartite permutation graph and $(\sigma_A - a_1, \sigma_B - X_1)$ is a strong ordering of G'. We will prove the claim by showing that θ is a cut of the layout φ' returned by Algorithm MinCutBPG on input G' and $(\sigma_A - a_1, \sigma_B - X_1)$. Since $a_1 b \notin E$, a_1 has no neighbors in $R = R(j, \varphi, G)$ as a result of the properties of a strong ordering. None of the vertices in X_1 has a neighbor in R either, because they are adjacent to a_1 only. Therefore, θ is a cut of $\varphi - (\{a_1\} \cup X_1)$. We will show that all vertices of $L \setminus (\{a_1\} \cup X_1)$ that appear to the left of b in $\varphi - (\{a_1\} \cup X_1)$ also appear to the left of b in φ'. This will imply that θ is a cut of φ' as well. Clearly, the relative orderings of the A-vertices and of the B-vertices are the same in φ' as in φ. Let us analyze how the deletion of the vertices in $\{a_1\} \cup X_1$ can affect the ranks of vertices and the boxes that they belong to. Deleting $\{a_1\} \cup X_1$ does not change the rank of any A-vertex or the rank of b, since these vertices were not adjacent to any of the vertices in $\{a_1\} \cup X_1$. Consequently, b appears in the same box after the deletion of a_1 as it did before. Let $a \neq a_1$ be the rightmost A-vertex of L; note that a might not be defined in case a_1 is the only A-vertex of L. Either a or b is the rightmost vertex of L in φ. In either case, since the ranks of a and b did not change, a and b have the same relative order to each other in φ' as in φ. The only vertices whose ranks might change by the deletion of $\{a_1\} \cup X_1$ are the B-vertices of L that were adjacent to a_1. However, these vertices cannot appear to the right of b in φ', as the algorithm respects the strong ordering $(\sigma_A - a_1, \sigma_B - X_1)$. As a result, the set of vertices that appear to the left of b is the same in φ' as in φ, which means that θ is a cut of φ'. Since $\mathrm{cw}(G') \leq \mathrm{cw}(G)$ and the size of the cut did not change, we conclude that G' is a counterexample with at least one fewer vertex than G, giving us the desired contradiction.

Lemma 3. *Let $\theta(j, \varphi, G)$ be a worst cut of φ. Then both $G[L(j, \varphi, G)]$ and $G[R(j, \varphi, G)]$ are complete bipartite graphs.*

Proof. Let a and b be the rightmost A-vertex and B-vertex of $L = L(j, \varphi, G)$, respectively. By Lemma 2, a_1 is adjacent to b and b_1 is adjacent to a. By the definition of a strong ordering, a_1 is adjacent to b_1 and a is adjacent to b. Since G is connected, and σ_A and σ_B have the adjacency property by Lemma 1, a and a_1 are adjacent to all B-vertices in L, and b and b_1 are adjacent to all A-vertices in L. As a result, every vertex of $A \cap L$ is adjacent to every vertex of $B \cap L$. This means that $G[L(j, \varphi, G)]$ is complete bipartite. By symmetry the same holds for $G[R(j, \varphi, G)]$.

Lemma 4. *There is a worst cut $\theta(j, \varphi, G)$ of φ such that v_j and v_{j+1} belong to different color classes.*

Proof. Let $L = L(j, \varphi, G)$ and let $R = R(j, \varphi, G)$. Assume that either L or R, say L, contains vertices of only one color class. Since G is connected, G

contains vertices from both color classes. Let us consider the smallest index $k \geq j$ such that there is a vertex of the other color class in position $k+1$. Then $|\theta(k, \varphi, G)| \geq |\theta(j, \varphi, G)|$ because $L \subseteq L(k, \varphi, G)$, there are no edges between the vertices of $L(k, \varphi, G)$, and each vertex of $L(k, \varphi, G)$ has a neighbor in $R(k, \varphi, G)$. Hence we can conclude that there is a worst cut at the gap between two vertices of opposite color. The case where R contains only vertices of one color class is completely symmetric. For the rest of the proof, assume that both L and R contain vertices of both color classes.

