Degree-Constrained Spanning and Steiner Trees on Structured Graph Classes

C. Sanu[1], R. Mahendra Kumar[1], A. Mohanapriya[2], and N. Sadagopan[1]

[1] Indian Institute of Information Technology, Design and Manufacturing, Kancheepuram, India
[2] Institute of Mathematical Sciences, Chennai, India.
{cs24d0007,coe18d004,sadagopan}@iiitdm.ac.in, mohana@imsc.res.in

Abstract. A tree is called k-ary if the degree of every vertex is at most $k+1$. Given a connected graph G, the *k-ary Spanning Tree* problem asks whether G admits a spanning tree that is k-ary. For $k = 1$, the problem coincides with the Hamiltonian Path problem; however, the complexity landscape for $k \geq 2$ is significantly more intricate.

We investigate the classical and parameterized complexity of the k-ary Spanning Tree problem for all fixed $k \geq 2$ on several structured graph classes, including chordal bipartite graphs, strongly chordal split graphs, and subclasses of split graphs. We prove that the problem is NP-complete on chordal bipartite graphs and on strongly chordal split graphs. Moreover, under the Exponential Time Hypothesis (ETH), these hardness results rule out subexponential-time $2^{o(n)}$ algorithms on these classes. In contrast, the problem is polynomial-time solvable on bipartite chain graphs, which is a subclass of chordal bipartite graphs. For $K_{1,r}$-free split graphs, we establish a sharp dichotomy: the problem is polynomial-time solvable when $r \leq k + 2$ and NP-complete when $r \geq k + 4$, leaving the boundary case $r = k + 3$ open.

We further study the *k-ary Steiner Tree* problem and show that, for all graph classes considered, it exhibits the same complexity behavior as the spanning variant. Finally, we express both problems in counting monadic second-order logic (CMSO), and by Courcelle's theorem obtain fixed-parameter tractable algorithms parameterized by the treewidth of the input graph.

Keywords: k-ary spanning (Steiner) tree · chordal bipartite graphs · strongly chordal split graphs · $K_{1,r}$-free split graphs

1 Introduction

The Hamiltonian Path problem (HP) and the Steiner Tree problem (StT) are two classical and extensively studied problems in graph theory. Since their introduction, both problems have attracted significant attention due to their deep theoretical importance and wide range of practical applications in areas such as network design, circuit layout, and communication systems. In this paper,

we investigate natural generalizations of these two problems through degree-constrained spanning structures.

There are two primary motivations for studying the k-ary spanning tree problem. The classical spanning tree problem is solvable in polynomial time; however, it imposes no restriction on vertex degrees in the resulting tree. In a spanning tree of an n-vertex graph, a vertex may have degree ranging from 1 to $n-1$. In practical applications such as communication or transportation networks, such unrestricted degrees may lead to undesirable configurations, including bottlenecks, single points of failure, and reduced reliability. To address these concerns, it is often desirable to construct degree-constrained spanning trees in which the degree of every vertex is bounded by a fixed integer.

A tree T is said to be a k-ary tree if the degree of every vertex in T is at most $k+1$. For a connected graph G, the k-ary Spanning Tree problem (k-ary SpT), for $k \geq 2$, asks whether G contains a spanning tree in which every vertex has degree at most $k+1$.

The second motivation stems from the well-known Hamiltonian Path problem. For a connected graph G, a Hamiltonian path is a path that spans $V(G)$. The Hamiltonian Path problem is one of the most fundamental NP-complete problems in combinatorial optimization. It is known [8] that HP is NP-complete on bipartite graphs, chordal bipartite graphs, strongly chordal graphs, and split graphs. A refined complexity dichotomy for split graphs is reported in [10]: the problem is NP-complete for $K_{1,5}$-free split graphs, whereas it is polynomial-time solvable for $K_{1,4}$-free split graphs. On the other hand, HP is polynomial-time solvable on interval graphs [2], bipartite chain graphs, bipartite permutation graphs [3], and distance-hereditary graphs [6].

The Hamiltonian Path problem can be equivalently reformulated as the *unary spanning tree problem* (or 1-ary spanning tree problem): given a connected graph G, does there exist a spanning tree in which every vertex has degree at most 2? Thus, HP corresponds exactly to the special case $k = 1$ of the k-ary spanning tree problem.

This relationship provides a natural hierarchy (see Figure 1).

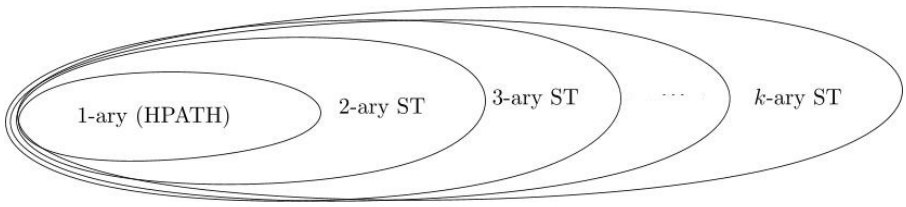

Fig. 1. A comparison about yes-instances of k-ary spanning tree, $\forall k \geq 1$.

Observation 1 *For $q \geq 1$, a q-ary spanning tree is also a k-ary spanning tree for any $k \geq q$.*

From Observation 1, if an instance $\mathcal{I}$ is a yes-instance of HP, then it is also a yes-instance of k-ary SpT for all $k \geq 1$. However, the converse does not hold: a no-instance of HP may become a yes-instance of k-ary SpT for sufficiently large k.

For example, consider a caterpillar graph in which each vertex of the backbone path has k pendant vertices attached. Such a graph does not admit a Hamiltonian path, but it admits a q-ary spanning tree for all $q \geq k+1$. For $q < k+1$, it remains a no-instance. Therefore, the boundary between yes-instances and no-instances varies with k, and the computational complexity of the k-ary spanning tree problem must be analyzed separately for each $k \geq 2$.

We next turn to the Steiner Tree problem. Given a connected graph $G = (V, E)$ and a subset of vertices $R \subseteq V(G)$, the Steiner Tree problem asks for a minimum-cardinality set $S \subseteq V(G) \setminus R$ such that the subgraph induced by $R \cup S$ is connected. The problem is NP-complete in split graphs and chordal bipartite graphs [9]. Further, it is known that the problem is polynomial-time solvable on strongly chordal graphs, bipartite distance-hereditary graphs, interval graphs, and convex bipartite graphs [5,11].

In a classical Steiner tree, vertices may have arbitrary degree. Motivated by practical considerations similar to those for spanning trees, we introduce a degree-constrained version of the problem. The *k-ary Steiner Tree problem* (k-ary StT), for $k \geq 2$, asks the following: given a connected graph G and a terminal set $R \subseteq V(G)$, find a minimum-cardinality set $S \subseteq V(G) \setminus R$ such that there exists a k-ary tree spanning $R \cup S$.

Interestingly, while the classical Steiner Tree problem is polynomial-time solvable on strongly chordal graphs, the k-ary Steiner Tree problem becomes NP-complete on strongly chordal split graphs for every $k \geq 2$. This contrast highlights the significant impact of degree constraints on computational complexity.

In this paper, we undertake a systematic study of the computational complexity of the k-ary spanning tree problem and the k-ary Steiner Tree problem for each $k \geq 2$, across various graph classes. Our results reveal a rich and nuanced complexity landscape, demonstrating that even mild degree constraints fundamentally alter the tractability of classical connectivity problems.

Our Contributions. In this paper, we investigate the computational complexity of the k-ary Spanning Tree problem (k-ary SpT) and the k-ary Steiner Tree problem (k-ary StT). For each $k \geq 2$, we establish the following results:

1. We prove that k-ary SpT and k-ary StT are NP-complete on chordal bipartite graphs (Section 2.1).
2. We show that k-ary SpT and k-ary StT are polynomial-time solvable on bipartite chain graphs (Section 2.2).
3. We prove that both k-ary SpT and k-ary StT are NP-complete on strongly chordal split graphs (Section 3.1).
4. We further show that k-ary SpT is NP-complete on $K_{1,r}$-free split graphs when $r \geq k+4$, and polynomial-time solvable on $K_{1,r}$-free split graphs when $r \leq k+2$ (Section 3.2).

5. For the k-ary StT on $K_{1,r}$-free split graphs, we prove that:
   - if $r \geq 5$ and $k \leq r-1$, then the problem is NP-complete;
   - if $r \geq 5$ and $k \geq r$, then the problem is polynomial-time solvable;
   - if $r \leq 4$, then the problem is polynomial-time solvable for all $k \geq 2$.

   (Section 3.2)
6. We establish the fixed-parameter tractability of the k-ary Spanning Tree and k-ary Steiner Tree problems parameterized by treewidth. Specifically, we present MSO_2 formulations for both problems and, via Courcelle's Theorem, obtain algorithms running in time $f(\mathrm{tw}(G)) \cdot |V(G)|$ for every fixed integer $k \geq 1$. Consequently, both problems are solvable in linear time on graph classes of bounded treewidth.(Section 4)

Graph Class	**Condition**	**Complexity**
General Graphs	$k \geq 1$ fixed	NP-complete
Chordal Bipartite	$k \geq 1$ fixed	NP-complete
Bipartite Chain Graphs	$k \geq 1$ fixed	Polynomial-time
Strongly Chordal Split	$k \geq 1$ fixed	NP-complete
$K_{1,r}$-free Split	$r \leq k+2$	Polynomial-time
	$r = k+3$	Open
	$r \geq k+4$	NP-complete
Bounded Treewidth	k fixed	FPT in $\mathrm{tw}(G)$

Table 1. Complexity of the k-ary Spanning Tree and k-ary Steiner Tree problems (for fixed k).

The Table 1 summarizes the landscape and reveals a sharp threshold phenomenon on $K_{1,r}$-free split graphs: the problems are polynomial-time solvable when $r \leq k+2$, NP-complete when $r \geq k+4$, and the intermediate case $r = k+3$ remains open.

Preliminaries In this paper, we consider simple, connected, undirected graphs. For a graph G, let $V(G)$ and $E(G)$ denote its vertex and edge sets. For $u \in V(G)$, let $N_G(u) = \{v \mid \{u,v\} \in E(G)\}$ and $d_G(u) = |N_G(u)|$. The maximum degree of G is $\Delta(G)$. A vertex of degree one is called a pendant vertex. A graph H is an *induced subgraph* of G if for all $u, v \in V(H)$, $\{u,v\} \in E(H)$ if and only if $\{u,v\} \in E(G)$.

