Journal of Combinatorial Optimization
https://doi.org/10.1007/s10878-022-00897-4

Algorithms for maximum internal spanning tree problem for some graph classes

Gopika Sharma[1] · Arti Pandey[1] · Michael C. Wigal[2]

Accepted: 13 August 2022

Abstract

For a given graph G, a maximum internal spanning tree of G is a spanning tree of G with maximum number of internal vertices. The MAXIMUM INTERNAL SPANNING TREE (MIST) problem is to find a maximum internal spanning tree of the given graph. The MIST problem is a generalization of the Hamiltonian path problem. Since the Hamiltonian path problem is NP-hard, even for bipartite and chordal graphs, two important subclasses of graphs, the MIST problem also remains NP-hard for these graph classes. In this paper, we propose linear-time algorithms to compute a maximum internal spanning tree of cographs, block graphs, cactus graphs, chain graphs and bipartite permutation graphs. The optimal path cover problem, which asks to find a path cover of the given graph with maximum number of edges, is also a well studied problem. In this paper, we also study the relationship between the number of internal vertices in maximum internal spanning tree and number of edges in optimal path cover for the special graph classes mentioned above.

Keywords Maximum internal spanning tree · Bipartite graphs · Chordal graphs · Optimal path cover · NP-completeness · Graph Algorithms

---

Michael C. Wigal is supported by an NSF Graduate Research Fellowship under Grant No. DGE-1650044.

✉ Arti Pandey
arti@iitrpr.ac.in

Gopika Sharma
2017maz0007@iitrpr.ac.in

Michael C. Wigal
wigal@gatech.edu

1 Department of Mathematics, Indian Institute of Technology Ropar, Rupnagar, Punjab 140001, India

2 School of Mathematics, Georgia Institute of Technology, Atlanta, GA 30332, USA

Published online: 05 September 2022

1 Introduction

The Maximum Internal Spanning Tree (MIST) problem is a degree based spanning tree optimization problem, in which we ask to find a spanning tree of a given graph such that the number of vertices of degree at least two is maximized. The MIST problem is motivated by telecommunication network design (Salamon 2010). We also believe that MIST problem has its own theoretical importance as it is a generalization of the Hamiltonian path problem, a known NP-complete problem (Garey and Johnson 1979). The Hamiltonian path problem remains NP-complete for chordal graphs and chordal bipartite graphs (Lai and Wei 1993; Müller 1996). Hence, we also do not expect polynomial time algorithms for the MIST problem in chordal graphs and chordal bipartite graphs.

The dual problem to MIST, the Minimum Leaves Spanning Tree (MLST) problem asks to find a spanning tree with minimum number of leaves for a given graph. The MLST problem cannot be approximated within any constant factor unless P=NP (Lu and Ravi 1992). Unlike MLST, several constant factor approximation algorithms have been proposed for the MIST problem in literature. In 2003, Prieto et al. (Prieto and Sloper 2003) gave a 2-approximation algorithm for the MIST problem whose running time was later improved by Salamon et al. in 2008 (Salamon and Wiener 2008). Salamon also gave approximation algorithms for claw-free and cubic graphs with approximation factors $\frac{3}{2}$ and $\frac{6}{5}$ respectively (Salamon and Wiener 2008). In 2009, Salamon (Salamon 2009) gave a $\frac{7}{4}$ -approximation algorithm for graphs with no pendant vertices and later, in 2015, Knauer et al. (Knauer and Spoerhase 2015) showed that a simplified and faster version of Salamon's algorithm yields a $\frac{5}{3}$ -approximation algorithm even on general graphs. In 2014, Li et al. proposed a $\frac{3}{2}$-approximation algorithm using a different approach for general undirected graphs and improved this ratio to $\frac{4}{3}$ for graphs without leaves (Li and Zhu 2014). Li et al. gave a $\frac{3}{2}$ -approximation algorithm for general graphs using depth-5 local search (Li et al 2017). In 2018, Chen et al. presented a $\frac{17}{13}$ -approximation algorithm which is the best approximation factor till now (Chen et al 2018). Recently, Li et al. proved that the MIST problem is Max-SNP-hard (Li et al 2021). Several FPT-algorithms have also been designed for the MIST problem where the considered parameter is the solution size (Prieto and Sloper 2003; Li et al 2017; Cohen et al 2010; Fomin et al 2013; Binkele-Raible et al 2013; Li et al 2015).

For finding efficient algorithms for the MIST problem, it is often useful to reduce the MIST problem to the path cover problem. A *path cover* P of a graph is a spanning subgraph such that every component of P is a path. A path cover with maximum number of edges is called an *optimal path cover* of G. If P^* denotes an optimal path cover of a graph, then number of edges in P^* is denoted by $|E(P^*)|$. In 2018, Li et al. proposed a polynomial time algorithm for the MIST problem in interval graphs (Li et al 2018). They also proved that number of internal vertices in any MIST of any graph G is at most $|E(P^*)| - 1$, where P^* is an optimal path cover of G. We will observe that number of internal vertices in any MIST of a chain graph is either $|E(P^*)| - 1$ or $|E(P^*)| - 2$ and is $|E(P^*)| - 1$ for cographs. For bipartite permutation, block and cactus graphs, we prove that there is no constant k such that $|E(P^*)| - k$ is the lower

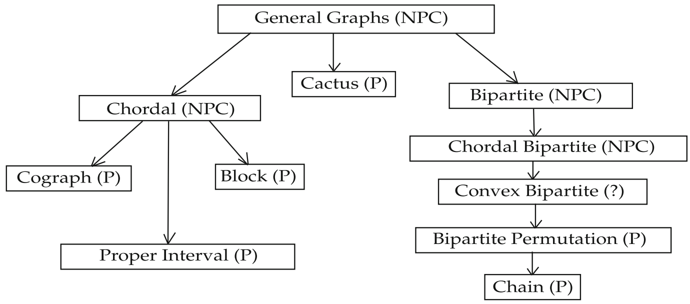


Fig. 1 Hierarchy relationship between some classes of graphs

bound on the number of internal vertices in any MIST of such graphs. We also propose linear-time algorithms for the MIST problem in bipartite permutation graphs, block graphs, cactus graphs and cographs. A hierarchy relationship between these classes of graphs is shown in Fig. 1.

The structure of the paper is as follows. In Section 2, we give some basic definitions and notations used in the paper. In Section 3, we discuss MIST problem for block graphs and cactus graphs and provide linear-time algorithms for both these graph classes. In Section 4, we prove that MIST of cographs can be computed in linear-time by providing an algorithm. In Section 5, we present a linear-time algorithm to find a MIST of an arbitrary bipartite permutation graph. In Section 6, we prove a bound for chain graphs regarding number of internal vertices in its MIST. For all the graph classes mentioned above, we discuss the relationship between number of internal vertices in any MIST of a graph G and number of edges in an optimal path cover of G in Section 7. Finally, Section 8 concludes the paper.

2 Preliminaries

Let $G = (V, E)$ be a graph. In this paper, we only consider simple, undirected and connected graphs. For a vertex $u \in G$, $d_G(u)$ denotes the degree of u in G and $N_G(u)$ denotes the neighborhood of u in G. When there is no ambiguity regarding the graph G, we simply use $d(u)$ and $N(u)$, to represent the degree of u and neighborhood of u, respectively. A vertex u in V is called *pendant* if $d(u) = 1$. The set of pendant vertices in G is denoted by $P(G)$. The vertex adjacent to a pendant vertex u is called the *support vertex* of u, and is denoted by $S(u)$. A vertex $u \in V(G)$ is called *internal*, if u is not pendant, that is, $d(u) \geq 2$. Let $I(G)$ denotes the set of internal vertices in G, and $i(G) = |I(G)|$. For a set $A \subseteq V$ and a spanning tree T of G, we define $i_T(A) = |I(T) \cap A|$. Hence, for an empty set A, $i_T(A) = 0$.

For vertices x and y, we denote an edge between x and y by xy. For a subset S of $V(G)$, $G - S$ denotes the subgraph of G obtained after removing vertices of S and edges incident on these vertices from G. If $S = \{v\}$, then we simply write $G - v$ for $G - S$. A vertex v of a graph G is called a *cut vertex* if $G - v$ is disconnected.

Throughout this paper, n denotes the number of vertices and m denotes the number of edges in G. A graph G is said to be *bipartite* if V can be partitioned into two disjoint sets X and Y such that every edge of G joins a vertex in X to a vertex in Y. Such a partition (X, Y) of V is called a *bipartition*. A bipartite graph with bipartition (X, Y) of V is denoted by $G = (X, Y, E)$. For a set $S \subseteq V$, an *induced subgraph* is the graph whose vertex set is S and edge set consists of all the edges in E that have both the endpoints in S, and is denoted by $G[S]$. If $G[C]$, $C \subseteq V$, is a complete subgraph of G, then C is called a *clique* of G.

A subgraph of G is called a *spanning subgraph* if it contains all the vertices of G. A *path cover* P of a graph is a spanning subgraph such that every component of P is a path. A path cover is an *optimal path cover* if it has the maximum number of edges. A spanning subgraph of G which is also a tree is called a *spanning tree* of G. A spanning tree is called a *maximum internal spanning tree(MIST)* if it contains the maximum number of internal vertices among all the spanning trees of G. For a graph G, the number of internal vertices in any MIST of G is denoted by $Opt(G)$.

Now we state a useful theorem which gives an upper bound on the number of internal vertices in a MIST with respect to the graph's optimal path cover.

Theorem 1 (Li et al 2018) *For a graph G, the number of internal vertices in a maximum internal spanning tree of G is less than the number of edges in an optimal path cover of G, that is,* $Opt(G) \leq |E(P^*)| - 1$, *where* P^* *denotes an optimal path cover of G. Moreover, this bound is tight.*

Note that the vertices which are pendant in G itself, will be pendant in any MIST of G. Hence, we have the following lemma.

Lemma 1 *For a graph G, if v is a pendant vertex and u is the support vertex of v in G, then v remains a pendant vertex and u remains adjacent support vertex of v in any MIST of G.*

Suppose G is not a tree and $u \in V(G)$ is adjacent to k pendant vertices, say $a_1, \ldots, a_k$. Let $G' = G - \{a_2, \ldots, a_k\}$. Then based on Lemma 1, the number of internal vertices in a MIST of G will be same as the number of internal vertices in any MIST of G'. It is also easy to obtain a MIST of G from any MIST of G'. Hence, throughout this work, we assume that every vertex has at most one pendant vertex adjacent with it.

Below, we give another result regarding the number of pendant vertices in a spanning tree of a bipartite graph. Note that, if we have α number of internal vertices in a spanning tree of G from one partite set, then at least $\alpha + 1$ vertices must be present in the neighborhood of these α vertices, which lie in the other partite set of the bipartite graph G.

Lemma 2 *Let* $G = (X, Y, E)$ *be a bipartite graph with* $A \subseteq X$ *and* $B \subseteq Y$. *If* $N(A) = B$, *then there are at least* $\max\{0, |A|-|B|+1\}$ *pendant vertices from A in any spanning tree of G. Similarly, if* $N(B) = A$, *then there are at least* $\max\{0, |B|-|A|+1\}$ *pendant vertices from B in any spanning tree of G.*

3 Block and cactus graphs

In this section, we discuss the MIST problem for block graphs and cactus graphs. We will show that the MIST problem can be solved in linear-time for both classes of graphs. Block and cactus graphs will also provide our first family of examples in which $Opt(G)$ cannot be lower bounded in terms of $|E(P^*)| - k$ where P^* is an optimal path cover of G and k is some constant.