Assume first that both v_j and v_{j+1} are B-vertices. Let a_i be the rightmost A-vertex in L, which means that a_{i+1} is the leftmost A-vertex in R. Both $b = v_j$ and $b' = v_{j+1}$ are between a_i and a_{i+1}; more precisely, $a_i \prec_\varphi b \prec_\varphi b' \prec_\varphi a_{i+1}$. If $rank_\varphi(b) = 1$ then $b \in X_{i+1}$ by Observation 2. Then by Observation 1, $b' \in X_{i+1}$ as well, and consequently $rank_\varphi(b') = 1$. Thus we can conclude that b and b' have the same neighborhood and they have one more neighbor in R than in L. In this case $\theta(j, \varphi, G)$ cannot be a worst cut, because the cut just to the right of b' has larger size. Therefore, $rank_\varphi(b) \leq 0$, which means that b has at least as many neighbors to the left as it has to the right. Since b has no neighbors appearing between a_i and b, the cut just to the right of a_i is of size at least $|\theta(j, \varphi, G)|$. Hence we can take that cut as the worst cut. Consequently there is a worst cut at the gap between an A-vertex and a B-vertex.

Assume now that both v_j and v_{j+1} are A-vertices, say a_i and a_{i+1}. First we show that in this case both a_i and a_{i+1} are bipartite universal. Assume for contradiction that this is not true, and let b be the leftmost B-vertex in R which is not a neighbor of a_i. We claim that $G' = G + a_i b$ is also a counterexample, contradicting the assumption that G is a tight counterexample. Recall that $G[L]$ and $G[R]$ are complete bipartite graphs due to Lemma 3. Now observe that G' is a bipartite permutation graph and (σ_A, σ_B) is a strong ordering of G'. Let φ' be the layout computed by Algorithm MinCutBPG on input G' and (σ_A, σ_B). Let us analyze how the layout φ can change to φ' due to the addition of edge $a_i b$. Observe that $X_{i,i+1}$ is empty before the addition of edge $a_i b$, since a_i and a_{i+1} are consecutive in φ. When we add edge $a_i b$, vertex b gets one more neighbor to the left, and thus might appear in a box further to the left than the box it was in before. By Observation 1 we know that b was not in X_i or $X_{i,i+1}$ before the addition of edge $a_i b$. Now it can enter $X_{i,i+1}$ but it cannot enter X_i, since it only gained one more neighbor. This means that it can move past a_{i+1} toward the left, but it cannot move past a_i. Thus $L(j, \varphi', G') = L$ and $R(j, \varphi', G') = R$, although some vertices in R might have changed positions. Consequently, $\theta(j, \varphi', G') = \theta(j, \varphi, G) \cup \{a_i b\}$ is a cut of φ', which means that φ' has a cut whose size is 1 more than a worst cut of φ. Since $\mathrm{cw}(G') \leq \mathrm{cw}(G) + 1$, G' is a counterexample, contradicting the assumption that G is a tight counterexample. Thus there cannot be a B-vertex in R that a_i is not adjacent to. By Lemma 3 we know that a_i is adjacent to all B-vertices in L, and hence a_i is bipartite universal. By symmetry and with similar arguments, a_{i+1} is also bipartite universal. This means that $rank_\varphi(a_i) = rank_\varphi(a_{i+1})$. If this rank is negative, then the cut at the $(j-1)$th gap is a larger cut than $\theta(j, \varphi, G)$ since a_i and a_{i+1} have more neighbors in L than in R. Symmetrically, if this rank is positive then the cut at the $(j+1)$th gap is a larger cut. Therefore $rank_{\varphi}(a_i) = rank_{\varphi}(a_{i+1}) = 0$, because otherwise we get a contradiction to the assumption that $\theta(j, \varphi, G)$ is a worst cut. This means that a_i and a_{i+1} have as many neighbors in L as they have in R. Since a_i and a_{i+1} are both bipartite universal and they are not adjacent to each other, the cut at the $(j-1)$th gap and the cut at the $(j+1)$th gap have the same size as $\theta(j, \varphi, G)$. Hence we can take one of these cuts as a worst cut. We can repeat this argument until we reach a B-vertex on the other side of a worst cut.

Lemma 5. *There is a worst cut $\theta(j, \varphi, G)$ of φ such that both v_j and v_{j+1} are bipartite universal.*