A tree T is *k-ary* if $\Delta(T) \leq k+1$. A vertex $v \in V(T)$ is *saturated* if $d_T(v) = k+1$, and has *space* otherwise. A graph is *chordal* if every cycle of length at least 4 has a chord. A bipartite graph is *chordal bipartite* if every cycle of length at least 6 has a chord. A graph is *strongly chordal* if it is chordal and every even cycle of length at least 6 has an odd chord, i.e., an edge connecting two vertices at odd distance at least 3 on the cycle. A bipartite graph $G(X,Y)$ is called a *chain graph* if the vertices can be ordered as $X = \{x_1, x_2, \ldots, x_{|X|}\}$

and $Y = \{y_1, y_2, \ldots, y_{|Y|}\}$ such that $N(x_1) \subseteq N(x_2) \subseteq \cdots \subseteq N(x_{|X|})$ and $N(y_1) \subseteq N(y_2) \subseteq \cdots \subseteq N(y_{|Y|})$. A graph G is a *split graph* if its vertex set can be partitioned into a clique K and an independent set I. For a vertex $u \in K$, let $N_G^I(u) = N_G(u) \cap I$ and $d_G^I(u) = |N_G^I(u)|$. Define $\Delta_G^I = \max_{u \in K} d_G^I(u)$, and for a set $S \subseteq V(G)$, let $N_G^I(S) = \bigcup_{x \in S}(N_G(x) \cap I)$. A split graph is said to be $K_{1,r}$*-free* if it does not contain an induced subgraph isomorphic to $K_{1,r}$. A *comb* is a tree $T = (A, F)$ where $A = \{a_1, a_2, \ldots, a_{2p}\}$ and $F = \{\{a_i, a_{p+i}\} \mid 1 \leq i \leq p\} \cup \{\{a_i, a_{i+1}\} \mid 1 \leq i < p\}$. The path $a_1 a_2 \cdots a_p$ is called the *backbone* of the comb, and the vertices $a_{p+1}, \ldots, a_{2p}$ are its *teeth*. A *caterpillar graph* is a tree such that removing all pendant vertices results in a chordless path.

Proofs of statements marked with (★) are deferred to the Appendix.

2 Bipartite graphs

In this section, we investigate the computational complexity of the k-ary Spanning Tree (SpT) and k-ary Steiner Tree (StT) problems on bipartite graphs. We first establish NP-completeness results on chordal bipartite graphs, and then turn to bipartite chain graphs, a proper subclass characterized by nested neighborhood ordering, and show that the additional structural restrictions allow polynomial-time solvability.

2.1 Chordal bipartite graphs

We leverage this hardness result to show that both the k-ary Spanning Tree and the k-ary Steiner Tree problems remain NP-complete on this class. Our reductions are from HPATH on chordal bipartite graphs [8] and preserve chordal bipartiteness.

Theorem 1. (★) *For chordal bipartite graphs and any $k \geq 2$, the k-ary SpT is NP-complete.*

Theorem 2. (★) *For chordal bipartite graphs and any $k \geq 2$, the k-ary StT is NP-complete.*

Theorem 3 (ETH Lower Bound). *Assuming the Exponential Time Hypothesis (ETH), for every fixed $k \geq 2$ there is no algorithm that solves the k-ary Spanning Tree or the k-ary Steiner Tree problem on chordal bipartite graphs in time $2^{o(n)}$, where $n = |V(G)|$.*

Proof. Both Theorems 1 and 2 are obtained by polynomial-time reductions from HPATH on chordal bipartite graphs. The construction increases the number of vertices only linearly. Hence, any $2^{o(n)}$-time algorithm for k-ary SpT or k-ary StT on chordal bipartite graphs would imply a $2^{o(n)}$-time algorithm for HPATH on chordal bipartite graphs.

Since HPATH does not admit a subexponential-time algorithm under ETH, the result follows. □

2.2 Bipartite Chain Graphs

Having established that the k-ary SpT problem is NP-complete on chordal bipartite graphs, it is natural to examine whether polynomial-time solvability can be recovered in proper subclasses. In this section, we study bipartite chain graphs, a well-known subclass of chordal bipartite graphs characterized by nested neighborhood structure.

We first show that the binary spanning tree problem admits a polynomial-time algorithm on bipartite chain graphs. We then extend this result to the k-ary Spanning Tree problem for every fixed $k \geq 3$.

We fix the following notation. Let G be a bipartite chain graph with bipartition (A, B), where $A = \{x_1, \ldots, x_m\}$ and $B = \{y_1, \ldots, y_n\}$, and $E(G) \subseteq \{\{x, y\} \mid x \in A, y \in B\}$. Let $A = A_1 \cup A_2$ and $B = B_1 \cup B_2$, where $A_1 = \{x_1, \ldots, x_i\}$, $A_2 = \{x_{i+1}, \ldots, x_m\}$, $B_1 = \{y_1, \ldots, y_j\}$, and $B_2 = \{y_{j+1}, \ldots, y_n\}$, such that (A_1, B_1) induces a *base biclique*. The base biclique (A_1, B_1) is defined as a maximal complete bipartite subgraph $K_{i,j}$ of G for which $|i - j|$ is minimum among all maximal bicliques of G.To determine (A_1, B_1), order the vertices of A in non-increasing order of degree. For each $t \in \{1, \ldots, m\}$, the vertices $\{x_1, \ldots, x_t\}$ together with their common neighborhood induce a biclique $K_{t,d_G(x_t)}$. Among these bicliques, select $K_{i,j}$ minimizing $|i - j|$; this biclique is the base biclique of G.

Lemma 1. *Let G be a bipartite chain graph. Then, $\forall x \in A_2, \exists y_k \in B_1$ such that $y_k \notin N_G(x)$ and $\forall y \in B_2, \exists x_k \in A_1$ such that $x_k \notin N_G(y)$.*

Proof. Suppose there exists x in A_2 such that for all y_k in B_1, $\{x, y_k\} \in E(G)$. Then $(A_1 \cup \{x\}, B_1)$ is the base biclique, contradicting the fact that (A_1, B_1) is the base biclique. A similar argument is true for $y \in B_2$. Therefore, the lemma follows. □

Lemma 2. (★) *Let G be a bipartite chain graph. Then*

$$\forall x \in A_2,\ N_G(x) \subseteq B_1 \quad \textit{and} \quad \forall y \in B_2,\ N_G(y) \subseteq A_1.$$

We shall now see the *Nested Neighborhood Ordering (NNO)* among the vertices of G. Let G be a bipartite chain graph with (A_1, B_1) being the base biclique. Let $A_2 = (u_1, u_2, ..., u_p)$ and $B_2 = (v_1, v_2, ..., v_q)$ are the orderings of vertices. If $d_G(u_1) \leq d_G(u_2) \leq d_G(u_3) \leq \ldots \leq d_G(u_p)$, then $N(u_1) \subseteq N(u_2) \subseteq N(u_3) \subseteq \ldots \subseteq N(u_p)$. Further, if $d_G(v_1) \leq d_G(v_2) \leq d_G(v_3) \leq \ldots \leq d_G(v_q)$, then $N(v_1) \subseteq N(v_2) \subseteq N(v_3) \subseteq \ldots \subseteq N(v_q)$. We refer to the above ordering of vertices as *Nested Neighbourhood Ordering (NNO)* of G. We shall arrange the vertices in A_2 in non-decreasing order of their degrees so that we can work with *NNO* of G.

Theorem 4. *Let $G(A_1 \cup A_2, B_1 \cup B_2)$ be a bipartite chain graph. Then G admits a binary spanning tree if and only if the following conditions hold:*

(i) (***Degree condition***) *The vertices of* A_2 *admit an ordering* $(u_1, \ldots, u_p)$ *such that*

$$d_G(u_g) \geq \left\lceil \frac{g}{2} \right\rceil \quad \textit{for all } 1 \leq g \leq p,$$

and the vertices of B_2 *admit an ordering* $(v_1, \ldots, v_q)$ *such that*

$$d_G(v_h) \geq \left\lceil \frac{h}{2} \right\rceil \quad \textit{for all } 1 \leq h \leq q.$$

(ii) (***Balance condition***) *Let*

$$A_3 = A_1 \setminus \{x_1, \ldots, x_{\lceil q/2 \rceil}\}, \qquad B_3 = B_1 \setminus \{y_1, \ldots, y_{\lceil p/2 \rceil}\}.$$

Then one of the following holds:

- *If* $|A_2|$ *is even, then*

$$|A_3| \leq 3|B_3| - 1,$$

otherwise

$$|A_3| \leq 2|B_3|.$$

- *Symmetrically, if* $|B_2|$ *is even, then*

$$|B_3| \leq 3|A_3| - 1,$$

otherwise

$$|B_3| \leq 2|A_3|.$$

Proof. Let $A_3 = A_1 \setminus \{x_1, \ldots, x_{\lceil q/2 \rceil}\}, \qquad B_3 = B_1 \setminus \{y_1, \ldots, y_{\lceil p/2 \rceil}\}$. We prove both directions.

($\Rightarrow$) Assume that G admits a binary spanning tree T.

(i) Degree condition. Suppose, to the contrary, that there exists a first vertex $u_g \in A_2$ in the chain ordering such that $d_G(u_g) < \left\lceil \frac{g}{2} \right\rceil$, while for every $k < g$, $d_G(u_k) \geq \left\lceil \frac{k}{2} \right\rceil$.

Let $S = \{u_1, \ldots, u_{g-1}\}$. Since G is a chain graph (NNO property), $N(u_1) \subseteq N(u_2) \subseteq \cdots \subseteq N(u_g)$. Hence, $|N_G(S)| = d_G(u_{g-1}) = \left\lceil \frac{g-1}{2} \right\rceil$. Now, $|S| = g - 1 = 2\left\lceil \frac{g-1}{2} \right\rceil$ or $2\left\lceil \frac{g-1}{2} \right\rceil - 1$.Thus, $|S| \geq 2|N_G(S)| - 1$.

In any binary tree, every internal vertex has degree exactly three. In a bipartite binary tree with bipartition (X, Y), a simple degree counting argument implies $|X| \leq 2|Y|$ and $|Y| \leq 2|X|$. However, in the induced subgraph on $S \cup \{u_g\} \cup N_G(S)$, the number of vertices on the A-side strictly exceeds twice the number on the B-side, since $d_G(u_g) \leq |N_G(S)| - 1$. Hence no binary tree can span this induced subgraph, contradicting the existence of T. Thus the degree condition holds. The argument for B_2 is symmetric.

(ii) Balance condition. Consider the induced subgraph on $A_3 \cup B_3$. In any binary tree T' spanning these vertices, every internal vertex has degree three. Let $a = |A_3|$ and $b = |B_3|$. Degree counting in bipartite trees yields $a \leq 3b - 1$ if $|A_2|$ is even, and $a \leq 2b$ otherwise, with the symmetric bounds when

roles of A and B are reversed. If these inequalities fail, the degree constraints of a binary tree cannot be satisfied, contradicting the existence of T. Hence the balance condition holds.

($\Leftarrow$) Assume that conditions (i) and (ii) hold. We construct a binary spanning tree T of G.