A *block* of a graph G is a maximal connected subgraph with no cut vertices. Note that a block of G is either an edge or a 2-connected subgraph. The set of blocks of a graph is called the *block decomposition* of G and is denoted by $B(G)$. Let $B_0 \in B(G)$ and u, v be two vertices belonging to B_0, then a path between u and v, which contains all the vertices of the block B_0, is called a *spanning path between u and v in B_0*. We say a block B is *good* if there exists distinct $u, v \in V(B)$ such that both u and v are cut vertices of G and B has a spanning path between u and v. A block is said to be *bad* otherwise. Let $\mathrm{Bad}(G)$ denote the set of bad blocks of G.

A *block graph* is a graph in which every block is a clique. If a block graph G contains only one block then G is a complete graph. A block graph is said to be *nontrivial* if it contains at least two blocks. Note that a trivial block has a Hamiltonian path. Thus for the remainder of the section we only consider nontrivial block graphs.

Let G be a nontrivial block graph. Bad blocks of G have another characterization which we state as the Proposition 1.

Proposition 1 *A block B of a nontrivial block graph G is bad if and only if it contains exactly one cut vertex of G.*

Proof A block containing exactly one cut vertex of G can not be a good block. So, assume B is a bad block of G and it contains at least 2 cut vertices of G. Then, these 2 vertices has a spanning path between them in B as $G[V(B)]$ is a clique. This implies that B is a good block, a contradiction. Hence, B contains exactly one cut vertex of G. □

A graph G is a *cactus graph* if every block of G is either a cycle or an edge. If a cactus graph G contains only one block then G is either a cycle or an edge and in that case finding a MIST of G is trivial. Again, a cactus graph is said to be *nontrivial* if it contains at least two blocks and now we only consider nontrivial cactus graphs.

Let G be a nontrivial cactus graph. A block of G is called an *end block* of G if it contains exactly one cut vertex of G. Note that an end block of a cactus graph G is also a bad block of G. Bad blocks of a cactus graph G have another characterization which we state in the following Proposition.

Proposition 2 *A block B of a nontrivial cactus graph G is bad if and only if B does not contain two adjacent cut vertices of G.*

Proof First, let B be a bad block of G. If B is an end block then it does not have two distinct cut vertices of G and so, there is nothing to prove. If B is not an end block, then it contains at least 2 cut vertices of G. Since B is a bad block and two adjacent vertices of a block of a cactus graph have a spanning path between them so, B does not contain two adjacent cut vertices of G. Conversely, let B be a block such that it does not contain two adjacent cut vertices of G. Then no two cut vertices of G has a spanning path between them in B. Hence, B is a bad block of G. $\square$

If B_i and B_j are two blocks of a block/cactus graph G and $V(B_i) \cap V(B_j) \neq \emptyset$, then $|V(B_i) \cap V(B_j)| = 1$ and the vertex $x \in V(B_i) \cap V(B_j)$ is a cut vertex of G. Below, we state two propositions which hold true for both block and cactus graphs.

Proposition 3 *Let T be a MIST of a nontrivial block/cactus graph G. Then, T must have at least one leaf in every bad block of G.*

Proof Let B be a bad block of G. If B is an edge, then one vertex of B is itself pendant in G. So, we may assume that B is not an edge. Now, suppose that every vertex of block B is internal in T then the degree of each vertex of B is at least 2 in T.

First, let G be a block graph. By Proposition 1, B has exactly one cut vertex of G, say u. Let $T' = T[V(B)]$. As T' is a forest, it must contain at least two leafs. As for any $x \in V(B) \setminus \{u\}$, $d_T(x) = d_{T'}(x)$, B must contain at least one leaf of T.

Now, let G be a cactus graph and $u \in V(B)$ be a cut vertex of G. Let x and y be neighbors of u in B. Since B is a bad block, by Proposition 2, x and y are non-cut vertices in G which implies that their degree is exactly 2 in G and edges xu, uy belong to T. Now, let $v \in V(B)$ be any non-cut vertex of G and let x' and y' be neighbors of v. Since $d_G(v)$ is 2 and it is an internal vertex in T, edges $x'v$, vy' belong to T. So, we see that every edge of the block B of G belongs to T, a contradiction.

Hence, T must have at least one leaf in every bad block. $\square$

Proposition 4 *Let T be a MIST of a nontrivial block/cactus graph G. Then, $Opt(G) \leq n - |Bad(G)|$, where $Opt(G)$ denotes the number of internal vertices in T.*

Proof By Proposition 3, we have that $|P(T)| \geq |\text{Bad}(G)|$, where $P(T)$ denotes the set of leaves of T. So,

$$\begin{aligned} Opt(G) &= \text{number of internal vertices in a MIST of } G \\ &= n - \text{number of pendant vertices in a MIST of } G \\ &= n - |P(T)| \\ &\leq n - |\text{Bad}(G)| \end{aligned}$$ $\square$

Recall that block decomposition of a graph G is the set of blocks of G. It can be computed in $O(n)$ time using the following approach. Let b be a cut vertex of a block/cactus graph G and $G_1, G_2, \ldots, G_t$ be the connected components of the graph $G - b$. Let H_i denotes the subgraph $G[V(G_i) \cup \{b\}]$, for each $1 \leq i \leq t$. We call $H_1, H_2, \ldots, H_t$ the b-components of G. The block decomposition of a block/cactus graph can be found by recursively choosing a cut vertex b and computing the b-components.

3.1 Algorithm for block and cactus graphs

In this subsection, we first prove a theorem which relates the number of internal vertices in a MIST of a block/cactus graph G to the number of bad components of G. Then, we outline a linear-time algorithm to compute a MIST of G.

Theorem 2 *Let G be a graph with a nontrivial block decomposition such that each block has a spanning path with a cut vertex as an endpoint. Then G has a spanning tree T in which number of internal vertices is $n - |Bad(G)|$.*

Proof Let l be the number of blocks in G and $B_i \in B(G)$ be an arbitary block of G. If B_i is good, then let P_i be a spanning path between two cut vertices of B_i. If B_i is bad, we let P_i be a spanning path with a single cut vertex as an endpoint. Let $T = \bigcup_{i=1}^{l} P_i$. Note that T is a spanning tree of G. Furthermore, as any cut vertex of G cannot be a leaf of T, we have $i(T) = n - |Bad(G)|$. □

The proof of Theorem 2 gives a simple algorithm for a block or cactus graph. First find a block decomposition, this takes $O(n)$ time. Then for each block B, determine if B is bad or not and find the corresponding path. This takes $O(|B|)$ time. In total we have a linear-time algorithm. As both block and cactus graphs satisfy the hypothesis of Theorem 2, combining with Proposition 4 we have the following,

Corollary 1 *If G is a block or cactus graph, then $Opt(G) = n - |Bad(G)|$.*

4 Cographs

In this section, we discuss the MIST problem for cographs. The complement-reducible graphs or *cographs* have been discovered independently by several authors since the 1970s (Seinsche 1974; Jung 1978). In the literature, the cographs are also known as P_4-free graphs, D^*-graphs, Hereditary Dacey graphs and 2-parity graphs. The class of cographs is defined recursively as follows:

- A graph containing a single vertex is a cograph;
- Complement of a graph is also a cograph;
- Disjoint union of two cographs is also a cograph.

Cographs admit a rooted tree representation (Lerchs 1972). This tree is called a cotree of a cograph G, denoted $T(G)$. The set of leaves of the left subtree (right subtree) of an interior vertex x of $T(G)$ is denoted by $L(x_{left})$ $(L(x_{right}))$. Lin et al (1995), gave an algorithm to preprocess a cotree $T(G)$ with root r such that it is a binary rooted tree possessing the following properties:

1. Every internal vertex has exactly two children.
2. Every internal vertex is labeled 0 or 1 with the root r receiving label 1, such that no two adjacent internal vertices receive the same label.
3. The leaves of $T(G)$ form a bijective correspondence with $V(G)$, such that $x, y \in V(G)$ are adjacent if and only if their lower common ancestor of x and y in $T(G)$ has label 1.

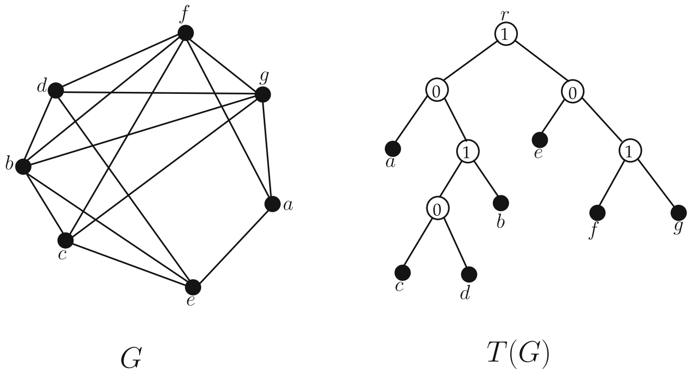


Fig. 2 Illustrating a cograph and its cotree

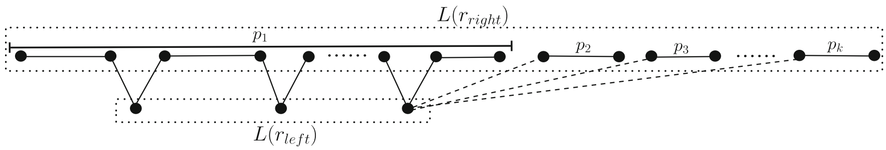


Fig. 3 Optimal path cover of G contains more than 1 path components

4. For all interior vertices x of $T(G)$ assigned label 1, $|L(x_{left})| \geq |L(x_{right})|$.

Fig. 2 illustrates a cograph G along with its cotree $T(G)$.

Proposition 5 *For any $x \in L(r_{left})$, $y \in L(r_{right})$, we have $xy \in E(G)$.*

Proof A leaf of $T(G)$ represents a vertex of the graph G. Let $x \in L(r_{left})$, $y \in L(r_{right})$. As the least common ancestor of x and y is r and r has label 1, by property 3 of a cotree, $xy \in E(G)$. □

Recall that a path cover P of a graph G is a spanning subgraph such that every component of P is a path. A path cover is an optimal path cover if it has the maximum number of edges. Lin et al (1995) gave a linear-time algorithm to compute an optimal path cover of a cograph G. The optimal path cover P^* constructed in (Lin et al 1995) is one of the following type:

- The path cover P^* contains a single path component which is a Hamiltonian path of G.
- The path cover P^* contains at least two path components. In this case, there exists exactly one path p in P^* which contains vertices from both the sets $L(r_{left})$ and $L(r_{right})$ and all other paths in P^* contain vertices from $L(r_{right})$ only. Fig. 3 illustrates this case.

Algorithm 1 uses the optimal path cover constructed from (Lin et al 1995) to compute a MIST of a cograph G.

Note that by Theorem 1 we have $Opt(G) \leq |E(P^*)| - 1$ for an optimal path cover P^*. Below, we give a theorem which implies that Algorithm 1 also outputs a spanning tree which attains this upper bound.

Algorithm 1 Algorithm for finding a MIST of a cograph G

Input: A cograph G and a cotree $T(G)$ of G
Output: A Maximum Internal Spanning Tree T of G
1 Let $P^* = \{P_1, P_2, \ldots, P_k\}$ be the optimal path cover of G computed by the algorithm in (Lin et al 1995);
2 $V(T) = V(G)$ and $E(T) = E(P^*)$;
3 **if** $k = 1$ **then**
4   return T;
5 **else**
    /* P_1 is the path which contains vertices from both the sets $L(r_{left})$ and $L(r_{right})$ and all other paths in P^* contain vertices from $L(r_{right})$ only */
6   Let $u \in (V(P_1) \cap L(r_{left}))$;
7   Let v_i be an end vertex of the path P_i, for $2 \leq i \leq k$;
8     $E(T) = E(T) \cup \{uv_2, uv_3, \ldots, uv_k\}$;
9   return T.