Proof. By Lemma 4, we know that there is a worst cut $\theta = \theta(j, \varphi, G)$ such that v_j and v_{j+1} belong to different color classes. Let us now show that both v_j and v_{j+1} are bipartite universal. Let $a = v_j \in A$ and let $b = v_{j+1} \in B$. By Lemma 3 we know that a is adjacent to every B-vertex in L and b is adjacent to every A-vertex in R. If $ab_t \in E$ then a is bipartite universal as a result of the properties of a strong ordering. If $ab_t \notin E$, we claim that $G' = G - b_t$ is also a counterexample, contradicting the assumption that G is a tight counterexample. We observe that G' is a bipartite permutation graph with strong ordering $(\sigma_A, \sigma_B - b_t)$. Since b_t has no neighbors in L as a result of the properties of a strong ordering, θ is a cut of $\varphi - b_t$. Let φ' be the layout computed by MinCutBPG on input G' and $(\sigma_A, \sigma_B - b_t)$. Since no B-vertex was adjacent to b_t, every remaining B-vertex appears in the same box after the deletion of b_t as it did before. However, an A-vertex a_i that was adjacent to b_t might move one position to the left inside the box X_i. Hence a_i can move past b toward the left, but it cannot move past a, since the algorithm respects the strong ordering. Consequently, all vertices of L to the left of a in φ appear also to the left of a in φ'. Thus θ is a cut of φ'. Since $\text{cw}(G') \leq \text{cw}(G)$ and the size of the cut did not change, G' is a counterexample, contradicting the assumption that G is a tight counterexample. Hence a is bipartite universal. To show that b is bipartite universal we use similar arguments: by symmetry, if $a_1 b \notin E$ then $G' = G - a_1$ is a counterexample as well. Finally, the case where $v_j \in B$ and $v_{j+1} \in A$ is completely symmetric.

Corollary 1. *There is a worst cut $\theta(j, \varphi, G)$ of φ such that v_j and v_{j+1} belong to different color classes and they are both bipartite universal.*

Proof. The proof of Lemma 5 takes a cut as mentioned in Lemma 4, and shows the claim of Lemma 5 using the same cut. Hence, there is a cut that satisfies both lemmas at the same time, and the corollary follows.

The proof of the following lemma has been omitted due to page restrictions.

Lemma 6. *There is a worst cut $\theta(j, \varphi, G)$ such that there are $\lfloor |A|/2 \rfloor$ A-vertices and $\lceil |B|/2 \rceil$ B-vertices on one side of the jth gap of φ, and there are $\lceil |A|/2 \rceil$ A-vertices and $\lfloor |B|/2 \rfloor$ B-vertices on the other side of the jth gap.*

We are now ready to prove the main theorem of this paper.

Theorem 2. *The cutwidth of a bipartite permutation graph can be computed in linear time.*

Proof. We describe the main algorithm for computing the cutwidth of a bipartite permutation graph G. First we compute a strong ordering of each connected component of G. Then we run MinCutBPG on each connected component with the computed strong ordering of that connected component. We concatenate the returned layouts from each of these calls into one layout φ for G. The order in which the layouts are concatenated does not matter, as the cuts at the concatenation points are empty. We check every position j with $1 \leq j < n$ to find a largest cut $\theta(j, \varphi, G)$, and we output $|\theta(j, \varphi, G)|$ as the cutwidth of G. If Algorithm MinCutBPG is correct then clearly the output of the described algorithm is equal to $\mathrm{cw}(G)$.

Before we prove the correctness of Algorithm MinCutBPG, let us analyze the running time of the above algorithm. By the results of [17], computing a strong ordering for each connected component of G takes in total $O(n + m)$ time. The running time of Algorithm MinCutBPG is also $O(n + m)$. To see this, observe that in the first loop, when deciding the box of a B-vertex, we never need to consider boxes to the left of the most recently considered box. By Observation 1, the next B-vertex is placed in either the box in which the previous B-vertex was placed, or a box further to the right. Thus running MinCutBPG on each connected component takes $O(n + m)$ time for the whole graph. Concatenating the returned layouts and finding the largest cut takes $O(n)$ time, and the overall running time follows.

Let us prove that Algorithm MinCutBPG correctly computes the cutwidth of a connected bipartite permutation graph. Assume for contradiction that there is a tight counterexample $G = (A, B, E)$. By Lemma 6, we know that there is a worst cut $\theta = \theta(j, \varphi, G)$ of the layout φ computed by Algorithm MinCutBPG on G, such that there are $\lfloor |A|/2 \rfloor$ A-vertices and $\lceil |B|/2 \rceil$ B-vertices on one side of the jth gap of φ, and $\lceil |A|/2 \rceil$ A-vertices and $\lfloor |B|/2 \rfloor$ B-vertices on the other side. Let $F = \{ab \notin E \mid a \in A \wedge b \in B\}$. Then $F \cap E = \emptyset$ and $(A, B, (E \cup F))$ is a complete bipartite graph. Since by Lemma 3 vertices on either side of θ induce a complete bipartite graph, we have that for each $ab \in F$, a and b are on different sides of θ. Thus we can conclude the following about the size of θ:

$$|\theta| = \left\lfloor \frac{|A|}{2} \right\rfloor \left\lfloor \frac{|B|}{2} \right\rfloor + \left\lceil \frac{|A|}{2} \right\rceil \left\lceil \frac{|B|}{2} \right\rceil - |F| \ .$$