Step 1: Construction on A_2.

Let $V(T_1) = A_2 \cup \{y_1, \ldots, y_{\lceil p/2 \rceil}\}$. Using the degree condition and nested neighborhoods, construct the alternating path $P_1 = (u_1, y_1, u_3, y_2, u_5, \ldots)$. For each even index $2g$, add the edge $u_{2g}y_g$, which exists by the degree assumption and nesting property.

The resulting graph T_1' is connected, acyclic, and every vertex has degree at most three. Thus T_1' is a binary tree spanning T_1.

Step 2: Construction on B_2.

Symmetrically, let $V(T_2) = B_2 \cup \{x_1, \ldots, x_{\lceil q/2 \rceil}\}$. Construct the alternating path $P_2 = (v_1, x_1, v_3, x_2, v_5, \ldots)$, and attach each even-indexed v_{2h} to x_h. The resulting tree T_2' is binary.

Step 3: Construction on $A_3 \cup B_3$.

Since $A_3 \cup B_3$ induces a complete bipartite graph, and the balance condition holds, we can construct a binary tree T_3' on these vertices by forming an alternating path and attaching remaining vertices so that no degree exceeds three. The inequality guarantees feasibility of this construction.

Step 4: Connecting the components.

By construction, there exist vertices in T_1', T_2', and T_3' with degree at most two. Using two suitable cross edges (permitted since the graph is bipartite chain), we connect T_1', T_2', and T_3' without creating cycles and without exceeding degree three.

Let $T = T_1' \cup T_2' \cup T_3'$ together with these connecting edges. Notice that T is connected, T is acyclic, every vertex has degree at most three, and $V(T) = V(G)$. Thus T is a binary spanning tree of G. □

Theorem 5. *Let G be a bipartite chain graph. The Binary Spanning Tree problem is polynomial-time solvable.*

Proof. By Theorem 4, G admits a binary spanning tree if and only if it satisfies degree and balance conditions. The proof of Theorem 4 is constructive and provides a procedure that either constructs a binary spanning tree or correctly reports that none exists.

All required checks (degree ordering, and size inequalities) can be verified in polynomial time. Hence the Binary Spanning Tree problem is polynomial-time solvable on bipartite chain graphs.

The structural characterization given in Theorem 4 naturally extends to higher arity.

Theorem 6. *Let G be a bipartite chain graph. For every fixed $k \geq 3$, the k-ary Spanning Tree problem is polynomial-time solvable.*

Proof. The construction in Theorem 4 relies on the nested neighborhood ordering (NNO) and a degree–balance condition ensuring that no vertex exceeds the allowed maximum degree. For general k, the same approach applies by replacing the degree bound three with $k + 1$ and adjusting the balance inequalities accordingly. Since the NNO can be computed in polynomial time and all required degree checks and constructions involve only linear scans and edge insertions, the resulting algorithm runs in polynomial time for every fixed k. □

The hereditary nature of bipartite chain graphs and the preservation of the nested neighborhood ordering under vertex deletion allow us to reduce the k-ary Steiner Tree problem to the k-ary Spanning Tree problem. Consequently, the structural characterization obtained above extends to the Steiner setting.

Observation 2 *Let G be a bipartite chain graph with nested neighborhood ordering $A = (u_1, \ldots, u_p)$ and $B = (v_1, \ldots, v_q)$. For any vertex $x \in V(G)$, the graph $G - x$ also admits a nested neighborhood ordering obtained by deleting x from the corresponding sequence.*

Theorem 7. *(★) Let G be a bipartite chain graph. For every fixed $k \geq 2$, the k-ary Steiner Tree problem is polynomial-time solvable.*

3 Split graphs

Having analyzed bipartite graph classes, we now turn to split graphs, a fundamental subclass of chordal graphs whose vertex set partitions into a clique and an independent set. Unlike bipartite graphs, split graphs permit dense structure on the clique side while maintaining controlled sparsity on the independent side, leading to subtler complexity behavior.

We examine how additional structural constraints—namely strong chordality and $K_{1,r}$-freeness—affect the complexity of the k-ary Spanning Tree and k-ary Steiner Tree problems. We prove NP-completeness on strongly chordal split graphs and establish a sharp dichotomy for $K_{1,r}$-free split graphs.

3.1 Strongly chordal split graphs

Strongly chordal split graphs form a natural subclass of split graphs obtained by imposing the strong chordality condition, namely that every even cycle of length at least six has an odd chord. Although this restriction limits the induced cycle structure, it does not preclude computational hardness. In fact, Hamiltonian Path remains NP-complete on strongly chordal split graphs, even with two pendant vertices [8]. Leveraging this result, we prove that both problems are NP-complete on this class. Our reductions preserve the split partition as well as strong chordality.

Theorem 8. *For strongly chordal split graphs and any $k \geq 2$, the k-ary Spanning Tree problem is NP-complete.*

Proof. Membership in NP is immediate, since a spanning tree can be verified in polynomial time and the degree bound can be checked in linear time. We reduce from HPATH on strongly chordal split graphs with two pendant vertices, which is NP-complete [8].

Construction. Let G be a strongly chordal split graph with split partition $(K(G), I(G))$ and pendant vertices $u_1, u_n \in I(G)$.

We construct a graph H as follows.

Vertices. $V(H) = V(G) \cup \{w_{ij} \mid v_i \in K(G),\ 1 \leq j \leq k-1\} \cup \{x_{ij} \mid u_i \in I(G),\ 1 \leq j \leq k-1\} \cup \{y^i_{jl} \mid u_i \in I(G),\ 1 \leq j \leq k-1,\ 1 \leq l \leq k\} \cup \{d_1, d_2\} \cup \{p_i \mid 1 \leq i \leq 2k\}$.

Edges. The edge set contains: all edges of G; $\{v_i, w_{ij}\}$ for $v_i \in K(G)$; $\{u_i, x_{ij}\}$ for $u_i \in I(G)$; $\{x_{ij}, v_t\}$ for all $u_i \in I(G)$ and $v_t \in K(G)$; $\{x_{ij}, y^i_{jl}\}$; $\{u_1, d_1\}, \{u_n, d_2\}$; $\{d_1, p_i\}$ for $1 \leq i \leq k$ and $\{d_2, p_i\}$ for $k+1 \leq i \leq 2k$; $\{d_1, d_2\}$; and all edges among the vertices x_{ij} (making them part of the clique).

Structure preservation. The clique side of H is $K(H) = K(G) \cup \{x_{ij}\} \cup \{d_1, d_2\}$, and the independent side is $I(H) = I(G) \cup \{w_{ij}\} \cup \{y^i_{jl}\} \cup \{p_i\}$. Thus H is a split graph.

The added vertices are either pendant vertices or are attached entirely within the clique side. No new induced cycle of length at least four is introduced. Since strong chordality is preserved under such local extensions, H remains a strongly chordal split graph. The construction clearly runs in polynomial time.

Correctness. We prove that G has a Hamiltonian path if and only if H has a k-ary spanning tree.

($\Rightarrow$) Let $P = (u_1, v_1, \ldots, v_m, u_n)$ be a Hamiltonian path of G. We construct a tree T as follows (*i*) Including all edges of P; (*ii*) Attaching each w_{ij} to v_i; (*iii*) Attaching each x_{ij} to u_i; (*iv*) Attaching each y^i_{jl} to x_{ij}; (*v*) Attaching d_1 to u_1 and d_2 to u_n; (*vi*) Attaching the p_i vertices to d_1 and d_2 respectively.

The resulting graph is connected, acyclic, and spans $V(H)$. Each original vertex gains exactly $k-1$ pendant neighbors. Thus its degree in T is at most $k+1$. All new vertices are leaves or have degree at most $k+1$. Hence T is a k-ary spanning tree of H.

($\Leftarrow$) Let T be a k-ary spanning tree of H. All vertices w_{ij}, y^i_{jl}, and p_i are pendant in H. Thus their incident edges are forced in T.

Hence the following are true (*i*) Each v_i already has $k-1$ incident edges to w_{ij}; (*ii*) Each u_i already has $k-1$ incident edges to x_{ij}; (*iii*) Each x_{ij} already has k incident edges to y^i_{jl}. Therefore, the following is true (*i*) Each x_{ij} can have at most one additional neighbor in $K(H)$; (*ii*) Each $v_i \in K(G)$ can have at most two neighbors in $K(H)$; (*iii*) d_1 and d_2 can each have at most one additional neighbor. Using the fact that $|K(G)| = |I(G)| - 1$ [8], connectivity forces the vertices of $K(G)$ and $I(G)$ to form a single backbone path in T. This backbone is a spanning path of G, hence a Hamiltonian path. Thus G has a Hamiltonian path if and only if H has a k-ary spanning tree. Therefore, the problem is NP-complete. □

Theorem 9. (★) *For strongly chordal split graphs and any $k \geq 2$, the k-ary StT is NP-complete.*

Theorem 10 (ETH Lower Bound). *Assuming the Exponential Time Hypothesis (ETH), for every fixed $k \geq 2$ there is no algorithm that solves the k-ary Spanning Tree or the k-ary Steiner Tree problem on strongly chordal split graphs in time $2^{o(n)}$, where $n = |V(G)|$.*

3.2 $K_{1,r}$-free split graphs

In this section, we establish a sharp complexity dichotomy for the k-ary Spanning Tree (SpT) and k-ary Steiner Tree (StT) problems on $K_{1,r}$-free split graphs. Recall that in a split graph $G(K \cup I, E)$, the graph is $K_{1,r}$-free if and only if $\Delta_G^I \leq r - 1$.

Theorem 11. *(★) Let G be a $K_{1,r}$-free split graph.*

- *If $r \geq k + 4$, then the k-ary spanning tree problem is NP-complete.*
- *If $r \leq k + 2$, then the problem is polynomial-time solvable.*
- *The complexity of the case $r = k + 3$ remains open.*

We obtain analogous results for the k-ary Steiner tree problem on $K_{1,r}$-free split graphs.

4 CMSO Formulation and FPT Algorithms Parameterized by Treewidth

In this section, we present counting monadic second-order logic (CMSO) formulations for the *k-ary Spanning Tree* and *k-ary Steiner Tree* problems (★) and derive fixed-parameter tractability results when parameterized by the treewidth of the input graph.

k-ary Spanning Tree. Let $G = (V, E)$ be a connected graph. A tree is called *k-ary* if the degree of every vertex is at most $k + 1$. The k-ary Spanning Tree problem asks whether G admits a spanning tree satisfying this degree constraint.

We express this problem in MSO_2. We use the standard MSO_2 incidence predicate $\mathsf{inc}(v, e)$, which holds if and only if vertex v is an endpoint of edge e. For convenience, we define the auxiliary predicate

$$\mathsf{Ends}(u, v, e) := \mathsf{inc}(u, e) \wedge \mathsf{inc}(v, e) \wedge u \neq v,$$

which denotes that edge e connects vertices u and v.