Theorem 3 *Algorithm 1 outputs a spanning tree T of a cograph G such that, $i(T) = |E(P^*)| - 1$, where P^* is an optimal path cover of G. Hence, $Opt(G) = |E(P^*)| - 1$.*

Proof Let $P^* = \{P_1, P_2, \ldots, P_k\}$ be the optimal path cover computed in step 1 of Algorithm 1. If $|P^*| = 1$, then G has a Hamiltonian path and Algorithm 1 returns a Hamiltonian path. Now, suppose $|P^*| > 1$, then the path P_1 contains vertices from both sets $L(r_{left})$ and $L(r_{right})$ and $P_i \cap L(r_{left}) = \emptyset$ for all $i \geq 2$. Now, let $u \in V(P_1) \cap L(r_{left})$ such that u is not an end vertex of P_1.

For each path in $P_i \in P^* \setminus \{P_1\}$, consider a pendent vertex v_i of the path. By Proposition 5, v_i and u are adjacent. Let $T = \bigcup_{i=1}^{k} P_i \cup \{v_i u : 2 \leq i \leq k\}$. These new edges connect one internal vertex with a pendant vertex of path of P^*. This is illustrated by the dash edges in Fig. 3. Note then the number of internal vertices of T is $|E(P^*)| - 1$, hence $i(T) = Opt(G) = |E(P^*)| - 1$ by Theorem 1. □

Note that step 1 of Algorithm 1 can be performed in linear-time (Lin et al 1995). Furthermore, note that the construction of T in Theorem 3 is also linear-time. Therefore Algorithm 1 outputs a MIST of G in linear-time.

5 Bipartite permutation graphs

In this section, we discuss the MIST problem for bipartite permutation graphs. A graph $G = (V, E)$ with $V = \{v_1, v_2, \ldots, v_n\}$ is said to be a *permutation graph* if there is a permutation σ over $\{1, 2, \ldots, n\}$ such that $v_i v_j \in E$ if and only if $(i - j)(\sigma^{-1}(i) - \sigma^{-1}(j)) < 0$. Intuitively, each vertex v in a permutation graph corresponds to a line segment l_v joining two points on two parallel lines L_1 and L_2, which is called a line representation. Then, two vertices v and u are adjacent if and only if the corresponding line segments l_v and l_u are crossing. Vertex indices give the ordering of the points on L_1, and the permutation of the indices gives the ordering of the points on L_2. Fig. 4 shows an example of a permutation graph along with its line representation. When a permutation graph is bipartite, it is said to be a *bipartite permutation graph*.

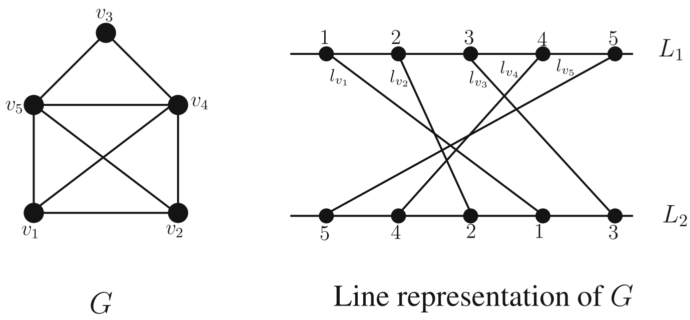


Fig. 4 A permutation graph G on 5 vertices with permutation $(\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5) = (5, 4, 2, 1, 3)$

A strong ordering $(<_X, <_Y)$ of a bipartite graph $G = (X, Y, E)$ consists of an ordering $<_X$ of X and an ordering $<_Y$ of Y, such that for all edges $ab, a'b'$ with $a, a' \in X$ and $b, b' \in Y$: if $a <_X a'$ and $b' <_Y b$, then ab' and $a'b$ are edges in G. An ordering $<_X$ of X has the *adjacency property* if, for every vertex in Y, its neighbors in X are consecutive in $<_X$. The ordering $<_X$ has the *enclosure property* if, for every pair of vertices y, y' of Y with $N(y) \subseteq N(y')$, the vertices of $N(y') \setminus N(y)$ appear consecutively in $<_X$. These properties are useful for characterizing bipartite permutation graphs.

Heggernes et al (2012) proved that *a bipartite graph is a bipartite permutation graph if and only if it admits a strong ordering*. Furthermore if we assume that the graph is connected, then we can say more.

Lemma 3 (Heggernes et al 2012) *Let $(<_X, <_Y)$ be a strong ordering of a connected bipartite permutation graph $G = (X, Y, E)$. Then both $<_X$ and $<_Y$ have the adjacency property and the enclosure property.*

Throughout this section, $G = (X, Y, E)$ denotes a connected bipartite permutation graph. A strong ordering of a bipartite permutation graph can be computed in linear-time (Spinrad et al 1987). Let $(<_X, <_Y)$ be a strong ordering of G, where $<_X = (x_1, x_2, \ldots, x_{n_1})$ and $<_Y = (y_1, y_2, \ldots, y_{n_2})$. We write strong ordering of vertices of G as $(<_X, <_Y) = (x_1, x_2, \ldots, x_{n_1}, y_1, y_2, \ldots, y_{n_2})$. For $u, v \in V(G)$, we write $u <_X v$ if $u, v \in X$ and u appears before v in the strong ordering; we define $u <_Y v$ in a similar manner. We write $x_i < x_j$ (or, $y_i < y_j$) when $i < j$. For vertices u, v of G, $u \leq v$ denotes either $u <_X v$, $u <_Y v$, or $u = v$ holds.

Since each vertex of G satisfies the adjacency property, the neighborhood of any vertex consists of consecutive vertices in the strong ordering. We define the first neighbor of a vertex as the vertex with minimum index in its neighborhood and the last neighbor of a vertex as the vertex with maximum index in its neighborhood. We notate the first and last neighbors of a vertex u as $f(u)$ and $l(u)$ respectively. Combining the above statements for a bipartite permutation graph G with its strong ordering $(<_X, <_Y)$, G has the following properties (Lai and Wei 1997):

1. For any vertex of G, vertices in its neighborhood are consecutive with respect to the ordering $<_X$ or $<_Y$.
2. If $u < v$ then $f(u) \leq f(v)$ and $l(u) \leq l(v)$, for each pair of vertices $u, v \in V(G)$.

Now, we define some terminology which we require for the remainder of this section. A vertex $x_i \in X, (1 \leq i \leq n_1)$ with $l(x_i) = y_j$ is of *type 1* if $j \geq i$. A vertex $y_i \in Y, (1 \leq i \leq n_2)$ with $l(y_i) = x_j$ is of *type 1* if $j \geq i+1$. Similarly, a vertex $x_i \in X, (1 \leq i \leq n_1)$ with $l(x_i) = y_j$ is of *type 2* if $j \geq i+1$ and a vertex $y_i \in Y, (1 \leq i \leq n_2)$ with $l(y_i) = x_j$ is of *type 2* if $j \geq i$. Note that a type 2 vertex $x \in X$ is also a type 1 vertex but the converse may not be true. Furthermore, a type 1 vertex $y \in Y$ is also a type 2 vertex. Characterizing the vertices in this way is an important distinction for our algorithm. We now prove two important lemmas which will be used to prove the correctness of Algorithm 2.

Lemma 4 *Let $X' = \{x_1, x_2, \ldots, x_k, x_{k+1}\} \subseteq X$, $Y' = \{y_1, y_2, \ldots, y_k\} \subseteq Y$. Furthermore, suppose each vertex of X' and Y' is of type 1 except x_{k+1}, $l(x_{k+1}) = y_k$ and $N(X') = Y'$. Then there exists a MIST T of G, in which x_1 and x_{k+1} are pendant. Moreover; if $X \setminus X' \neq \emptyset$, then the support vertex of x_{k+1} is of degree at least 3 in T.*

Proof We first show $x_i y_i, y_i x_{i+1} \in E(G)$ for all $1 \leq i \leq k$. Suppose there exists $1 \leq i \leq k$ such that $x_i y_i \notin E(G)$. Let $l(x_i) = y_j$ and $l(y_i) = x_l$. As both x_i and y_i are type 1, we have $y_j \geq y_i$ and $x_l > x_i$. As $(<_X, <_Y)$ is a strong ordering, we have $x_i y_i \in E(G)$, a contradiction. Thus we may assume $x_i y_i \in E(G)$. Furthermore as y_i is type 1, we have $x_i y_i, y_i x_{i+1} \in E(G)$ for all $1 \leq i \leq k$.

Suppose $X = X'$. Note that as $(<_X, <_Y)$ is a strong ordering of G, we have for all $x \in X$, $l(x) \leq l(x_{k+1}) = y_k$. As we assumed G is connected, we have that $Y = Y'$ as well. Note that this implies that G has the Hamiltonian path $x_1 y_1 x_2 \ldots x_k y_k x_{k+1}$ which is a MIST. So, we may assume that $X \setminus X' \neq \emptyset$.

Now, let T^* be a MIST of G. If x_1 and x_{k+1} are pendant in T^* and degree of $S(x_{k+1})$ is at least 3 in T^*, then we are done. Suppose otherwise, and we modify T^* in the following way. We first remove all edges of T^* incident with the vertices of X' and then add edges $x_1 y_1, y_1 x_2, x_2 y_2, \ldots, x_k y_k$ and $y_k x_{k+1}$ to obtain a new graph T. Note that as $N(X') = Y'$, T is connected.

First suppose T contains no cycle. Note that T is a spanning tree of G. If $d_T(y_k) = 2$, then as $N(X') = Y'$ we can choose an edge $v y_i (i < k)$ in T such that $v \in X \setminus X'$. Since the strong ordering $(<_X, <_Y)$ of the vertices of G satisfies property 2, we have $v y_k \in E(G)$. So we can further modify T by removing the edge $v y_i$ and replacing with the edge $v y_k$. We see that

$$\begin{aligned} i(T^*) &= i_{T^*}(X') + i_{T^*}(X \setminus X') + i_{T^*}(Y') + i_{T^*}(Y \setminus Y') \\ &\leq (k-1) + i_{T^*}(X \setminus X') + k + i_{T^*}(Y \setminus Y') = i(T). \end{aligned}$$

So, we have $i(T^*) \leq i(T)$. Since T is a spanning tree and T^* is a MIST of G, we have that T is also a MIST. Thus, we obtain our desired MIST in which x_1 and x_{k+1} are pendant and the support vertex of x_{k+1} is of degree at least 3.