Let S be *any* set of $\lfloor |A|/2 \rfloor + \lceil |B|/2 \rceil$ vertices of G. We claim that $|\theta(S, G)| \geq |\theta|$, regardless of how many A-vertices and how many B-vertices there are in S. To consider all possibilities, let there be $\lfloor |A|/2 \rfloor - x$ A-vertices and $\lceil |B|/2 \rceil + x$ B-vertices in S, for an appropriate (positive, zero or negative) integer x. Consequently, there are $\lceil |A|/2 \rceil + x$ A-vertices and $\lfloor |B|/2 \rfloor - x$ B-vertices in $(A \cup B) \setminus S$. Some of the set F of missing edges might have endpoints on different sides of the cut $\theta(S,G)$ and some might not. Since $(A, B, (E \cup F))$ is a complete bipartite graph, we know the following about the size of $\theta(S,G)$:

$$|\theta(S,G)| \geq \left(\left\lfloor \frac{|A|}{2} \right\rfloor - x\right)\left(\left\lfloor \frac{|B|}{2} \right\rfloor - x\right) + \left(\left\lceil \frac{|A|}{2} \right\rceil + x\right)\left(\left\lceil \frac{|B|}{2} \right\rceil + x\right) - |F|$$

$$= \left\lfloor \frac{|A|}{2} \right\rfloor \left\lfloor \frac{|B|}{2} \right\rfloor - \left\lfloor \frac{|A|}{2} \right\rfloor x - \left\lfloor \frac{|B|}{2} \right\rfloor x + x^2 + \left\lceil \frac{|A|}{2} \right\rceil \left\lceil \frac{|B|}{2} \right\rceil + \left\lceil \frac{|A|}{2} \right\rceil x + \left\lceil \frac{|B|}{2} \right\rceil x + x^2 - |F|$$

$$= |\theta| + 2x^2 + x\left(\left\lceil \frac{|A|}{2} \right\rceil - \left\lfloor \frac{|A|}{2} \right\rfloor + \left\lceil \frac{|B|}{2} \right\rceil - \left\lfloor \frac{|B|}{2} \right\rfloor\right).$$

Note that the value of the expression in parentheses in the last line of the equation is 0, 1, or 2. Consequently, for all possible values of x, we have that $|\theta(S,G)| \geq |\theta|$.

Let φ^* be an optimal layout of G, and let $j = \lfloor |A|/2 \rfloor + \lceil |B|/2 \rceil$. Let $S^* = L(j, \varphi^*, G)$. Hence S^* contains $\lfloor |A|/2 \rfloor + \lceil |B|/2 \rceil$ vertices and $\theta(j, \varphi^*, G) = \theta(S^*, G)$. Clearly $\mathrm{cw}(G) \geq |\theta(j, \varphi^*, G)| = |\theta(S^*, G)|$. However, for any such set S^*, we have shown above that a worst cut θ of the layout computed by Algorithm MinCutBPG has the property $|\theta(S^*, G)| \geq |\theta|$. Therefore, $\mathrm{cw}(G) \geq |\theta|$, contradicting the assumption that G is a counterexample. Consequently, no counterexample exists, and the algorithm correctly computes the cutwidth of every connected bipartite permutation graph.

4 Concluding Remarks

Algorithm MinCutBPG takes as input a connected bipartite permutation graph $G = (A, B, E)$ and a strong ordering (σ_A, σ_B) of G. Before the algorithm is called, $O(n + m)$ time is spent on recognizing G as a bipartite permutation graph and computing a strong ordering of G. Within the same running time one can assign two integers $\ell(v)$ and $r(v)$ to every vertex $v \in A \cup B$ for the following purpose. If $v \in A$ then $\ell(v)$ and $r(v)$ are the positions of the leftmost and the rightmost neighbor of v in σ_B. If $v \in B$ then $\ell(v)$ and $r(v)$ are the positions of the leftmost and the rightmost neighbor of v in σ_A. Observe that with this information, $d(v)$ can be computed in constant time, and the middle neighbor of a vertex can be found in constant time. Consequently, if $\ell(v)$ and $r(v)$ are supplied to MinCutBPG as input for every $v \in A \cup B$, the running time of MinCutBPG is in fact $O(n)$.