We existentially quantify an edge set $T \subseteq E$, intended to represent the spanning tree, and require that T is spanning, connected, acyclic, and respects the degree bound.

Formally, the MSO_2 sentence is

$$\exists T \subseteq E \, \big(\mathsf{Spanning}(T) \ \wedge \ \mathsf{Connected}(T) \ \wedge \ \mathsf{Acyclic}(T) \ \wedge \ \mathsf{DegreeBound}(T)\big),$$

where the predicates are defined as follows.

Spanning: $\mathsf{Spanning}(T) := \forall v \in V \ \exists e \in T : \ \mathsf{inc}(v, e)$.

Connectivity: $\mathsf{Connected}(T) := \forall X \subset V \ \big(X \neq \emptyset \ \rightarrow \ \exists e \in T \ \exists u \in X \ \exists v \in V \setminus X : \mathsf{Ends}(u, v, e)\big)$.

Acyclicity: $\mathsf{Acyclic}(T) := \neg\exists C \subseteq V \ \Big(\forall v \in C : \ \exists u, w \in C\big(u \neq w \wedge \exists e_1, e_2 \in T : \mathsf{Ends}(v, u, e_1) \wedge \mathsf{Ends}(v, w, e_2)\big)\Big)$.

Degree bound: $\mathsf{DegreeBound}(T) := \forall v \in V : \ \neg\exists u_1, \ldots, u_{k+2} \in V\Big(\bigwedge_{i \neq j} u_i \neq u_j \ \wedge \ \bigwedge_{i=1}^{k+2} \exists e_i \in T : \mathsf{Ends}(v, u_i, e_i)\Big)$.

Theorem 12. *(★) For every fixed integer $k \geq 1$, the k-ary Spanning Tree problem and the k-ary Steiner Tree problem are fixed-parameter tractable when parameterized by the treewidth of the input graph.*

Conclusions and Directions for further research. We provided a fine-grained complexity classification of the k-ary Spanning Tree and k-ary Steiner Tree problems on structured bipartite and split graph classes. While both problems are NP-complete on chordal bipartite graphs for every fixed $k \geq 2$, they admit polynomial-time algorithms on bipartite chain graphs due to their nested neighborhood structure.

For $K_{1,r}$-free split graphs, we established a sharp threshold: the k-ary spanning tree problem is polynomial-time solvable when $r \leq k + 2$ and NP-complete when $r \geq k + 4$, leaving the boundary case $r = k + 3$ open. This delineates a precise structural frontier between tractability and intractability for bounded-degree spanning structures. Resolving the remaining gap remains an interesting direction for future research.

References

1. Takanori Akiyama, Takao Nishizeki, and Nobuji Saito. NP-completeness of the Hamiltonian cycle problem for bipartite graphs. *Journal of Information processing*, 3(2):73–76, 1980.
2. Alan A Bertossi and Maurizio A Bonuccelli. Hamiltonian circuits in interval graph generalizations. *Information Processing Letters*, 23(4):195–200, 1986.
3. Andreas Brandstädt and Dieter Kratsch. *On the restriction of some NP-complete graph problems to permutation graphs*. Springer, 1985.
4. Andreas Brandstädt and Raffaele Mosca. *On the Structure and Stability Number of P_5-and Co-chair-free Graphs*. Univ., Fachbereich Informatik, 2001.
5. Alessandro D'Atri and Marina Moscarini. Distance-hereditary graphs, steiner trees, and connected domination. *SIAM Journal on Computing*, 17(3):521–538, 1988.
6. Ruo-Wei Hung and Maw-Shang Chang. Linear-time algorithms for the Hamiltonian problems on distance-hereditary graphs. *Theoretical Computer Science*, 341(1-3):411–440, 2005.
7. Richard M Karp. *Reducibility among combinatorial problems*. Springer, 2010.
8. Haiko Müller. Hamiltonian circuits in chordal bipartite graphs. *Discrete Mathematics*, 156(1-3):291–298, 1996.
9. Haiko Müller and Andreas Brandstädt. The NP-completeness of steiner tree and dominating set for chordal bipartite graphs. *Theoretical Computer Science*, 53(2):257–265, 1987.

10. P Renjith and N Sadagopan. Hamiltonian Path in $K_{1,r}$-free Split Graphs-A Dichotomy. In *Conference on Algorithms and Discrete Applied Mathematics*, pages 30–44. Springer, 2018.
11. Kevin White, Martin Farber, and William Pulleyblank. Steiner trees, connected domination and strongly chordal graphs. *Networks*, 15(1):109–124, 1985.

5 Appendix

Proof of Theorem 1

Proof. It is easy to see that k-ary SpT on chordal bipartite graphs and any $k \geq 2$ is in NP as the given certificate can be verified in deterministic polynomial time. We present a deterministic polynomial-time reduction that reduces an instance of HPATH on chordal bipartite graph G to the corresponding instance of the k-ary SpT problem on chordal bipartite graph G' as follows: We shall now define $V(G') = V(G) \cup \{u_i \mid 1 \leq i < k, u \in V(G)\}$. The edge set $E(G') = E(G) \cup \{\{u, u_i\} \mid u \in V(G), 1 \leq i < k\}$. Observe that, u_i, $1 \leq i < k$ are pendant vertices in G'. Thus G' remains a chordal bipartite graph. We claim that G is a yes-instance of HPATH if and only if G' is a yes-instance of the k-ary SpT.

($\Rightarrow$) Suppose that there exists a Hamiltonian path P in G. Let the path P be $(v_1, v_2, v_3, \ldots, v_n)$. We construct a k-ary spanning tree T as follows: For each vertex $u \in P$, attach the pendant vertices u_i, $1 \leq i < k$. Observe that T is a tree that spanning $V(G')$ and degree of each vertex in T is at most $k + 1$. Thus, we obtain a k-ary spanning tree T of G'.

($\Leftarrow$) Suppose that there exists a k-ary spanning tree T of G'. Observe that all u_i's, $1 \leq i < k$ are pendant vertices in G' and hence the edges $\{\{u, u_i\}, u \in V(G), 1 \leq i < k\}$ are mandatory in T. We claim that removing the pendant vertices from G' yields a Hamiltonian path P in G. Suppose that P is not a Hamiltonian path in G. Let H be the graph obtained after removing the pendant vertices. Observe that the degree of each vertex in H is at most 2. A connected graph that can be constructed from this degree sequence is a path graph. Note that H has to span the entire $V(G)$. If H is not a spanning path of G, then it is a contradiction that G' has a k-ary spanning tree as a subgraph.

Therefore, for chordal bipartite graphs and any $k \geq 2$, the k-ary SpT is NP-complete. □

Now, we shall show that the computational complexity of k-ary StT on chordal bipartite graphs is NP-complete in the following theorem. Our reduction is from HPATH on chordal bipartite graphs [8].

Proof of Theorem 2

Proof. It is easy to see that k-ary StT on chordal bipartite graphs and any $k \geq 2$ is in NP as the given certificate can be verified in deterministic polynomial time. We present a deterministic polynomial-time reduction that reduces an instance of HPATH on chordal bipartite graphs G to the corresponding instance of k-ary DEC_StT (G', R, l) problem on G', i.e.the decision version of k-ary StT with terminal set R that asks for the presence of a Steiner set of size at most l in G'. Our construction of G' is same as the construction of G' in Theorem **??**. We claim that G is an yes-instance of HPATH if and only if G' is an yes-instance of the DEC_StT $(G', R, l = n)$, where n is the number of vertices of G, and terminal set $R = \{u_i \mid 1 \leq i < k, u \in V(G)\}$. The proof is similar to the proof of the claim in Theorem 1. □

Proof of Lemma 2

Proof. We prove the first statement; the second follows symmetrically.

Suppose, for contradiction, that there exists a vertex $x_a \in A_2$ such that $N_G(x_a) \not\subseteq B_1$. If $N_G(x_a) = B_1$, then $(A_1 \cup \{x_a\}, B_1)$ forms a larger biclique than the chosen base biclique, contradicting maximality.

Otherwise, there exists $y_b \in B_2$ such that $\{x_a, y_b\} \in E(G)$. Since G is connected and bipartite chain graphs are P_5-free, we examine the induced subgraph on suitably chosen vertices from A_1 and B_1 (guaranteed by Lemma 1). One can then construct an induced path of length five: $x_a - y_b - x_c - y_k - x_k$, which contradicts the P_5-freeness of bipartite chain graphs [4]. Hence $N_G(x_a) \subseteq B_1$. □

Proof of Theorem 7

Proof. Let $R \subseteq V(G)$ be the set of terminals. By Observation 2, the class of bipartite chain graphs is hereditary under vertex deletion. Hence the induced subgraph on any vertex subset retains a nested neighborhood ordering.

To solve the k-ary Steiner Tree problem, we proceed as follows. For the given terminal set R, we iteratively delete non-terminal vertices whose removal preserves connectivity among the remaining vertices and maintains feasibility under the degree constraints. Because the structure is preserved under deletion, the instance reduces to checking whether a k-ary spanning tree exists in a suitable induced subgraph.

Thus, the k-ary Steiner Tree problem can be solved by invoking the k-ary spanning tree algorithm as a black box. Since that algorithm runs in polynomial time for fixed k, the Steiner version is also polynomial time solvable.

Proof of Theorem 9

Proof. It is easy to see that k-ary StT problem on strongly chordal split graphs and any $k \geq 2$ is in NP as the given certificate can be verified in deterministic polynomial time. We present a deterministic polynomial-time reduction that reduces an instance of HPATH on strongly chordal split graphs with two pendant vertices G to the corresponding instance of k-ary DEC_StT (H, R, l) problem on H, i.e.the decision version of k-ary StT problem with terminal set R that asks for the presence of a Steiner set of size at most l in H. Our construction of H is same as the construction of H in Theorem 8. We claim that G is a yes-instance of HPATH if and only if H is a yes-instance of the k-ary DEC_StT $(H, R, l = n)$,where $R = \{w_{ij} \mid v_i \in K(G), 1 \leq j \leq k-1\} \cup \{x_{ij} \mid u_i \in I(G), 1 \leq j \leq k-1\} \cup \{y^i_{jl} \mid u_i \in I(G), 1 \leq j \leq (k-1), 1 \leq l \leq k\} \cup \{d_i \mid 1 \leq i \leq 2\} \cup \{p_i \mid 1 \leq i \leq 2k\}$ and n is the number of vertices in the graph G. The proof is similar to the proof of the claim in Theorem 8. □

$K_{1,r}$-free split graphs In this section, we prove that for any $k \geq 2$, the k-ary SpT remains NP-complete on $K_{1,r}$-free split graphs when $r \geq k+4$ and is polynomial-time solvable on $K_{1,r}$-free split graphs, when $r \leq k+2$. Similarly, we show that for $K_{1,r}$-free split graphs, $r \geq 5$, the k-ary StT, when $k \leq r-1$ is

NP-complete and it is polynomial-time solvable on $K_{1,r}$-free split graphs, when $k \geq r$. Also, we show that for $K_{1,r}$-free split graphs, $r \leq 4$, the k-ary StT is polynomial time solvable, $k \geq 2$.