Now, suppose T contains a cycle C. This implies that there exists $v \in X \setminus X'$ such that $v y_i, v y_j \in E(C)$ with $i < j \leq k$. Now, we modify T by removing the edge $v y_i$. This step reduces degree of v by 1 while leaving the graph T connected. We repeat this modification until T has no more cycles, thus T will be a spanning tree of G. Let us assume that there are α such vertices which became pendant in this process of updating

T. Let $A \subseteq X \setminus X'$ be the set of α vertices. Note these α vertices were internal in T^*. Suppose $i_{T^*}(X') > k - (\alpha+1)$. As $N(X') = Y'$, then the subforest of T^* induced by the set $X' \cup Y' \cup A$ would have at least $2(k-\alpha) + (1+\alpha) + 2\alpha = 2k+1+\alpha$ edges. As $2k+1+\alpha > |X' \cup Y' \cup A| - 1$, this contradicts the fact that T^* was a tree. Thus we have $i_{T^*}(X') \leq k - (\alpha + 1)$. It follows,

$$\begin{aligned} i(T^*) &= i_{T^*}(X') + i_{T^*}(X \setminus X') + i_{T^*}(Y') + i_{T^*}(Y \setminus Y') \\ &\leq (k - (\alpha+1)) + i_{T^*}(X \setminus X') + k + i_{T^*}(Y \setminus Y') \\ &= (k-1) + (i_{T^*}(X \setminus X') - \alpha) + k + i_{T^*}(Y \setminus Y') = i(T). \end{aligned}$$

Again, we have $i(T^*) \leq i(T)$ which implies that T is also a MIST. If $d_T(y_k) = 2$, then we can choose an edge $vy_i (i < k)$ in T, such that $v \in X \setminus X'$. Since the strong ordering $(<_X, <_Y)$ satisfies property 2, we have $vy_k \in E(G)$. So we update the tree T by removing the edge vy_i and adding the edge vy_k. Thus, we obtain a MIST T in which x_1 and x_{k+1} are pendant and support vertex of x_{k+1} is of degree at least 3. □

Lemma 5 *Let $X' = \{x_1, x_2, \ldots, x_k\} \subseteq X$, $Y' = \{y_1, y_2, \ldots, y_k\} \subseteq Y$. Furthermore, suppose each vertex of X' and Y' is of type 1 except y_k, $l(y_k) = x_k$ and $N(Y') = X'$.*

(a) If $x_i y_{i+1} \in E(G)$ for all $1 \leq i \leq (k-1)$, then there exists a MIST T of G, in which y_1 is pendant.
(b) If there exists $1 \leq t \leq (k-1)$ such that $x_t y_{t+1} \notin E(G)$, then there exists a MIST T of G, in which x_1 and y_k are pendant. Moreover; if $Y \setminus Y' \neq \emptyset$, then support vertex of y_k is of degree at least 3 in T.

Proof We first argue that $x_i y_i, y_i x_{i+1} \in E(G)$ for $1 \leq i \leq k-1$. First assume for some i, $x_i y_i \notin E(G)$. As both x_i and y_i are type 1, we have $x_i < l(y_i)$ and $y_i < l(x_i)$. As $(<_X, <_Y)$ is a strong ordering, we have $x_i y_i \in E(G)$, a contradiction. Furthermore, as y_i is type 1, we have $y_i x_{i+1} \in E(G)$ for all $1 \leq i \leq k-1$.

Suppose $Y' = Y$. As $N(Y') = X'$, and G is connected we have that $X = X'$ as well. Note then if $x_i y_{i+1} \in E(G)$ for all $1 \leq i \leq (k-1)$, then $y_1 x_1 \ldots x_{k-1} y_k x_k$ is a Hamiltonian path. Otherwise, if there exists $1 \leq t \leq (k-1)$ such that $x_t y_{t+1} \notin E(G)$, then the path $x_1 y_1, y_1 x_2, x_2 y_3, \ldots, y_{k-1} x_k$ and $x_k y_k$ gives the desired Hamiltonian path.

So, we may assume that $Y \setminus Y' \neq \emptyset$ and we will first prove part (a). Let T^* be a MIST of G and suppose y_1 is not pendant in T^*. Let T be the graph obtained from T^* where we remove all edges incident to the vertices of Y' and add edges $y_1 x_1, x_1 y_2, y_2 x_2, \ldots, x_{k-1} y_k$ and $y_k x_k$. Note as $N(Y') = X'$, y_1 is pendant in T.

First, suppose T contains no cycles. Note then that T is a spanning tree of G. We argue that we may assume $d_T(x_k) \geq 2$. Suppose otherwise, that is, $d_T(x_k) = 1$. Let $v \in Y \setminus Y'$ such that $vx_i (i < k)$. As the strong ordering of the vertices of G satsifies property 2, we have $vx_k \in E(G)$ as well. So we further modify T by removing the edge vx_i and adding the edge vx_k. We see that

$$\begin{aligned} i(T^*) &= i_{T^*}(X') + i_{T^*}(X \setminus X') + i_{T^*}(Y') + i_{T^*}(Y \setminus Y') \\ &\leq k + i_{T^*}(X \setminus X') + (k-1) + i_{T^*}(Y \setminus Y') = i(T). \end{aligned}$$

So, we have that $i(T^*) \leq i(T)$. Since T is a spanning tree and T^* is a MIST of G, T is also a MIST of G.

Next, suppose T is not a tree. We now modify T to remove the cycles. Let C be a cycle of T. Note then there is a vertex $v \in Y \setminus Y'$ such that $vx_i, vx_j \in E(C)$ with $i < j \leq k$. We then modify T by removing the edge vx_i. Note that the degree of v decreases by 1. We repeat this process until no cycles remain in T. Assume that α cycles were removed during this process and thus at most α pendant vertices were created in this process. As $N(Y') = X'$ and T^* is a tree, we have $i_{T^*}(Y') \leq k - \alpha - 1$. We see that,

$$\begin{aligned} i(T^*) &= i_{T^*}(X') + i_{T^*}(X \setminus X') + i_{T^*}(Y') + i_{T^*}(Y \setminus Y') \\ &\leq k + i_{T^*}(X \setminus X') + (k - \alpha - 1) + i_{T^*}(Y \setminus Y') \\ &\leq k + i_{T^*}(X \setminus X') + (k - 1) + (i_{T^*}(Y \setminus Y') - \alpha) = i(T) \end{aligned}$$

Again, we have that $i(T^*) \leq i(T)$ which implies that T is also a MIST. Hence, part (a) holds.

Next, we prove part (b). Let T^* be a MIST of G. If x_1 and y_k are pendant in T^* and degree of $S(y_k)$ is at least 3 in T^*, then we are done, so assume otherwise. Let T be the graph obtained from modifying T^* where we remove all edges incident on the vertices of Y' and add edges $x_1y_1, y_1x_2, x_2y_2, \ldots, y_{k-1}x_k$ and x_ky_k.

First suppose T contains no cycle, then T is a spanning tree of G. If $d_T(x_k) \geq 3$, then we are done modifying, so suppose $d_T(x_k) = 2$. As $Y \setminus Y' \neq \emptyset$ and $N(Y') = X'$, there exists an edge $vx_i (i < k)$ in T. Since the strong ordering of the vertices of G satisfies property 2, we have $vx_k \in E(G)$. Thus we further modify T where we remove vx_i and add the edge vx_k. As we assumed there exists a $1 \leq t \leq (k-1)$ such that $x_ty_{t+1} \notin E(G)$, we have $N(\{x_1, x_2, ..., x_t\}) = \{y_1, y_2, ..., y_t\}$. Let $X'' = \{x_1, x_2, \ldots, x_t\}$ and note $N(X'') = Y'' = \{y_1, y_2, ..., y_t\}$. By Lemma 2, we see that for any spanning tree of G, X'' contains at least one pendant vertex. So, $i_{T^*}(X') \leq (k-1)$.We see that

$$\begin{aligned} i(T^*) &= i_{T^*}(X') + i_{T^*}(X \setminus X') + i_{T^*}(Y') + i_{T^*}(Y \setminus Y') \\ &\leq (k-1) + i_{T^*}(X \setminus X') + (k-1) + i_{T^*}(Y \setminus Y') = i(T). \end{aligned}$$

Again, we have $i(T^*) \leq i(T)$ which implies that T is a MIST. Thus, we obtained a MIST T in which x_1 and y_k are pendant and support vertex of y_k is of degree at least 3.

Now, suppose T contains a cycle. We now modify T to be a spanning tree. Let C be a cycle contained in T. This implies that there is a vertex $v \in Y \setminus Y'$ such that $vx_i, vx_j \in E(C)$ with $i < j \leq k$. We remove the edge vx_i from T. This decreases the degree of v by 1. We repeat this process until no cycles remain in T. Let $A \subseteq Y \setminus Y'$ with $|A| = \alpha$ be the set of the vertices made pendant in this process. Suppose $i_{T^*}(Y') \geq (k-\alpha)$. As $N(Y') = X'$, the subgraph of T^* induced by $X' \cup Y' \cup A$ has at least $(2k - \alpha) + k + 2\alpha = 2k + \alpha$ edges. As $|X' \cup Y' \cup A| = 2k + \alpha$, this contradicts the fact that T^* is a tree. Thus we may assume $i_{T^*}(Y') \leq k - \alpha - 1$. As before, we may assume $d_T(x_k) \geq 3$. It follows,

$$
\begin{aligned}
i(T^*) &= i_{T^*}(X') + i_{T^*}(X \setminus X') + i_{T^*}(Y') + i_{T^*}(Y \setminus Y') \\
&\le (k-1) + i_{T^*}(X \setminus X') + (k-\alpha-1) + i_{T^*}(Y \setminus Y') \\
&\le (k-1) + i_{T^*}(X \setminus X') + (k-1) + (i_{T^*}(Y \setminus Y') - \alpha) = i(T)
\end{aligned}
$$

This implies that T is also a MIST. Thus, we obtained a MIST T in which x_1 and y_k are pendant and support vertex of y_k is of degree at least 3. □

We state similar results when the vertices are of type 2. By symmetry, the proofs of Lemmas 6 and 7 follow from Lemmas 4 and 5.

Lemma 6 *Let $X' = \{x_1, x_2, \ldots, x_k\} \subseteq X$, $Y' = \{y_1, y_2, \ldots, y_k, y_{k+1}\} \subseteq Y$. Furthermore, suppose each vertex of X' and Y' is of type 2 except y_{k+1}, $l(y_{k+1}) = x_k$ and $N(Y') = X'$. Then there exists a MIST T of G, in which y_1 and y_{k+1} are pendant. Moreover; if $Y \setminus Y' \neq \emptyset$, then support vertex of y_{k+1} is of degree at least 3 in T.*

Lemma 7 *Let $X' = \{x_1, x_2, \ldots, x_k\} \subseteq X$, $Y' = \{y_1, y_2, \ldots, y_k\} \subseteq Y$. Furthermore, suppose each vertex of X' and Y' is of type 2 except x_k, $l(x_k) = y_k$ and $N(X') = Y'$.*

(a) If $y_i x_{i+1} \in E(G)\ \forall\ 1 \le i \le (k-1)$, then there exists a MIST T of G, in which x_1 is pendant.
(b) If $\exists\ 1 \le t \le (k-1)$ such that $y_t x_{t+1} \notin E(G)$, then there exists a MIST T of G, in which y_1 and x_k are pendant. Moreover; if $X \setminus X' \neq \emptyset$, then support vertex of x_k is of degree at least 3 in T.

Next, we propose an algorithm to find a MIST of G based on the Lemmas 4, 5, 6 and 7. In our algorithm, we first find a vertex u such that it is a pendant vertex in some MIST T of G and the degree of support vertex of u in T is at least 3. Now, if we remove u from G and call the remaining graph G', then we see that the number of internal vertices in a MIST of G is same as the number of internal vertices in a MIST of G'. Note that we can easily construct a MIST of G from a MIST of G' by adding the pendant vertex u to the corresponding support vertex. So, after finding the vertex u, the problem is reduced to finding MIST of $G \setminus \{u\}$, say G'. We continue doing the same until no such vertex u exists and then the resultant graph has a Hamiltonian path.

In our algorithm, we visit the vertices alternatively from the partitions X and Y. We consider two special orderings $(x_1, y_1, x_2, y_2, \ldots)$ and $(y_1, x_1, y_2, x_2, \ldots)$ of $V(G)$ which we call α and β, respectively. Below, we describe the method to find a vertex u which is pendant in some MIST T of G and $d_T(S(u))$ is at least 3.