With our results in addition to the results of [10], the cutwidth of two unrelated subclasses of permutation graphs can be computed in linear time: threshold graphs and bipartite permutation graphs. We leave as an open problem to decide the computational complexity of computing the cutwidth of permutation graphs. In fact, it would be interesting to know the computational complexity of cutwidth on other well known subclasses of permutation graphs, like cographs or even their subclass trivially perfect graphs.

References

1. Adolphson, D., Hu, T.C.: Optimal linear ordering. SIAM J. Appl. Math. 25, 403–423 (1973)
2. Arora, S., Frieze, A., Kaplan, H.: A new rounding procedure for the assignment problem with applications to dense graphs arrangements. In: Proceedings of FOCS 1996, pp. 21–30. IEEE, Los Alamitos (1996)
3. Blin, G., Fertin, G., Hermelin, D., Vialette, S.: Fixed-parameter algorithms for protein similarity search under RNA structure constraints. In: Kratsch, D. (ed.) WG 2005. LNCS, vol. 3787, pp. 271–282. Springer, Heidelberg (2005)
4. Brandstädt, A., Le, V.B., Spinrad, J.: Graph Classes: A Survey. SIAM, Philadelphia (1999)
5. Brandstädt, A., Lozin, V.V.: On the linear structure and clique-width of bipartite permutation graphs. Ars Combinatorica 67, 273–289 (2003)
6. Chung, M.J., Makedon, F., Sudborough, I.H., Turner, J.: Polynomial time algorithms for the min cut problem on degree restricted d trees. In: Proceedings of FOCS 1982, pp. 262–271. IEEE, Los Alamitos (1982)
7. Díaz, J., Penrose, M., Petit, J., Serna, M.J.: Approximating layout problems on random geometric graphs. Journal of Algorithms 39, 78–117 (2001)
8. Díaz, J., Petit, J., Serna, M.J.: A survey of graph layout problems. ACM Computing Surveys 34, 313–356 (2002)
9. Gavril, F.: Some NP-complete problems on graphs. In: 11th Conference on Information Sciences and Systems, pp. 91–95. John Hopkins University, Baltimore (1977)
10. Heggernes, P., Lokshtanov, D., Mihai, R., Papadopoulos, C.: Cutwidth of split graphs, threshold graphs, and proper interval graphs. In: Broersma, H., Erlebach, T., Friedetzky, T., Paulusma, D. (eds.) WG 2008. LNCS, vol. 5344, pp. 218–229. Springer, Heidelberg (2008)
11. Jünger, M., Reinelt, G., Rinaldi, G.: The traveling salesman problem. In: Handbook on Operations Research and Management Sciences, vol. 7, pp. 225–330. North-Holland, Amsterdam (1995)
12. Karger, D.R.: A randomized fully polynomial approximation scheme for the all terminal network reliability problem. In: Proceedings of STOC 1996, pp. 11–17. ACM, New York (1996)
13. Leighton, F.T., Rao, S.: An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In: Proceedings of FOCS 1988, pp. 422–431. IEEE, Los Alamitos (1988)
14. Makedon, F., Sudborough, I.H.: Minimizing width in linear layouts. In: Díaz, J. (ed.) ICALP 1983. LNCS, vol. 154, pp. 478–490. Springer, Heidelberg (1983)
15. Monien, B., Sudborough, I.H.: Min cut is NP-complete for edge weighted trees. In: Kott, L. (ed.) ICALP 1986. LNCS, vol. 226, pp. 265–274. Springer, Heidelberg (1986)
16. Mutzel, P.: A polyhedral approach to planar augmentation and related problems. In: Spirakis, P.G. (ed.) ESA 1995. LNCS, vol. 979, pp. 497–507. Springer, Heidelberg (1995)
17. Spinrad, J., Brandstädt, A., Stewart, L.: Bipartite permutation graphs. Discrete Applied Mathematics 18, 279–292 (1987)
18. Thilikos, D.M., Serna, M.J., Bodlaender, H.L.: Cutwidth I: A linear time fixed parameter algorithm. Journal of Algorithms 56, 1–24 (2005)

19. Thilikos, D.M., Serna, M.J., Bodlaender, H.L.: Cutwidth II: Algorithms for partial w-trees of bounded degree. Journal of Algorithms 56, 24–49 (2005)
20. Yannakakis, M.: A polynomial algorithm for the min cut linear arrangement of trees. Journal of ACM 32, 950–988 (1985)
21. Yuan, J., Zhou, S.: Optimal labelling of unit interval graphs. Appl. Math. J. Chinese Univ. Ser. B (English edition) 10, 337–344 (1995)