To show the computational complexity of the problems under consideration, we need a specific instance of $K_{1,5}$-free split graphs with $|K| = |I| - 1$ and two pendant vertices in I. Akiyama et al. [1] proved the NP-completeness of the Hamiltonian cycle in planar bipartite graphs with a maximum degree of 3. It is easy to see that if the bipartite graph has partitions of different sizes for the vertex set, then it is a no instance for the Hamiltonian cycle problem. It follows that the Hamiltonian cycle problem in planar bipartite graphs with maximum degree 3 and equal-sized vertex partitions is NP-hard. In [10], a reduction is given from the Hamiltonian cycle problem in a planar bipartite graph $G(A, B)$ with maximum degree 3 and $|A| = |B|$ to the Hamiltonian cycle problem in $K_{1,5}$-free split graph with $|K| = |I|$. Further, in [10], the Hamiltonian cycle problem in $K_{1,5}$-free split graph with $|K| = |I|$ is used to show that HPATH on $K_{1,5}$-free split graphs is NP-complete. In [10], it is also show that for $K_{1,5}$-free split graphs with $|K| = |I| - 1$ with two pendant vertices, HPATH is NP-complete. We use these special instances of $K_{1,5}$-free split graphs to show that the binary spanning tree problem is NP-complete on $K_{1,6}$-free split graphs. In general, theorem 13 holds.

Theorem 13. *For any $k \geq 2$,the k-ary spanning tree problem is NP-complete on $K_{1,r}$-free split graphs, where $r \geq k + 4$.*

Proof. It is easy to see that k-ary spanning tree problem on $K_{1,r}$-free split graphs, where $r \geq k + 4$ is in NP as the given certificate can be verified in deterministic polynomial time. We present a deterministic polynomial-time reduction that reduces an instance G of HPATH on $K_{1,5}$-free split graphs G with two pendant vertices to the corresponding instance H of k-ary ST problem. Our construction of H is same as the construction of H in Theorem 8. Observe that as the input instance G is $K_{1,5}$-free split graph, the Δ_G^I is 4 and hence by adding $k-1$ w_{ij}'s to v_i's $\in K(G)$, the Δ_H^I becomes $k-1+4= k+3$. Thus H is a $K_{1,r}$-free split graph, where $r \geq k + 4$. We claim that G is a yes-instance of HPATH if and only if H is a yes-instance of the k-ary ST problem. Further proof is similar to the proof of the claim in Theorem 8. □

We shall next present the polynomial-time algorithm for the binary spanning tree problem on $K_{1,r}$-free split graphs, where $r \leq 4$. We further generalize it for k-ary sapnning tree for any $k \geq 2$, on $K_{1,r}$-free split graphs, where $r \leq k + 2$.

Lemma 3. *For $K_{1,r}$-free split graphs, $1 \leq r \leq 2$, the binary spanning tree problem is polynomial-time solvable.*

Proof. We know that $K_{1,1}$-free split graphs are edgeless graphs which are trivial no instance of binary ST problem. Also $K_{1,2}$-free split graphs are complete graphs which are trivial yes instance of binary ST problem. Hence binary ST problem is trivially solvable for $K_{1,r}$-free split graphs, $1 \leq r \leq 2$.

Lemma 4. *[10] G is $K_{1,3}$-free split graph if and only if one of the following conditions holds.*
(i) $\Delta_G^I \leq 1$, or (ii) If there exists a vertex $u \in K$ such that $d_G^I(u) = 2$, then for all $v \in K$, $N_G^I(u) \cap N_G^I(v) \neq \emptyset$

Observation 3 *For a $K_{1,3}$-free split graph G, $\Delta_G^I \leq 2$*

Lemma 5. *[10] Let G be a $K_{1,3}$-free split graph. If $\Delta_G^I = 2$ then $|I| \leq 3$.*

Observation 4 *For a $K_{1,4}$-free split graph G, $\Delta_G^I \leq 3$*

Observation 5 *For a $K_{1,4}$-free split graph G with $\Delta_G^I \leq 3$, let $v \in K(G)$ and $deg_G^I(v) = 3$. For any vertex $u \in K(G) \setminus v$, $N_G^I(u) \cap N_G^I(v) \neq \emptyset$*

Using the above structural properties, for $K_{1,3}$-free and $K_{1,3}$-free split graphs, we solve the binary spanning tree problem in polynomial time.

Theorem 14. *For connected $K_{1,3}$-free split graphs, the binary spanning tree problem is polynomial-time solvable.*

Proof. We know from Lemma 4 that for $K_{1,3}$-free split graphs, $\Delta_G^I \leq 2$. Since $\Delta_G^I \leq 2$, we have the following three cases (1) $\Delta_G^I = 0$, (2) $\Delta_G^I = 1$ and (3) $\Delta_G^I = 2$. Case 1: $\Delta_G^I = 0$. This implies that $I = \emptyset$ in G as G is connected. Since G is clique, we obtain a path P on $|K|$ vertices, which is a binary tree T spanning $V(G)$. Case 2: $\Delta_G^I = 1$. Since $\Delta_G^I = 1$, any two vertices of I cannot have a common neighbor in K. This shows that $|K| \geq |I|$ otherwise it leads to a contradiction by pigeonhole principle. To construct a binary tree T spanning $V(G)$, we first obtain a path P on $|K|$ vertices. Let the path be $P = (x_1, x_2, \ldots, x_{|K|})$. For each vertex $v \in I$, we choose any one of its neighbors from P and add the corresponding edge to obtain T. Observe that T is a binary tree spanning $V(G)$; in particular, T is a comb. Case 3: $\Delta_G^I = 2$. Since $\Delta_G^I = 2$, it is known from Lemma 5 that $|I| \leq 3$. Case 3.1: $|I| \leq 1$. We first obtain a path P on $|K|$ vertices. When $|I| = 0$, the graph induced on $V(P)$ is a binary tree spanning $V(G)$. In case of $|I| = 1$, For the vertex $u \in I$, we choose any one of its neighbors from P and add the corresponding edge to obtain T.Case 3.2: $|I| = 2$. Let $I = \{y_1, y_2\}$. Case 3.2.1: $N_G(y_1) \cap N_G(y_2) = \emptyset$. This situation is not possible as $\Delta_G^I = 2$. Case 3.2.2: $N_G(y_1) \cap N_G(y_2) \neq \emptyset$. Without loss of generality, assume that $x_1 \in N_G(y_1) \cap N_G(y_2)$. Obtain a path P on K vertices starting at the vertex x_1. Consider $V(T) = V(P) \cup I$ and $E(T) = E(P) \cup \{\{x_1, y_1\}, \{x_1, y_2\}\}$. It is easy to see that the degree of x_1 is three in T. Thus T is a binary tree, spanning $V(G)$. Case 3.3: $|I| = 3$. Let $I = \{y_1, y_2, y_3\}$. Since G is $K_{1,3}$-free split, for all $x \in K$, $|N_G^I(x)| \leq 2$. This implies that $N_G(y_1) \cap N_G(y_2) \cap N_G(y_3) = \emptyset$. Case 3.3.1: For any $y_i, y_j \in I$, $N_G(y_i) \cap N_G(y_j) = \emptyset$. Without loss of generality, assume that $N_G(y_1) \cap N_G(y_2) = \emptyset$. Let $x_1 \in N_G(y_1)$,$x_2 \in N_G(y_2)$c, $x_3 \in N_G(y_3)$. Obtain a path P on $|K|$ vertices and add the edges $\{x_1, y_1\}, \{x_2, y_2\}, \{x_3, y_3\}$ to get a binary tree spanning $V(G)$.Case 3.3.2: There exists $y_i, y_j \in I$, $N_G(y_i) \cap N_G(y_j) \neq \emptyset$. We obtain a binary tree T, spanning $V(G)$ similar to Case 3.2.2. It clear from all cases that finding a binary spanning tree in $K_{1,3}$-free split graphs is polynomial-time solvable. □

Next lets analyze the binary spanning tree problem on $K_{1,4}$-free split graphs. We define the following terminologies to be used by subsequent lemmas on $K_{1,4}$-free split graphs. Let $G(K \cup I, E)$ be a connected $K_{1,4}$-free split graph and $\Delta_G^I = 2$. Let i=$|\{v \in V(G) | d_G^I(v) = 0\}|$, j=$|\{v \in V(G) | d_G^I(v) = 1\}|$, k =$|\{v \in V(G) | d_G^I(v) = 2\}|$ and $|K|$= $i + j + k$. We define two more positive integer variables l and m based on a small procedure: Repeatedly remove I-degree 2 vertices and their neighbors, if exist from G until we get a subgraph H of G where $\Delta_G^I \leq 1$. Now the variable l denotes the number of iterations we gone through or the number of I-degree 2 vertices removed and the variable m denotes the number of I-degree 0 vertices newly introduced in the subgraph H which are of I-degree 2 in G.

Observe that when $k = 0$ binary spanning tree always exist in G and it can be constructed using the procedure given in case 1 and 2 of Theorem 14. When $k \neq 0$ we have the following lemma.

Lemma 6. *For a $K_{1,4}$-free connected split graph $G(K \cup I, E)$ with $k \neq 0$, the existence of binary spanning tree on G is decided by the four cases. Case 1: When $i = 0$ and $j = 0$, binary spanning tree exists if and only if $|I| \leq |K| + 2$. Case 2: When $i \neq 0$ and $j = 0$, binary spanning tree exists if and only if ($|I| = 2k$ and $i \geq k - 2$) or ($|I| < 2k$ and $i \geq l - m - 3$). Case 3: When $i = 0$ and $j \neq 0$, binary spanning tree exists if and only if $|I| \leq k + 2 + j$. Case 4: When $i \neq 0$ and $j \neq 0$, binary spanning tree exists if and only if ($|I| = 2k + j$ and $i \geq k - 2$) or ($|I| < 2k + j$ and $i \geq l - m - 3$).*

Proof. Let $v_1, \ldots, v_k$ be the I-degree 2 vertices of G, $u_1, \ldots, u_j$ be the I-degree 1 vertices of G, $w_1, \ldots, w_i$ be the I-degree 0 vertices of G and $i_1, \ldots, i_{|I|}$ be the I-side vertices of G. Case 1: When $|I| \leq k + 2$, the binary spanning tree T of G looks like a caterpillar and can be constructed as follows: First obtain a path $P = (v_1, v_2, \ldots, v_k)$ on vertices of K-side of G, which act as the backbone of the caterpillar. Now attach at most two I-side vertices to v_1 and v_k and at most one I-side vertex to remaining v_i's based on the adjacency relation of G and by ensuring that all I-side vertices remains as pendant vertices in T. Observe that when $|I| = k+2$, the binary spanning tree obtained is a full tree with all internal nodes having degree exactly 3 and the situation shown in Figure 2.