We first visit the vertices of G in the ordering α and search for the first vertex, which is not of type 1. Let u be such a vertex. If $u \in X$ or $u \in Y$ and the conditions of part (b) of Lemma 5 are satisfied, then there exists a MIST T of G in which u is a pendant vertex and the degree of support vertex of u in T is at least 3. So, we remove u from G and find a MIST T' of $G \setminus \{u\}$. Later, we obtain a MIST of G by adding u to T'. But, if $u \in Y$, say $u = y_k$ and conditions of part (a) of Lemma 5 are satisfied, then there exists a MIST T of G in which y_1 is a pendant vertex. In this case, we start visiting the vertices of G in the ordering β, starting from y_1. At this step, we do not maintain any information from α search.

Now, let u be the first vertex not of type 2 in the ordering β. If $u \in Y$ or $u \in X$ and the conditions of part (b) of Lemma 7 are satisfied, then there exists a MIST T of G in which u is a pendant vertex and the degree of support vertex of u in T is at least 3. So, we remove u from G and find a MIST T' of $G \setminus \{u\}$. Later, we obtain a MIST of G by adding u to T'. Here, if $u \in X$ and conditions of part (a) of Lemma 7 are satisfied, then there exists a MIST T of G in which x_1 is a pendant vertex. But, we have already explored this possibility while visiting the vertices of G in the ordering α. So, we do not get such a vertex u. To see this, suppose that we get such a vertex u. Then, $u = x_t$ for some t, where $t > k$. Now, part (a) of Lemma 7 tells that $y_i x_{i+1} \in E(G)$ for all $1 \leq i \leq (t-1)$ implying that $y_k x_{k+1} \in E(G)$. But, while visiting the vertices in the ordering α, we got a vertex y_k satisfying $l(y_k) = x_k$, so $y_k x_{k+1} \notin E(G)$, a contradiction.

The detailed procedure for computing a MIST of a bipartite permutation graph is presented in Algorithm 2. Algorithm 2 either finds a vertex which is not of type 1 or a vertex which is not of type 2. When such a vertex u is found, we call u as an *encountered* vertex. All the encountered vertices are found while executing the steps written in lines 4, 11, 17, 22, 31 or 39 of Algorithm 2. We see that the algorithm returns a spanning tree T of G. Before proving the correctness of the Algorithm 2, we state a necessary lemma.

Lemma 8 *Let G be the input bipartite permutation graph for the Algorithm 2 and a_1 denotes the first encountered vertex in either the α or β search. Suppose that T is the spanning tree of G returned by Algorithm 2. Let $X_1 \subseteq X$ be the set of vertices which are visited from X side till a_1 and $Y_1 \subseteq Y$ be the set of vertices which are visited from Y side till a_1. Then there exists a MIST T^* of G such that $E(T^*[X_1 \cup Y_1]) = E(T[X_1 \cup Y_1])$.*

Proof We have four cases to consider.

Case 1: $a_1 \in X$ and it is not of type 1. Then the vertex a_1 was found when flag = 1 in Algorithm 2, that is, when searching for the first vertex not of type 1. Let $a_1 = x_{k+1}$ for some k. Then the sets $X' = \{x_1, x_2, \ldots, x_k, x_{k+1}\} \subseteq X$, $Y' = \{y_1, y_2, \ldots, y_k\} \subseteq Y$ satisfy the hypothesis of Lemma 4. Thus by Lemma 4, there exists a MIST T^* of G such that $E(T^*[X_1 \cup Y_1]) = \{x_1 y_1, y_1 x_2, x_2 y_2, \ldots, x_k y_k, y_k x_{k+1}\}$. In particular, $E(T^*[X_1 \cup Y_1]) = E(T[X_1 \cup Y_1])$.

Case 2: $a_1 \in Y$ and it is not of type 1. Then the vertex a_1 was also found when flag = 1 in the algorithm. Let $a_1 = y_k$ for some k. Then the sets $X' = \{x_1, x_2, \ldots, x_k\} \subseteq X$, $Y' = \{y_1, y_2, \ldots, y_k\} \subseteq Y$ satisfy the hypothesis of part (b) of Lemma 5. Thus by Lemma 5, there exists a MIST T^* of G such that $E(T'[X_1 \cup Y_1]) = \{x_1 y_1, y_1 x_2, x_2 y_2, \ldots, x_k y_k\} = E(T[X_1 \cup Y_1])$.

By symmetry, the other two cases ($a_1 \in X$ and it is not of type 2; $a_1 \in Y$ and it is not of type 2) follow from Lemmas 6 and 7 . Thus there exists a MIST T^* of G such that $E(T^*[X_1 \cup Y_1]) = E(T[X_1 \cup Y_1])$ in all cases. □

Now, we prove the correctness of Algorithm 2.

Theorem 4 *Algorithm 2 returns a maximum internal spanning tree of G.*

Proof Let T^* be a MIST of G and T be the spanning tree of G returned by Algorithm 2. Recall in the execution of Algorithm 2, we either search for a vertex not of type 1 with

Algorithm 2 Algorithm for finding a MIST of a bipartite permutation graph G

Input: A bipartite permutation graph G and a strong ordering $(<_X, <_Y) = (x_1, x_2, \ldots, x_{n_1}, y_1, y_2, \ldots, y_{n_2})$ of $V(G)$. **Output:** A MIST T of G.

$V(T) = X \cup Y$, $E(T) = \emptyset$, $t = 0$; $flag = 1$;
$\alpha = (x_1, y_1, x_2, y_2, ...)$ and $\beta = (y_1, x_1, y_2, x_2, ...)$;
Visit the vertices of $V(G)$ in the ordering α;
Let u be the first vertex with minimum index in the ordering α which is not of type 1;
while $flag == 1$ **do**
    if $u \in X$ **then**
        Let $u = x_{k+1}$ for some k;
        if $k + 1 \neq n_1$ **then**
            $t = t + 1$; rename x_{k+1} as a_t; $E(T) = E(T) \cup \{y_k a_t\}$;
            Rename x_i as x_{i-1} for every $k + 2 \leq i \leq n_1$; $n_1 = n_1 - 1$;
            Continue looking for a next vertex which is not of type 1 in the ordering α, call it u;
        else
            $E(T) = E(T) \cup \{x_1 y_1, y_1 x_2, x_2 y_2, \ldots, x_k y_k, y_k x_{k+1}\}$; return T;
    else
        Let $u = y_k$ for some k;
        if $x_i y_{i+1} \in E(G)\ \forall\ 1 \leq i \leq (k - 1)$ **then**
            Find a vertex which is not of type 2 in the ordering β starting from y_1, call it u; $flag = 2$;
        else
            if $k \neq n_2$ **then**
                $t = t + 1$; rename y_k as a_t; $E(T) = E(T) \cup \{x_k a_t\}$;
                Rename y_i as y_{i-1} for every $k + 1 \leq i \leq n_2$; $n_2 = n_2 - 1$;
                Continue looking for a next vertex which is not of type 1 in the ordering α, call it u;
            else
                $E(T) = E(T) \cup \{x_1 y_1, y_1 x_2, x_2 y_2, \ldots, y_{k-1} x_k, x_k y_k\}$; return T;
while $flag == 2$ **do**
    if $u \in Y$ **then**
        Let $u = y_{k+1}$ for some k;
        if $k + 1 \neq n_2$ **then**
            $t = t + 1$; rename y_{k+1} as a_t; $E(T) = E(T) \cup \{x_k a_t\}$;
            Rename y_i as y_{i-1} for every $k + 2 \leq i \leq n_2$; $n_2 = n_2 - 1$;
            Continue looking for a next vertex which is not of type 2 in the ordering β, call it u;
        else
            $E(T) = E(T) \cup \{y_1 x_1, x_1 y_2, y_2 x_2, \ldots, y_k x_k, x_k y_{k+1}\}$; return T;
    else
        Let $u = x_k$ for some k;
        if $k \neq n_1$ **then**
            $t = t + 1$; rename x_k as a_t; $E(T) = E(T) \cup \{y_k a_t\}$;
            Rename x_i as x_{i-1} for every $k + 1 \leq i \leq n_1$; $n_1 = n_1 - 1$;
            Continue looking for a next vertex which is not of type 2 in the ordering β, call it u;
        else
            $E(T) = E(T) \cup \{y_1 x_1, x_1 y_2, y_2 x_2, \ldots, x_{k-1} y_k, y_k x_k\}$; return T;

the ordering α or we search for a vertex not of type 2 with the ordering β. This is ensured since either we never arrive at line 17 or we arrive at it once and after that flag remains 2 throughout the algorithm. Let $a_1, a_2, \ldots, a_p$ be the sequence of vertices encountered in the execution of Algorithm 2. Let X_1 and Y_1 denote the set of vertices visited till a_1 from X and Y side respectively. For $1 < i < p$, let X_i denotes the set of vertices visited from X side after a_{i-1} and upto a_i. Similarly, let Y_i denotes the set of vertices visited from Y side after a_{i-1} and upto a_i. Let X_p and Y_p denote the set of all vertices visited after a_{p-1} from X and Y side respectively.

First suppose Algorithm 2 is searching for a vertex not of type 1 with the ordering α and it never arrives at line 17. This means that flag is 1 throughout the algorithm. To prove that T is a MIST of G, we will prove that T^* can be modified so that it remains a MIST of G and $E(T^*)$ is same as $E(T)$, that is,

$$E(T^*[\bigcup_{j=1}^{p} X_j \cup \bigcup_{j=1}^{p} Y_j]) = E(T[\bigcup_{j=1}^{p} X_j \cup \bigcup_{j=1}^{p} Y_j]). \tag{1}$$

We prove (1) using induction on p. If $p = 1$, we have $E(T^*[X_1 \cup Y_1]) = E(T[X_1 \cup Y_1])$ due to Lemma 8. Hence, (1) is true for $p = 1$. Assume that (1) is true for $p = i$.

We now show that (1) is true for $p = i + 1$. So, consider vertex a_{i+1} for $i \geq 2$. Two possible cases arise.

Case 1: $a_{i+1} \in X$.

If $a_j \in X$ for each j, $1 \leq j \leq i$, then define $X^* = \cup_{j=1}^{i+1} X_j$ and $Y^* = \cup_{j=1}^{i+1} Y_j$. Otherwise, let j be the largest index such that $j \in \{1, 2, \ldots, i\}$ and $a_j \in Y$. Then define $X^* = \cup_{t=j+1}^{i+1} X_t$ and $Y^* = \cup_{t=j+1}^{i+1} Y_t$. Note that, in both the cases, we have $N(X^*) = Y^*$.

As $N(X^*) = Y^*$, by Lemma 2 we have that the number of pendant vertices from X^* in any spanning tree of G is at least $|X^*| - |Y^*| + 1$. Therefore, $i_{T^*}(X^*) \leq |Y^*| - 1$.

If (1) is not true for $p = i + 1$, we remove all edges of T^* who have one end in $\cup_{j=1}^{i} (X_j \cup Y_j)$ and the other in $(X_{i+1} \cup Y_{i+1})$ and all edges incident with the vertices of X_{i+1} within T^*. We then add all edges from $E(T[X_{i+1} \cup Y_{i+1}])$ and the edge of T which connects $\cup_{j=1}^{i} (X_j \cup Y_j)$ to $(X_{i+1} \cup Y_{i+1})$ in T^*. If cycles were created in this process, then we can remove those cycles without introducing more pendant vertices using the method discussed in Lemmas 4 and 5. Let T^*_{new} denote this updated tree. Define $X' = X \setminus (X^* \cup (\cup_{t=i+2}^{p} X_t))$ and $Y' = X \setminus (Y^* \cup (\cup_{t=i+2}^{p} Y_t))$. We have,

$$\begin{aligned} i(T^*) &= i_{T^*}(X') + i_{T^*}(X^*) + i_{T^*}(\bigcup_{t=i+2}^{p} X_t) + i_{T^*}(Y') + i_{T^*}(Y^*) + i_{T^*}(\bigcup_{t=i+2}^{p} Y_t) \\ &\leq i_{T^*}(X') + |Y^*| - 1 + i_{T^*}(\bigcup_{t=i+2}^{p} X_t) + i_{T^*}(Y') + |Y^*| + i_{T^*}(\bigcup_{t=i+2}^{p} Y_t) \\ &= i(T^*_{new}), \quad \text{as } i_{T^*_{new}}(X^*) = |Y^*| - 1. \end{aligned}$$

Thus T^*_{new} is also a MIST of G and (1) is true for $p = i + 1$ with $T^* = T^*_{new}$.