Case 2: When $|I| = 2k$ and $i \geq k-2$, the binary spanning tree T of G will get a hierarchical structure and can be constructed as follows: First connect $\{v_1, v_2\}$ to w_1, then connect $\{w_1, v_3\}$ to w_2, then connect $\{w_2, v_4\}$ to w_3 and so on, finally connect $\{w_{i-1}, v_{k-1}, v_k\}$ to w_i. This gives a hierarchical structure as shown in Figure 3. Now observe that when $|I| = 2k$ and $i = k - 2$, the binary spanning tree obtained is a full tree with all internal nodes having degree exactly 3 and the situation shown in Figure 3.

When $|I| < 2k$, we should meet the condition $i \geq l - m - 3$ to get the binary spanning tree. The variables l and m are defined based on a small procedure explained above: Repeatedly remove I-degree 2 vertices and their neighbors, if exist from G until we get a subgraph H of G where $\Delta_G^I \leq 1$. Now the variable l denotes the number of iterations we gone through or the number of I-degree 2

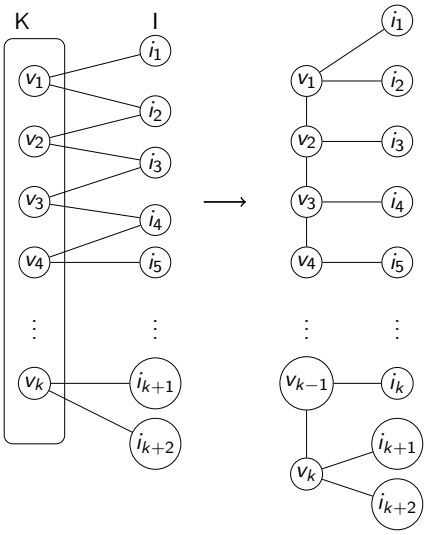

Fig. 2. Graph G and it's binary spanning tree - case 1 of lemma 6

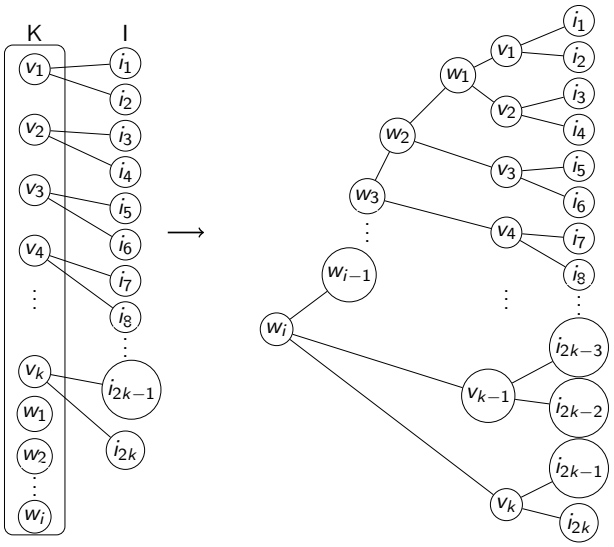

Fig. 3. Graph G and it's binary spanning tree - case 2 of lemma 6

vertices removed and the variable m denotes the number of I-degree 0 vertices newly introduced in the subgraph H which are of I-degree 2 in G. Now the procedure to construct the binary spanning tree is as follows. The l I-degree 2 vertices of G are connected to 2 vertices each from I-side of G and $l-1$ of them are connected each other using the I-degree 0 vertices w_i's and the m I-degree 2 vertices of G mentioned in above procedure. This gives a hierarchical structure as shown in Figure 3. Now remaining $k-l-m$ I-degree 2 vertices of K-side of G are connected to exactly one vertex each from I-side of G and it gives a comb like structure. Now observe that extreme end vertices of the backbone of this comb are of degree 2 and to which one of the vertices of l can be attached at one extreme end, and the other extreme end of the backbone of the comb is connected back to the root of the previous hierarchical structure to get the full binary spanning tree T. This structure is left as an exercise for the reader to figure out.

Case 3: When $|I| \leq k+j+2$, the binary spanning tree T of G looks like a caterpillar and can be constructed as follows: First obtain a path $P = (v_1, v_2, \ldots, v_t, v_{t+1}, \ldots, v_k)$ on I-degree 2 vertices of K-side of G. Now obtain another path $P_1 = (u_1, u_2, \ldots, u_j)$ on I-degree 1 vertices of K-side of G. First attach at most 1 vertex to each of the u_i's to get a comb like structure, may be with few tooth missing. Now attach at most two I-side vertices to v_1 and v_k and at most one I-side vertex to remaining v_i's based on the adjacency relation of G and by ensuring that all I-side vertices remains as pendant vertices in T. Now break the edge (v_t, v_{t+1}), $1 \leq t \leq k-1$ of path P and append the path P_1 in between to get the caterpillar like binary spanning tree T of G. Observe that When $|I| = k+j+2$, the binary spanning tree obtained is a full tree with all internal nodes having degree exactly 3 and the situation shown in Figure 4.

Case 4: When $|I| = 2k+j$ and $i \geq k-2$, the binary spanning tree T of G can be constructed as follows: First obtain a path $P = (u_1, u_2, \ldots, u_j)$ on all I-degree 1 vertices of G and then attach exactly 1 vertex to each of the u_i's to get a comb like structure. Now connect exactly 2 vertices to each of the v_i's and connect them each other using w_i's as follows: First connect $\{v_1, v_2\}$ to w_1, then connect $\{w_1, v_3\}$ to w_2, then connect $\{w_2, v_4\}$ to w_3 and so on, finally connect $\{w_{i-1}, v_{k-2}, v_{k-1}\}$ to w_i. This gives a hierarchical structure as shown in Figure 5. Now observe that extreme end vertices of the backbone P of the above comb are of degree 2 and to which one of the vertices of v_i's say v_k can be attached at one extreme end, and the other extreme end of the backbone P of the comb is connected back to the root w_i of the previous hierarchical structure to get the full binary spanning tree T as shown in Figure 5.

When $|I| < 2k+j$, we should meet the condition $i \geq l-m-3$ to get the binary spanning tree. The variables l and m are defined as in Case 2 of this lemma. Now the procedure to construct the binary spanning tree is as follows. The l I-degree 2 vertices of G are connected to 2 vertices each from I-side of G and $l-1$ of them are connected each other using the I-degree 0 vertices w_i's and the m I-degree 0 vertices of H mentioned in above procedure. This gives a hierarchical structure as shown in Figure 5 . Now remaining I-degree 1 vertices

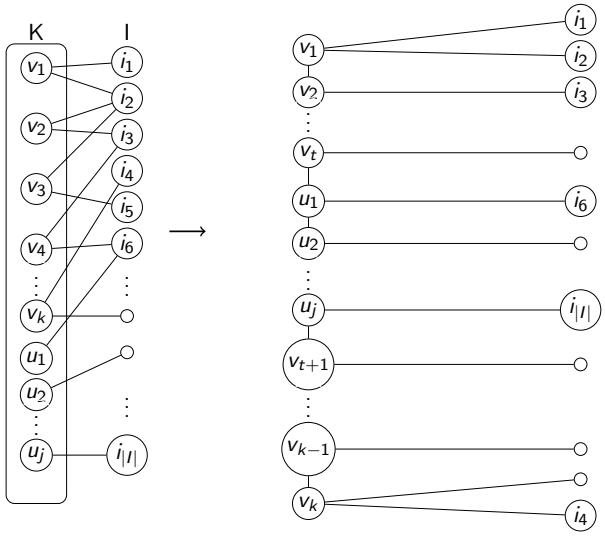

Fig. 4. Graph G and it's binary spanning tree - case 3 of lemma 6

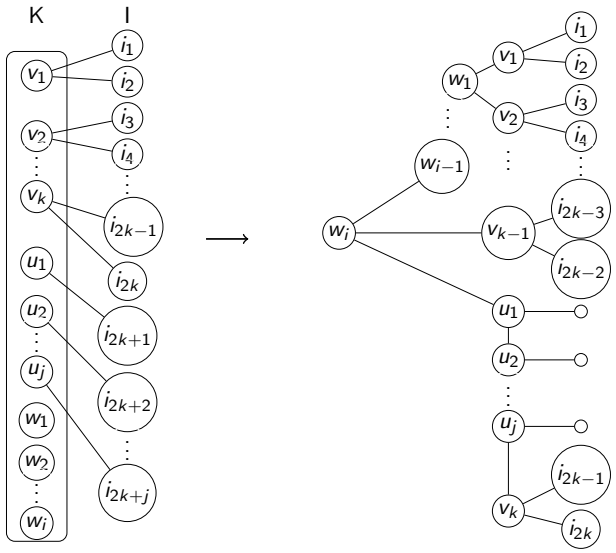

Fig. 5. Graph G and it's binary spanning tree - case 4 of lemma 6

of K-side of H are connected to at most one vertex each from I-side of G and it gives a comb like structure may be with few tooth missing. Now observe that extreme end vertices of the backbone of this comb are of degree at most 2 and to which one of the vertices of l can be attached at one extreme end, and the other extreme end of the backbone of the comb is connected back to the root of the previous hierarchical structure to get the full binary spanning tree T. This structure is left as an exercise for the reader to figure out. □

Lemma 7. *Let $G(K \cup I, E)$ be a connected $K_{1,4}$-free split graph and $\Delta_G^I = 3$. Consider any vertex $v \in K$ such that $deg_G^I(v) = 3$. Let $H = G \setminus \{v \cup N_G^I(v)\}$, then $\Delta_H^I = 2$ always.*

Proof. On the contrary, suppose $\Delta_H^I = 3$, then there exist a vertex $u \in K$ and $deg_H^I(u) = 3$, then $\{v \cup N_G^I(v) \cup u\}$ form a $K_{1,4}$ in G, which is a contradiction. □

Let us now see the conditions for existence of binary spanning tree on $K_{1,4}$-free split G with $\Delta_G^I = 3$. Lets define the variables x, y and z for G as: x=$|\{v \in V(G) | d_G^I(v) = 1\}|$, y=$|\{v \in V(G) | d_G^I(v) = 2\}|$, z=$|\{v \in V(G) | d_G^I(v) = 3\}|$ and $|K(G)| = x + y + z$. Note that there exist no I-degree 0 vertices in G. Variables i, j and k can be defined as above for H with $\Delta_H^I = 2$. Now observe that, when we obtain H from G, $i = x - 1$, $j = y - 1$ and $k = z - 1$. Let $K(G)$ and $I(G)$ denotes the K-side vertices and I-side vertices of G respectively. Similarly $K(H)$ and $I(H)$ denotes the K-side vertices and I-side vertices of H respectively. Observe that $|K(H)| = |K(G)| - 1$ and $|I(H)| = |I(G)| - 3$.

Observe that when $z = 0$ binary spanning tree always exist in G and it can be constructed using the Lemma 6. When $z \neq 0$ we have the following lemma.

Lemma 8. *For a $K_{1,4}$-free connected split graph $G(K \cup I, E)$ with $z \neq 0$, the existence of binary spanning tree on G is decided by the four cases. Case 1: When $i = 0$ and $j = 0$, binary spanning tree exists if and only if $|I(H)| \leq |K(H)|$. Case 2: When $i \neq 0$ and $j = 0$, binary spanning tree exists if and only if ($|I(H)| = 2k$ and $i \geq k$) or ($|I(H)| < 2k$ and $i \geq l - m - 1$). Case 3: When $i = 0$ and $j \neq 0$, binary spanning tree exists if and only if $|I(H)| \leq k + j$. Case 4: When $i \neq 0$ and $j \neq 0$, binary spanning tree exists if and only if ($|I(H)| = 2k + j$ and $i \geq k$) or ($|I(H)| < 2k + j$ and $i \geq l - m - 1$).*

Proof. Let $G(K \cup I, E)$ be a connected $K_{1,4}$-free split graph and $\Delta_G^I = 3$. Consider any vertex $v \in K$ such that $deg_G^I(v) = 3$. Let $H = G \setminus \{v \cup N_G^I(v)\}$, then by Lemma 7, $\Delta_H^I = 2$. Let the removed I-degree 3 vertex v has neighbors say $N_G^I(v) = \{a, b, c\}$. Now by Observation 5, for any vertex $u \in K(H)$, there exist some element say $c \in N_G^I(u) \cap N_G^I(v)$. Let $v_1, \ldots, v_k$ be the I-degree 2 vertices of H, $u_1, \ldots, u_j$ be the I-degree 1 vertices of H, $w_1, \ldots, w_i$ be the I-degree 0 vertices of H and $i_1, \ldots, i_{|I|}$ be the I-side vertices of H. Case 1: When $|I(H)| \leq |K(H)|$, equivalently $|I(G)| \leq K(G) + 2$, the binary spanning tree T of G looks like a caterpillar and can be constructed as follows: First obtain a path $P = (v_1, v_2, \ldots, v_k)$ on vertices of K-side of H. Now attach at most two I-side vertices to v_1 and at most one I-side vertex to remaining v_i's, $2 \leq i \leq k - 1$

based on the adjacency relation of H and by ensuring that all I-side vertices of H remains as pendant vertices in T. Now attach the removed vertices c and v to vertex v_k and finally attach only vertices a and b to vertex v. Observe that When $|I(G)| = K(G) + 2$, the binary spanning tree obtained is a full tree with all internal nodes having degree exactly 3 and the structure is left as an exercise for the reader to figure out.

Case 2: When $|I(H)| = 2k$ and $i \geq k$, the binary spanning tree T of G will get a hierarchical structure and can be constructed as follows: First connect $\{v_1, v_2\}$ to w_1, then connect $\{w_1, v_3\}$ to w_2, then connect $\{w_2, v_4\}$ to w_3 and so on, finally connect $\{w_{i-3}, v_{k-1}, v_{k-2}\}$ to w_{i-2}. This gives a hierarchical structure. Now by Observation 5, for the removed I-degree 3 vertex v, there exist some element say $c \in N_G^I(w_{i-1}) \cap N_G^I(v)$. Connect this c and v to w_{i-1}. Connect a and b to v. Further connect w_{i-1} and v_k to w_i. Finally w_i is connected back to w_{i-2}, the root of the hierarchical structure to get the binary spanning tree of G. Observe that when $|I(H)| = 2k$ and $i = k$, the binary spanning tree obtained is a full tree with all internal nodes having degree exactly 3 and the structure is left as an exercise for the reader to figure out.

When $|I(H)| < 2k$, we should meet the condition $i \geq l - m - 1$ to get the binary spanning tree. Here the binary spanning tree can be obtained similar to Case 2 of Lemma 6, and by attaching the removed I-degree 3 vertex v and its neighbors carefully, as explained before. This structure is left as an exercise for the reader to figure out.

Case 3: When $|I(H)| \leq k + j$, the binary spanning tree T of G looks like a caterpillar and can be constructed as follows: First obtain a path $P = (v_1, v_2, \ldots, v_t, v_{t+1}, \ldots, v_k, v)$ on I-degree 2 vertices of K-side of H and v. Now obtain another path $P_1 = (u_1, u_2, \ldots, u_j)$ on I-degree 1 vertices of K-side of H. First attach at most 1 vertex to each of the u_i's to get a comb like structure, may be with few tooth missing. Now attach at most two I-side vertices to v_1 and at most one I-side vertex to remaining v_i's, $2 \leq k - 1$ based on the adjacency relation of H and by ensuring that all I-side vertices remains as pendant vertices in T. By Observation 5, for the removed I-degree 3 vertex v, there exist some element say $c \in N_G^I(v_{k-1}) \cap N_G^I(v)$. Connect this c to v_{k-1}. Connect a and b to v. Finally break the edge (v_t, v_{t+1}), $1 \leq t \leq k-1$ of path P and append the path P_1 in between to get the caterpillar like binary spanning tree T of G. Observe that When $|I| = k + j$, the binary spanning tree obtained is a full tree with all internal nodes having degree exactly 3 and the structure is left as an exercise for the reader to figure out.

Case 4: When $|I(H)| = 2k + j$ and $i \geq k$, the binary spanning tree T of G can be constructed as follows: First obtain a path $P = (u_1, u_2, \ldots, u_j)$ on all I-degree 1 vertices of H and then attach exactly 1 vertex to each of the u_i's to get a comb like structure. Now connect exactly 2 vertices to each of the v_i's and connect them each other using w_i's as follows: First connect $\{v_1, v_2\}$ to w_1, then connect $\{w_1, v_3\}$ to w_2, then connect $\{w_2, v_4\}$ to w_3 and so on, finally connect $\{w_{i-3}, v_{k-2}, v_{k-1}\}$ to w_{i-2}. This gives a hierarchical structure. Now observe that extreme end vertices of the backbone P of the above comb are of degree 2 and to

which one of the vertices of v_i's say v_k can be attached at one extreme end. Now by Observation 5, for the removed I-degree 3 vertex v, there exist some element say $c \in N_G^I(w_{i-1}) \cap N_G^I(v)$. Connect this c and v to w_{i-1}. Connect a and b to v. Further connect w_{i-1} and other extreme end of the above comb like structure with backbone as P to w_i. Finally w_i is connected back to w_{i-2}, the root of the hierarchical structure obtained to get the final binary spanning tree of G.

When $|I(H)| < 2k + j$, we should meet the condition $i \geq l - m - 1$ to get the binary spanning tree. Here the binary spanning tree can be obtained similar to Case 4 of Lemma 6, and by attaching the removed I-degree 3 vertex v and its neighbors carefully, as explained before. This structure is left as an exercise for the reader to figure out. □

By Lemma 6 and Lemma 8, we have the following theorems.

Theorem 15. *For connected $K_{1,4}$-free split graphs, the binary spanning tree problem is polynomial time solvable.*

Theorem 16. *Let $k > 2$, and let G be a $K_{1,r}$-free split graphs, where $r \leq k + 2$. The k-ary spanning tree problem is polynomial-time solvable.*

Similar to the results obtained for the k-ary spanning tree problem on split graphs, we obtain results for the k-ary Steiner tree problem on split graphs. We highlight our results on the k-ary Steiner tree problem on split graphs, and the proofs are presented in the Appendix.

- For $K_{1,r}$-free split graphs, $r \geq 5$, the k-ary Steiner tree problem, when $k \leq r - 1$ is NP-complete.
- The k-ary Steiner tree problem is polynomial-time solvable on $K_{1,r}$-free split graphs, when $k \geq r$.
- We show that for $K_{1,r}$-free split graphs, $r \leq 4$, the k-ary Steiner tree problem is polynomial time solvable, $k \geq 2$.

The proof of Theorem 16.

Proof. We know that from the definition of $K_{1,r+1}$-free split graphs, for every $x \in K$, $d_G^I(x) \leq r$. This leads to the following cases: Case 1: For all $x \in K$, $d_G^I(x) \leq r - 1$. We construct a r-ary tree T, spanning $V(G)$ as follows. First, we obtain a path P on $|K|$ vertices. Consider a tree T, $V(T) = V(P) \cup I$ and $E(T) = E(P) \cup \{\{x, y\} | x \in K, y \in I\}$. By the definition of r-ary tree, we know that the degree of each vertex in a r-ary tree is at most $r + 1$. Since for all $x \in K$, $d_G^I(x) \leq r - 1$, the graph induced on T is a r-ary tree, spanning $V(G)$. (ii) There exist set of vertices $x_i \in K$ such that $d_G^I(x) = r$. Let H be the graph induced on $V(G) \setminus N_G[x_i]$. Similar to the idea followed in Theorem 15, to find the binary spanning tree in $K_{1,4}$-free split graphs can be adopted to obtain a tree T', spanning $V(H)$. Observe that, T' is a r-ary tree, spanning $V(H)$. Note that T' has a path P on $|K|$ vertices. Let x_i be the end vertex of the path. We extend T' by adding the following edges $\{x_i, x\} \cup \{\{x, y\} | y \in N^I(x) \cap I\}$ to T'. Note that $d_G(x) \leq r + 1$ in T'. Thus, we obtain a r-ary tree spanning $V(G)$. □

Theorem 17. *The k-ary StT is polynomial-time solvable on $K_{1,3}$-free split graphs ($K_{1,4}$-free split graphs).*

Proof. Proof of k-ary Steiner tree on $K_{1,3}$-free split graphs ($K_{1,4}$-free split graphs) is similar to the proof of k-ary spanning tree on $K_{1,3}$-free split graphs ($K_{1,4}$-free split graphs).