Case 2: $a_{i+1} \in Y$. Here, we discuss two subcases.

Subcase 2.1: $a_j \in Y \ \forall\ j; 1 \le j \le i$.

Here, for $X^* = \cup_{j=1}^{i+1} X_j$ and $Y^* = \cup_{j=1}^{i+1} Y_j$, we have $|X^*| = |Y^*| - i$. As $N(Y^*) = X^*$, by Lemma 2 we have that the number of pendant vertices from Y^* in any spanning tree of G is at least $|Y^*| - |X^*| + 1 = i + 1$. Therefore $i_{T^*}(Y^*) \le |Y^*| - (i+1)$. Here, $a_1 \in Y$, so, let $a_1 = y_k$ for some k. As we have assumed that flag is 1, this implies that there exists an index t, $1 \le t \le k-1$ such that $x_t y_{t+1} \notin E(G)$. So, for $X' = \{x_1, x_2, \ldots, x_t\}$ and $Y' = \{y_1, y_2, \ldots, y_t\}$, we have $N(X') = Y'$. Now, by Lemma 2, we know that the number of pendant vertices within X' in any spanning tree of G is at least $|X'| - |Y'| + 1 = 1$. So, $i_{T^*}(X') \le |X'| - 1$, implying that $i_{T^*}(X^*) \le |X^*| - 1$. If (1) is not true for $p = i + 1$, we construct another spanning tree T^*_{new} of G from T^* in the following way: remove all edges of T^* who have one end in $\cup_{j=1}^{i}(X_j \cup Y_j)$ and the other in $(X_{i+1} \cup Y_{i+1})$ and all edges incident with the vertices of Y_{i+1} within T^*. Then, add all edges from $E(T[X_{i+1} \cup Y_{i+1}])$ and the edge of T which connects $\cup_{j=1}^{i}(X_j \cup Y_j)$ to $(X_{i+1} \cup Y_{i+1})$ in T^*. As before, if cycles are present, further modify T^* to remove these cycles without introducing more pendant vertices. Now we have,

$$\begin{aligned} i(T^*) &= i_{T^*}(X^*) + i_{T^*}(X \setminus X^*) + i_{T^*}(Y^*) + i_{T^*}(Y \setminus Y^*) \\ &\le |X^*| - 1 + i_{T^*}(X \setminus X^*) + |Y^*| - (i+1) + i_{T^*}(Y \setminus Y^*) = i(T^*_{new}) \end{aligned}$$

Subcase 2.2: $a_j \in X$ for some $1 \le j \le i$.

We choose the largest $j \in \{1, 2, \ldots, i\}$ such that $a_j \in X$. Then for $X^* = \cup_{t=j+1}^{i+1} X_t$ and $Y^* = \cup_{t=j+1}^{i+1} Y_t$, we have $|X^*| = |Y^*| - (i - j)$. As $N(Y^*) = X^*$, by Lemma 2 we have that the number of pendant vertices from Y^* in any spanning tree of G is at least $|Y^*| - |X^*| + 1 = i - j + 1$. Therefore, $i_{T^*}(Y^*) \le |Y^*| - (i - j + 1)$. If (1) is not true for $p = i + 1$, we construct another spanning tree T^*_{new} of G from T^* using the same way as done in subcase 2.1. We have,

$$\begin{aligned} i(T^*) &= i_{T^*}(\bigcup_{t=1}^{j} X_t) + i_{T^*}(X^*) + i_{T^*}(\bigcup_{t=i+2}^{p} X_t) + i_{T^*}(\bigcup_{t=1}^{j} Y_t) + i_{T^*}(Y^*) + i_{T^*}(\bigcup_{t=i+2}^{p} Y_t) \\ &\le i_{T^*}(\bigcup_{t=1}^{j} X_t) + |X^*| + i_{T^*}(\bigcup_{t=i+2}^{p} X_t) + i_{T^*}(\bigcup_{t=1}^{j} Y_t) + |Y^*| - (i - j + 1) \\ &\quad + i_{T^*}(\bigcup_{t=i+2}^{p} Y_t) \\ &= i(T^*_{new}). \end{aligned}$$

Thus T^*_{new} is also a MIST of G and (1) is true for $p = i + 1$ with $T^* = T^*_{new}$.

Hence, we get that (1) is true for all p, that is, $E(T^*[X \cup Y]) = E(T[X \cup Y])$ in each possible case, when flag is 1.

If algorithm arrives at line 17, then flag changes to 2 and it remains 2 throughout the algorithm. So, it searches vertex not of type 2 in the ordering β starting from y_1.

There will be analogous arguments for this case also, using Lemmas 6 and 7 instead. For a quick justification why, with the assumption flag = 1, the above analysis fails if we encounter a vertex, say $u_1 = y_j$, such that u_1 is not type 1 and $x_i y_{i+1} \in E(G)$ for all $1 \leq i \leq (j-1)$. The analogous failure case for the flag = 2 is, when we encounter a vertex $u_2 = x_k$ that is not of type 2 and $y_i x_{i+1} \in E(G)$ for all $1 \leq i \leq (k-1)$. Note that these cases cannot simultaneously occur. Otherwise the analysis is symmetric. Consequently, Algorithm 2 returns a maximum internal spanning tree of G. $\square$

Now, we discuss the running time of Algorithm 2. Suppose Algorithm 2 returns a MIST T. Recall that we visit the vertices in one of the orders $\alpha = (x_1, y_1, x_2, y_2, \ldots)$, or $\beta = (y_1, x_1, y_2, x_2, \ldots)$. Furthermore, any vertex encountered during the execution of the algorithm must be pendant in T. As we never visit the same vertex twice, these pendant vertices are found in linear-time. The remaining graph must have a Hamiltonian path, and finding the Hamiltonian path is also linear-time in our algorithm. So, all the steps of Algorithm 2 can be executed in $O(n+m)$ time. Hence we have the following corollary.

Corollary 2 *A maximum internal spanning tree of a bipartite permutation graph can be computed in linear-time.*

6 Bounds for chain graphs

A bipartite graph $G = (X, Y, E)$ is a *chain graph* if the neighborhoods of the vertices of X form a chain, that is, the vertices of X can be linearly ordered, say $\{x_1, x_2, \ldots, x_{n_1}\}$ such that $N(x_1) \subseteq N(x_2) \subseteq \ldots \subseteq N(x_{n_1})$ and $n_1 = |X|$. If $G = (X, Y, E)$ is a chain graph, then the neighborhoods of the vertices of Y also form a chain. If $n_2 = |Y|$, an ordering $\alpha = (x_1, x_2, \ldots, x_{n_1}, y_1, y_2, \ldots, y_{n_2})$ is called a *chain ordering* if $N(x_1) \subseteq N(x_2) \subseteq \ldots \subseteq N(x_{n_1})$ and $N(y_1) \supseteq N(y_2) \supseteq \ldots \supseteq N(y_{n_2})$. If a vertex x_i appears before x_j in chain ordering, we write $x_i < x_j$. Given a chain graph G, a chain ordering of G can be computed in linear-time (Heggernes and Kratsch 2007). Note that a chain ordering is also a strong ordering. So, every chain graph is also bipartite permutation graph.

In this section, we will prove the following lower bound for number of internal vertices in a MIST of a chain graph G.

Theorem 5 *For a chain graph G, let P^* be an optimal path cover of G. Then $Opt(G) \geq |E(P^*)| - 2$.*

In order to prove Theorem 5, we look at optimal path covers of bipartite permuation graphs. Srikant et al (1993) gave an algorithm to find an optimal path cover of a bipartite permutation graph. Note that this algorithm applies to chain graphs as well. We will recall the algorithm given in (Srikant et al 1993), but first we cover some notations used in the algorithm. A path cover $P^* = \{P_1, P_2, \ldots, P_k\}$ is *contiguous* if it satisfies the following two conditions:

1. If $x \in X$ is the only vertex in P_i and if $x' < x < x''$, then x' and x'' belong to different paths.

2. If xy is an edge in P_i and $x'y'$ is an edge in P_j, where $i \neq j$ and $x < x'$, then $y < y'$.

A path P is *contiguous* if it is one of the following forms: $x_i y_j x_{i+1} y_{j+1} \dots y_{t-1} x_r$, $x_i y_j x_{i+1} y_{j+1} \dots y_{t-1} x_r y_t$, $y_j x_i y_{j+1} x_{i+1} \dots x_{r-1} y_t x_r$, or $y_j x_i y_{j+1} x_{i+1} \dots x_{r-1} y_t$ such that $r \geq i$ and $t \geq j$. Note that every path in a contiguous path cover is contiguous. Let P be a contiguous path which ends with some edge, say $x_p y_q$. If $y_q x_{p+1} \notin E(G)$, then we say that the path P is not extendable on the right. A contiguous path is said to be a *maximal contiguous path* if it is not extendable on the right. An optimal path cover $P^* = \{P_1, \dots, P_k\}$ is a *maximum optimal path cover* if each P_i covers the maximum number of vertices in $V(G) \setminus \{P_1 \cup P_2 \cup \dots P_{i-1}\}$. According to (Srikant et al 1993), there exists an optimal path cover which is a maximum optimal path cover for any bipartite permutation graph G such that every path in the path cover is a maximal contiguous path.

As a chain graph is an instance of a bipartite permutation graph, we recall the algorithm from (Srikant et al 1993) which finds this desired maximum optimal path cover for a chain graph (Algorithm 3). From this point, we will refer such a path cover as an optimal path cover only.

Algorithm 3 Algorithm for finding an optimal path cover of G

Input: A chain graph $G = (X, Y, E)$ with the ordering of its vertices
Output: An optimal path cover P of G

```
Mark all vertices in X and Y as not visited; let P = Ø.
while all vertices of G are not visited do
    Let x and y be the first vertices in X and Y which are not visited.
    Let P_x and P_y be the maximal contiguous paths starting from x and y, respectively.
    Q := Maximum of P_x and P_y.
    P := P ∪ Q.
    Mark all vertices in Q as visited.
Output P.
```

Let P and Q be two distinct paths in a graph G. We define a *combining edge* of P and Q as an edge of G whose one end vertex is in path P and another end vertex is in path Q. Let P^* be the optimal path cover obtained from Algorithm 3. A path in P^* is nontrivial if it has at least two vertices. We assume that the path components of P^* are ordered with respect to their appearance in Algorithm 3. Before proving Theorem 5, we prove some lemmas first.

Lemma 9 *Let P and Q be two consecutive nontrivial path components in P^* such that P ends at a vertex of X side and Q starts with a vertex of X side. Then there exists a combining edge of P and Q which joins an internal vertex of P to a pendant vertex of Q.*

Proof Suppose that P ends at some vertex x and Q starts from some vertex x', where $x < x'$. Let y be the vertex adjacent to x in P, then $y \in N(x')$ as G is a chain graph. So, the edge yx' is a combining edge for path components P and Q. We see that y is internal in P and x' is pendant in Q. Fig. 5 provides an illustration. □

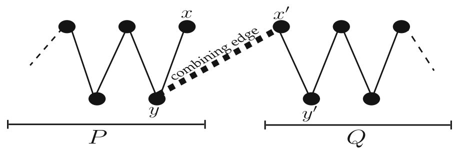


Fig. 5 Two consecutive nontrivial path components $P, Q \in P^*$ such that P ends at X side and Q starts from X side

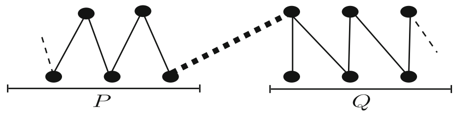


Fig. 6 Two consecutive nontrivial path components $P, Q \in P^*$ such that P ends at Y side and Q starts from Y side

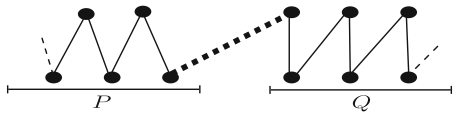


Fig. 7 Two consecutive nontrivial path components $P, Q \in P^*$ such that P ends at Y side and Q starts from X side

Lemma 10 *There are no consecutive nontrivial path components P and Q in P^* such that:*

1. *P ends at a vertex of Y side and Q starts with a vertex of Y side, or,*
2. *P ends at a vertex of Y side and Q starts with a vertex of X side.*

Proof First, suppose that there are two consecutive nontrivial path components P and Q in P^* such that P ends at a vertex of Y side and Q starts with a vertex of Y side. As P^* was constructed from Algorithm 3, every path component in P^* is maximal contiguous. But, in this case, P is extendable on right. So, this case will not arise. Fig. 6 provides an illustration.

Now, suppose that there are two consecutive nontrivial path components P and Q in P^* such that P ends at a vertex of Y side and Q starts with a vertex of X side. Due to the similar reason, this case will also not arise. Fig. 7 provides an illustration. Hence, the lemma holds. □

Lemma 11 *Let P and Q be two consecutive nontrivial path components in P^* such that P ends at a vertex of X side and Q starts with a vertex of Y side. Then the following is true:*

1. *If Q ends at a vertex of X side, then Q can be modified to another path Q' such that $V(Q) = V(Q')$, Q' is also a maximal contiguous path and there exists a combining edge of P and Q' which joins an internal vertex of P to a pendant vertex of Q'.*
2. *If Q ends at a vertex of Y side, then there exists a combining edge of P and Q which joins an internal vertex of P to an internal vertex of Q.*

Proof Suppose that P ends at some vertex x and Q starts from some vertex $y = y_j$. Let y' be the neighbor of x in P.

First, suppose that Q ends at a vertex of X side. Let $Q = yx_i y_{j+1} \ldots x_t y_k x_{t+1}$. As G is a chain graph, we have that $Q' = x_i y x_{i+1} \ldots y_{k-1} x_{t+1} y_k$ is also a path in G. Note that $V(Q) = V(Q')$ and Q' is a maximal contiguous path. We can replace Q with Q' in the path cover P^*. Now we see that edge $y'x_i \in E(G)$ as $N(x) \subseteq N(x_i)$.

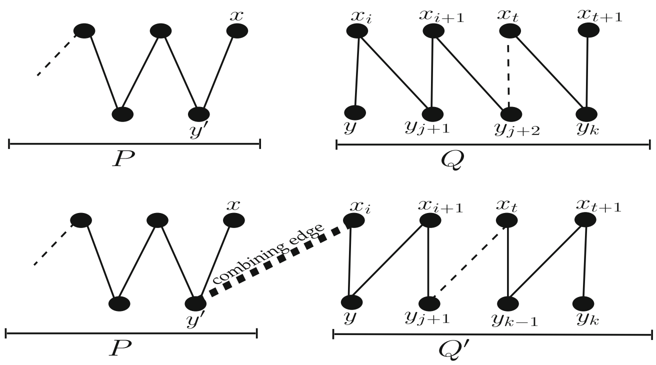
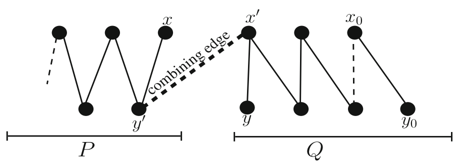


Fig. 8 Two consecutive nontrivial path components $P, Q \in P^*$ such that P ends at X side, Q starts from Y side and Q ends at X side

Fig. 9 Two consecutive nontrivial path components $P, Q \in P^*$ such that P ends at X side, Q starts from Y side and Q ends at Y side



So, the edge $y'x_i$ is a combining edge for path components P and Q'. We see that y' is internal in P and x_i is pendant in Q'. Fig. 8 provides an illustration.

Now, suppose that Q ends at a vertex of Y side. Let x' be the neighbor of y in Q. Since, $x < x'$ and G is a chain graph, edge $y'x' \in E(G)$. Here, we consider the edge $y'x'$ as the combining edge for path components P and Q. We see that y' is internal in P and x' is also internal in Q. Fig. 9 provides an illustration. □

Now, we give the proof of the Theorem 5.

Proof of Theorem 5 Let P^* be the optimal path cover of G obtained from Algorithm 3. Suppose P^* has k path components $P_1, P_2, \ldots, P_k$. Let us denote number of edges of the component P_i by e_i for every $1 \leq i \leq k$. This implies that $e_1 + e_2 + \ldots + e_k = |E(P^*)|$. Note that the number of internal vertices in a path with e_i edges is $e_i - 1$.

Let P and Q be two consecutive nontrivial path components in P^*. Then using Lemmas 9, 10 and 11, we see that in each possible case, we get a combining edge of P and Q. If we connect each consecutive nontrivial path component with these combining edges and connect the remaining single vertex components by an arbitrary edge incident with an internal vertex of a nontrivial path component, we obtain a spanning tree of G.

First, assume that we never get P and Q such that P ends at a vertex of X side, Q starts from a vertex of Y side and Q ends at a vertex of Y side. Note then every combining edge connects one internal vertex of P and one pendant vertex of Q. So, $i(T) = e_1 - 1 + e_2 + e_3 + \ldots + e_k = |E(P^*)| - 1$.

Now, assume that there exists some P and Q such that P ends at a vertex of X side, Q starts from a vertex of Y side and Q ends at a vertex of Y side. Here, suppose

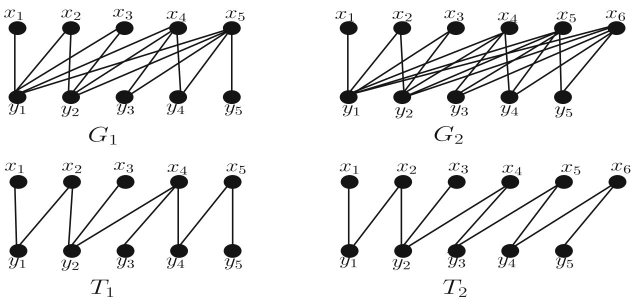


Fig. 10 examples showing that bounds are tight

that Q ends at the vertex y_0 and let x_0 be the neighbor of y_0 in Q. We claim that $x_0 = x_{n_1}$. If this is not the case then there exists a vertex x^* in X such that $x^* > x_0$ and $x^* \notin V(Q)$. But, since G is a chain graph, we have that $(y_0, x^*) \in E(G)$ which makes Q, a non-maximal path, a contradiction. Thus, $x_0 = x_{n_1}$ which implies that, if $Q'' \in P^*$ and appears after Q in Algorithm 3, then Q'' is a single vertex path component containing a vertex of Y. This implies that this case appears only once. So, $i(T) = e_1 - 1 + e_2 + e_3 + \ldots + e_k - 1 = |E(P^*)| - 2$.

Hence, the number of internal vertices in any MIST of G is at least $|E(P^*)| - 2$, that is, $Opt(G) \geq |E(P^*)| - 2$. □

Combining Theorem 1 and Theorem 5, we can state the following corollary.

Corollary 3 *For a chain graph G, if P^* denotes an optimal path cover then $Opt(G)$ is either $|E(P^*)| - 1$ or $|E(P^*)| - 2$.*

Now, we give examples of chain graphs which shows that both the bounds (given by Theorem 1 and Theorem 5) are tight. In Fig. 10, G_1 and G_2 are chain graphs and T_1 and T_2 are Maximum Internal Spanning Trees of G_1 and G_2 respectively. We can see that optimal path cover obtained from Algorithm 3 for the graph G_1 is $\{x_1y_1x_2y_2x_3, y_3x_4y_4x_5y_5\}$ which has 8 edges and its MIST T_1 has 6 internal vertices i.e. $Opt(G_1) = |E(P^*)| - 2 = 8 - 2 = 6$. Using Lemma 2, it can be verified that any MIST of G_1 has at least four pendant vertices, two from X side and two from Y side; so, G_1 can have at most 6 internal vertices in its MIST. Hence, T_1 is indeed a MIST of G_1. In a similar manner, optimal path cover obtained from Algorithm 3 for the graph G_2 is $\{x_1y_1x_2y_2x_3, y_3x_4y_4x_5y_5x_6\}$ which has 9 edges and its MIST T_2 has 8 internal vertices i.e. $Opt(G_2) = |E(P^*)| - 1 = 9 - 1 = 8$.

7 Relationship between *Opt*(*G*) and |*E*(*P**)|

In this section, we summarize the relationship between $Opt(G)$ and $|E(P^*)|$ for all the graph classes discussed in the previous sections.

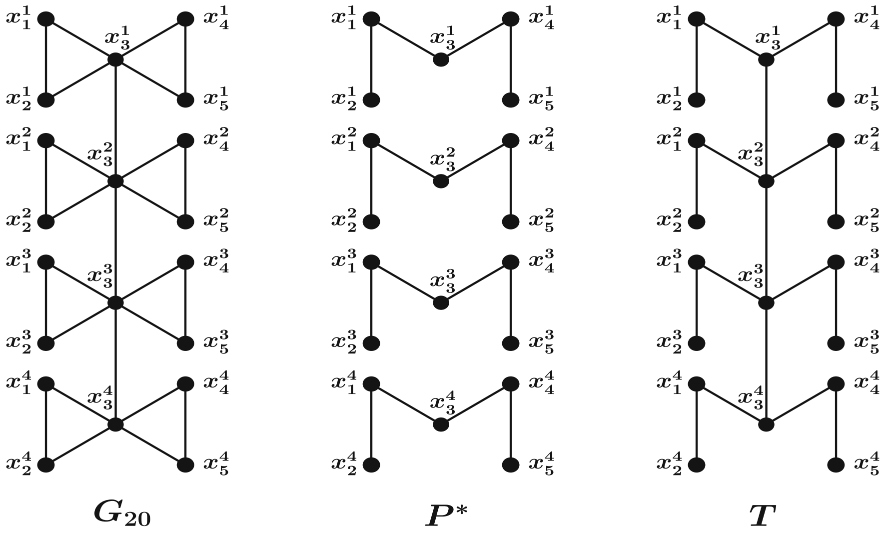


Fig. 11 Graph G_{20}, its optimal path cover P^* and its MIST T

7.1 Block/cactus graph

Let G be a block or cactus graph. Then we show that there does not exist a constant k such that $Opt(G) \geq |E(P^*)| - k$ where P^* is an optimal path cover of G. Recall Corollary 1 states that $Opt(G) = n - |Bad(G)|$ and Theorem 1 states $Opt(G) \leq |E(P^*)| - 1$. Note that the number of edges in the optimal path cover P^* and the number of components in P^* adds up to n. So, we see that $n - |Bad(G)| = |E(P^*)| - (|Bad(G)| - |P^*|)$. Thus, $Opt(G) = |E(P^*)| - (|Bad(G)| - |P^*|)$ for both block and cactus graphs.

For every integer $n = 5k$ $(k \geq 1)$, we construct a connected graph G_n with n vertices and $Opt(G_n) = |E(P^*)| - O(n)$. The graph G_n is both a block graph and a cactus graph as every block of G_n is either an edge or a clique on three vertices. The vertex set of G_n is $V(G_n) = V_1 \cup V_2 \cup \ldots \cup V_k$, where $V_i = \{x_1^i, x_2^i, \ldots, x_5^i\}$ for each $i \in \{1, 2, \ldots, k\}$. The edge set is $E(G_n) = E_1 \cup E_2 \cup \ldots \cup E_k \cup E'$, where $E_i = \{x_1^i x_2^i, x_2^i x_3^i, x_3^i x_1^i, x_3^i x_4^i, x_4^i x_5^i, x_5^i x_3^i\}$ for each i and E' contains the edges of the form $x_3^i x_3^{i+1}$ for $1 \leq i \leq (k-1)$. Note $|E(G_n)| = 7k - 1$. We obtain an optimal path cover P^* for G_n having $4k$ edges and k components (Pak-Ken 1999). The number of bad blocks in G_n is $2k$. Using Theorem 2, we obtain a MIST T of G_n with $n - |Bad(G)| = 5k - 2k = 3k$ internal vertices. Thus, $Opt(G_n) = 3k = 4k - k = 4k - \frac{n}{5} = |E(P^*)| - O(n)$. Fig. 11 provides an illustration for G_{20}.

Here, we see that $|Bad(G_n)| - |P^*| = 2k - k = k$ which implies that for arbitrary $n = 5k$, we have $Opt(G_n) = |E(P^*)| - k$. So, block and cactus graphs do not have lower bound for $Opt(G)$ of the form $|E(P^*)| - c$ for some fixed natural number c, independent of n.

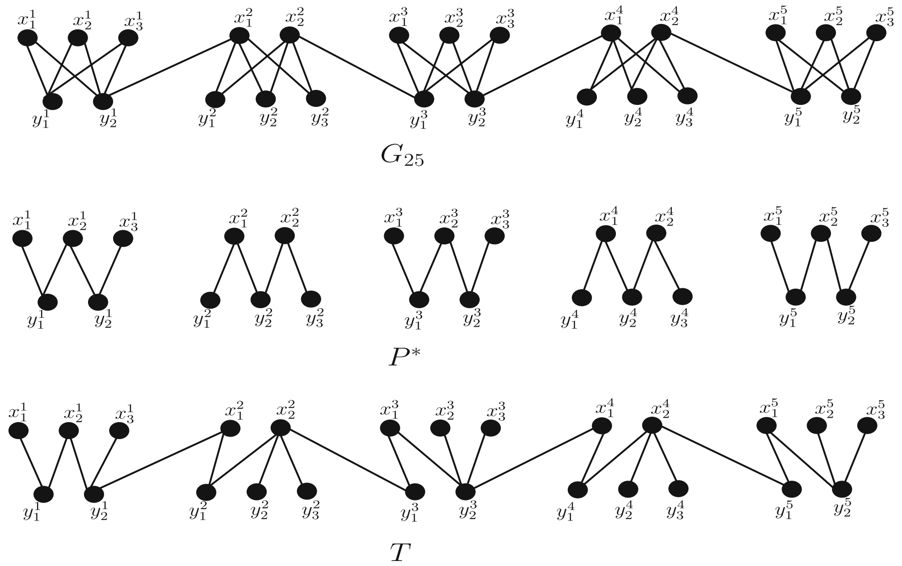


Fig. 12 Graph G_{25}, its optimal path cover P^* from Algorithm 3 and its MIST T from Algorithm 2

7.2 Bipartite permutation graph

Now, let G be a bipartite permutation graph, then $Opt(G)$ cannot be lower bounded with value $|E(P^*)| - k$ for any fixed natural number k. Below, for every natural number k, we give a construction of a bipartite permutation graph such that $Opt(G) = |E(P^*)| - O(5k)$.

For every integer $n = 5k$ $(k \geq 1)$, we construct a connected bipartite permutation graph G_n with n vertices and $Opt(G_n) = |E(P^*)| - O(n)$. For all $1 \leq i \leq k$, let $X_i = \{x_1^i, x_2^i\}$ and $Y_i = \{y_1^i, y_2^i, y_3^i\}$ if i is even and $X_i = \{x_1^i, x_2^i, x_3^i\}$ and $Y_i = \{y_1^i, y_2^i\}$ for odd i. Let $V(G_n) = V_1 \cup V_2 \cup \ldots \cup V_k$ where $V_i = X_i \cup Y_i$ for all $1 \leq i \leq k$. Let $E(G_n) = E_1 \cup E_2 \cup \ldots \cup E_k \cup E'$ where $E_i = \{xy | x \in X_i, y \in Y_i\}$ for each $1 \leq i \leq k$ and E' is the set of edgs of the form $y_2^i x_1^{i+1}$ if i is odd and $x_2^i y_1^{i+1}$ if i is even for each $1 \leq i \leq (k-1)$. We see that G_n is a bipartite permutation graph with n vertices and $n + 2k - 1$ edges. Algorithm 3 gives an optimal path cover P^* for G_n having $4k$ edges and Algorithm 2 gives a MIST with $3k$ internal vertices. So, we get that $Opt(G_n) = 3k = 4k - k = 4k - \frac{n}{5} = |E(P^*)| - O(n)$. Fig. 12 provides an illustration for G_{25}.

Thus $Opt(G)$ for bipartite permutation graphs do not have lower bound of the form $|E(P^*)| - k$ for some fixed natural number k, independent of n.

7.3 Chain graph and cographs

In Corollary 3, we have proved that $|E(P^*)| - 2 \leq Opt(G) \leq |E(P^*)| - 1$ where P^* is an optimal path cover of a chain graph G. For a cograph G, Theorem 3 states that $Opt(G) = |E(P^*)| - 1$ where P^* is an optimal path cover of G.

8 Conclusion

We studied the Maximum Internal Spanning Tree (MIST) problem, a generalization of Hamltonian path problem. As the MIST problem remains NP-hard even for bipartite graphs and chordal graphs due to a reduction from the Hamiltonian path problem (Lai and Wei 1993; Müller 1996), we further investigated the complexity of special instances of these classes, chain graphs, bipartite permutation graphs and block graphs. We also investigated cactus graphs and cographs, finding linear-time algorithms for the MIST problem for each of these graph classes.

Li et al (2018) proved an upper bound for $Opt(G)$ in terms of an optimal path cover. We further studied this relationship between path covers and $Opt(G)$ and showed tight lower bounds for chain graphs and cographs. We also showed this phenomenon does not hold for general graphs with a construction of bipartite permutation graph and block graph such that $Opt(G)$ is arbitrarily far from $|E(P^*)|$.

A convex bipartite graph G with bipartition (X, Y) and an ordering $X = (x_1, x_2, \ldots, x_n)$, is a bipartite graph such that for each $y \in Y$, the neighborhood of y in X appears consecutively. Complexity status of the MIST problem is still open for convex bipartite graphs, which is a superclass of bipartite permutation graphs and subclass of chordal bipartite graphs. Designing an algorithm for MIST in convex bipartite graphs will be a good research direction.

The weighted version of the MIST problem is also well studied in literature (Salamon 2009). Given a vertex-weighted connected graph G, the maximum weight internal spanning tree (MwIST) problem asks for a spanning tree T of G such that the total weight of internal vertices in T is maximized. Since MwIST problem is a generalization of the MIST problem, one may also investigate the complexity status of MwIST problem for some special classes of graphs.

To our knowledge, every known hardness proof for the MIST problem on families of graphs relies on a reduction to Hamiltonian path problem. We leave as an open question if there exists a family of graphs such that Hamiltonian path problem is polynomial time, yet the MIST problem remains NP-hard.

Funding The authors have not disclosed any funding.

Data availibility Enquiries about data availability should be directed to the authors.

Declarations

Conflict of interest The authors have not disclosed any competing interests.

References

Binkele-Raible D, Fernau H, Gaspers S et al (2013) Exact and parameterized algorithms for max internal spanning tree. Algorithmica 65(1):95–128

Chen ZZ, Harada Y, Guo F et al (2018) An approximation algorithm for maximum internal spanning tree. J Comb Optim 35(3):955–979

Cohen N, Fomin FV, Gutin G et al (2010) Algorithm for finding k-vertex out-trees and its application to k-internal out-branching problem. J Comput Syst Sci 76(7):650–662

Fomin FV, Gaspers S, Saurabh S et al (2013) A linear vertex kernel for maximum internal spanning tree. J Comput Syst Sci 79(1):1–6
Garey MR, Johnson DS (1979) Computers and intractability, vol 174. freeman San Francisco
Heggernes P, Kratsch D (2007) Linear-time certifying recognition algorithms and forbidden induced subgraphs. Nord J Comput 14(1–2):87–108
Heggernes P, Van't Hof P, Lokshtanov D et al (2012) Computing the cutwidth of bipartite permutation graphs in linear time. SIAM J Discret Math 26(3):1008–1021
Jung HA (1978) On a class of posets and the corresponding comparability graphs. J Comb Theory Series B 24(2):125–133
Knauer M, Spoerhase J (2015) Better approximation algorithms for the maximum internal spanning tree problem. Algorithmica 71(4):797–811
Lai TH, Wei SS (1993) The edge hamiltonian path problem is np-complete for bipartite graphs. Inf Process Lett 46(1):21–26
Lai TH, Wei SS (1997) Bipartite permutation graphs with application to the minimum buffer size problem. Discret Appl Math 74(1):33–55
Lerchs H (1972) On the clique-kernel structure of graphs. Dept of Computer Science, University of Toronto 1
Li W, Wang J, Chen J, et al (2015) A 2k-vertex kernel for maximum internal spanning tree. In: Workshop on algorithms and data structures, Springer, pp 495–505
Li W, Cao Y, Chen J et al (2017) Deeper local search for parameterized and approximation algorithms for maximum internal spanning tree. Inf Comput 252:187–200
Li X, Zhu D (2014) Approximating the maximum internal spanning tree problem via a maximum path-cycle cover. In: International symposium on algorithms and computation, Springer, pp 467–478
Li X, Feng H, Jiang H et al (2018) Solving the maximum internal spanning tree problem on interval graphs in polynomial time. Theor Comput Sci 734:32–37
Li X, Zhu D, Wang L (2021) A 4/3-approximation algorithm for the maximum internal spanning tree problem. J Comput Syst Sci 118:131–140
Lin R, Olariu S, Pruesse G (1995) An optimal path cover algorithm for cographs. Comput Math Appl 30(8):75–83
Lu HI, Ravi R (1992) The power of local optimization: Approximation algorithms for maximum-leaf spanning tree. In: Proceedings of the annual allerton conference on communication control and computing, University of Illinois, pp 533–533
Müller H (1996) Hamiltonian circuits in chordal bipartite graphs. Discret Math 156(1–3):291–298
Pak-Ken W (1999) Optimal path cover problem on block graphs. Theore Comput Sci 225(1–2):163–169
Prieto E, Sloper C (2003) Either/or: Using vertex cover structure in designing fpt-algorithms—the case of k-internal spanning tree. In: Workshop on algorithms and data structures, Springer, pp 474–483
Salamon G (2009) Approximating the maximum internal spanning tree problem. Theor Comput Sci 410(50):5273–5284
Salamon G (2010) Degree-based spanning tree optimization. PhD Thesis
Salamon G, Wiener G (2008) On finding spanning trees with few leaves. Inf Process Lett 105(5):164–169
Seinsche D (1974) On a property of the class of n-colorable graphs. J Comb Theory Series B 16(2):191–193
Spinrad J, Brandstädt A, Stewart L (1987) Bipartite permutation graphs. Discret Appl Math 18(3):279–292
Srikant R, Sundaram R, Singh KS et al (1993) Optimal path cover problem on block graphs and bipartite permutation graphs. Theor Comput Sci 115(2):351–357