Theorem 18. *For $K_{1,r}$-free split graphs, $r \geq 5$, the k-ary StT, $4 \leq k \leq r-1$ is NP-complete.*

Proof. It is known [7] that the Exact l-cover problem, $l \geq 3$ is NP-complete. We map an instance of Exact $(l = k-1)$-cover (Z, T) problem to the corresponding instance of k-ary Steiner tree problem (G, R, s) in $K_{1,r}$-free split graphs, $r \geq 5$ where $4 \leq k \leq r-1$ and $l = k-1$. Mapping is as follows: $I = Z$, $K = \{v_i \mid c_i \in T\}$, $1 \leq i \leq n$ and $V(G) = K \cup I$, $E(G) = E' \cup E''$, $E' = \{\{v_i, v_j\} \mid v_i, v_j \in K\}, 1 \leq i \neq j \leq n$, $E'' = \{\{v_i, u\} \mid v_i \in K, u \in I \text{ and } u \in c_i\}$. Note that $|V(G)| = |Z|+|T|$ and $|E(G)| = (\frac{|T|}{2})+l|T|$. The above construction is, therefore, polynomial to the size of the input.

We now show that instances created in this reduction are $K_{1,r}$-free split graphs, $r \geq 5$. Without loss of generality, assume that $r = 5$. This implies that $k = 4$, $l = 3$. On the contrary, assume that there exists a $K_{1,5}$ induced on vertices $\{u, v, w, x, y, z\}$. Clearly, at most, two vertices u, v from clique K can be included in the $K_{1,5}$. Note that $\{w, x, y, z\} \subseteq I$. Without loss of generality, assume that $d^I_G(v) = 4$. This shows that there exists a four-element subset $c \in T$ corresponding to the clique vertex $v \in K$, which is a contradiction as all subsets are of size 3 in collection T. Therefore, it follows that the reduced graph G is $K_{1,5}$-free split graph. Observe that the above argument can be extended to any $r \geq 6$.

We now show that there exists an Exact $(l = k-1)$-cover(Z, T) if and only if there exists a k-ary Steiner tree (G, R, s) in G with $R = I$ and at most $s = |\frac{|Z|}{(l=k-1)}|$ Steiner vertices.

($\Rightarrow$) Suppose there exists $T' \subseteq T$, $|T'| = \frac{|Z|}{(l=k-1)}$ that covers all of Z then the set of vertices $S = \{v \in K \mid c \in T'\}$ forms a Steiner set in G as $R = I$ and $|S| = \frac{|Z|}{(l=k-1)}$.

($\Leftarrow$) If there exists a Steiner set $S \subseteq K$ in G with at most $s = \frac{|Z|}{(l=k-1)}$ Steiner vertices, then the following observations are true; for all vertex $v \in S$, $d^I_G(v) = l = k-1$, $|S| = \frac{|Z|}{(l=k-1)}$ and $|N^I_G(S)| = |Z|$. Note that there does not exist $u, v \in S$ such that $N^I_G(u) \cap N^I_G(v) \neq \emptyset$. Therefore, $T' = \{c \in T \mid v \in S\}$ forms an Exact $(l = k-1)$-cover of Z. Thus, k-ary Steiner tree problem, $4 \leq k \leq r-1$ is NP-complete on $K_{1,r}$-free split graphs, $r \geq 5$. □

Theorem 19. *For $K_{1,r}$-free split graphs, the k-ary StT, $k \geq r$ is polynomial-time solvable.*

Proof. Recall that for every $x \in K$ in $K_{1,r}$-free split graphs, $d^I_G(x) \leq r-1$. We construct a k-ary Steiner tree T, $R = I$ as follows. Let $Z = \{x \mid N^I(x) \cap R \neq \emptyset\}$.

First, we obtain a path P on $|Z|$ vertices. We now construct a k-ary Steiner tree T by adding the following edges $\{\{x,y\}|y \in N^I(x) \cap R\}$ to T. Note that $d_G(x) \leq k+1$ in T. Thus, we obtain a k-ary Steiner tree $\square$

Theorem 20. *The 2-ary StT in $K_{1,5}$-free split graphs is NP-complete.*

Proof. We know that Exact 3 cover [7] problem is NP-complete. We use Exact 3 cover problem as a candidate instance to show 2-ary Steiner tree problem is NP-complete on $K_{1,5}$-free split graphs An instance of Exact 3 cover (Z,T) is reduced to the corresponding instance of 2-ary Steiner tree (G,R,l) problem as follows: $K = K_1 \cup K_2 \cup K_3$, $K_1 = \{u_i \mid c_i \in T\}$, $1 \leq i \leq n$, $K_2 = \{v_i, w_i \mid q_i \in Z\}$, $1 \leq i \leq 3q$, $I = I_1 \cup I_2 \cup I_3 \cup I_4$, $I_1 = Z$, $I_2 = \{x_{i1}, x_{i2} \mid w_i \in K_1\}$, $I_3 = \{y_i \mid v_i \in K_1\}$, $I_4 = \{z_i \mid v_{i+3} \in K_1\}, 2 \leq i \leq |K_1|$, $K_3 = \{p_{i1}, p_{i2} \mid z_i \in I_4\}$. $E(G) = E_1 \cup E_2$, $E_1 = \{\{v_i, v_j\} \mid v_i, v_j \in K\}, 1 \leq i \neq j \leq n$, $E_2 = \{\{v_i, u\} \mid v_i \in K_1, u \in I_1$ and $u \in c_i\}$, $E_3 = \{\{v_i, u\}, \{w_i, u\} \mid v_i, w_i \in K_1$, and $u \in I_1$, $E_4 = \{\{x_{i1}, w_i\}, \{x_{i2}, w_i\} \mid w_i \in K_1, x_{i1}, x_{i2} \in I_2\}$, $E_5 = \{\{y_i, v_i\} \mid v_i \in K_1, y_i \in I_3\}$, $E_6 = \{\{z_i, v_{i+3}\} \mid v_{i+3} \in K_1, z_i \in I_4\}$, $E_7 = \{\{z_i, p_{i1}\}, \{z_i, p_{i2}\} \mid p_{i1}, p_{i2} \in K_3, z_i \in I_4\}$. Note that $|V(G)| = 6|Z| + |T| + 10$ and $|E(G)| = (\frac{|T|}{2}) + l|T| + |K_2| + |I_2| + |I_3| + 3|I_4|$. The above construction is, therefore, polynomial to the size of the input.

We show that G is $K_{1,5}$-free split graphs. On the contrary, assume that there exists a $K_{1,5}$ in $K_1 \cup I_1$ on vertices $\{u,v,w,x,y,z\}$. Clearly, at most two vertices u, v from clique K_1 can be included in the $K_{1,5}$. Note that $\{w,x,y,z\} \subseteq I_1$. Without loss of generality, assume that $d^I_{1_G(v)} = 4$. This shows that there exists a four element subset $c \in T$ corresponding to the clique vertex $v \in K$, which is a contradiction. By our construction, vertices of K_2 and K_3 have degree two and cannot induce in G. Thus G is $K_{1,5}$-free split graph. We claim that if and only if G is a yes-instance of the 2-ary Steiner tree problem (H,R,l), where $R = I$.

($\Rightarrow$) Suppose there exists $T' \subseteq T$ such that it covers all of Z, then the set of vertices $S = \{u \in K_1 \mid c \in T'\} \cup K_2 \cup K_3$ forms a Steiner set in G as $R = I$. By our construction, all the newly added vertices have a degree of at most three. Also, observe that each $u \in K_1$ is adjacent to exactly three vertices. Thus the tree obtained is a 2-ary Steiner tree.

($\Leftarrow$) We observe that no two vertices belonging to K_1 cannot be adjacent in any 2-ary Steiner tree. Since the number of vertices used from K_1 is minimum and it covers all of I_1, $T = \{c \in T \mid v \in S\}$ forms an Exact-3 cover of Z. $\square$

A construction similar to that of Theorem 20 can be used to prove the 3-ary Steiner tree problem in $K_{1,5}$-free split graphs is NP-complete.

Theorem 21. *The 1-ary Steiner tree problem in $K_{1,r}$-free split graph, $r \geq 5$ is NP-complete.*

Proof. We observe that 1-ary Steiner tree problem when $R = V(G)$ is equivalent to the Hamiltonian path problem. Since the Hamiltonian path problem is NP-complete on $K_{1,r}$-free split graph, $r \geq 5$, the proof follows directly. $\square$

Theorem 22. *The 2-ary StT in $K_{1,r}$-free split graph, $r \geq 6$ is NP-complete.*

Proof. We know from [10] that the Hamiltonian path problem is NP-complete on $K_{1,5}$-free split graphs with $|K| = |I| - 1$. Now, we map an instance of the Hamiltonian problem on $K_{1,5}$-free split graphs to corresponding instance of 2-ary Steiner tree problem (H, R, l) on $K_{1,r}$-free split graph, $r \geq 6$ as follows; $V(H) = K' \cup I'$, $K' = K$ and $I' = I \cup \{w_i \mid v_i \in K\}$. The edge of H is $E(H) = E(G) \cup \{\{w_i, v_i\} \mid 1 \leq i \leq |K|$. This completes the construction of H. Since the newly added vertices are pendent vertices and adjacent to their corresponding clique vertices, the graph G becomes $K_{1,r}$-free split graphs, $r \geq 6$. We claim that G is a yes-instance of HPATH if and only if H is a yes-instance of the 2-ary Steiner tree problem $(H, R, l = n)$,where $R = I' \cup \{w_i \mid 1 \leq i < |K|\}$.

($\Rightarrow$) Suppose that a Hamiltonian path P exists in G. We construct a 2-ary Steiner tree as follows: By our construction, we know that w_i is adjacent to its corresponding vertex v_i in the clique. We obtain a 2-ary Steiner tree of H by including the corresponding pendant vertices for each vertex v_i in the path.

($\Leftarrow$) Suppose that there exists a 2-ary Steiner tree with at most $l = n$ Steiner vertices. By our construction, w_i is adjacent to v_i. This shows that in any 2-ary Steiner tree, v_i is adjacent to two vertices other than w_i. Since $2|K'| = |I'| - 1$, v_i must be adjacent to two vertices of I'. We observe that in H, any 2-ary Steiner tree without w_i, $1 \leq i \leq |K'|$ forms a path P. Since $V(G) = V(H) \setminus \{w_1, \ldots, w_{|K'|}\}$, the path P is a Hamiltonian path in G. Therefore, for $K_{1,r}$-free split graphs, $r \geq 6$, the 2-ary Steiner tree problem is NP-complete. □

k-ary Steiner Tree. Let $G = (V, E)$ be a graph and let $R \subseteq V$ be a specified set of terminals. A tree is called *k-ary* if the degree of every vertex is at most $k + 1$. The k-ary Steiner Tree problem asks whether G admits a tree that spans all vertices in R and satisfies the degree constraint.

We express this problem in MSO_2. We use the standard MSO_2 incidence predicate $\mathsf{inc}(v, e)$, which holds if and only if vertex v is an endpoint of edge e. For convenience, we define the auxiliary predicate

$$\mathsf{Ends}(u, v, e) := \mathsf{inc}(u, e) \wedge \mathsf{inc}(v, e) \wedge u \neq v,$$

which denotes that edge e connects vertices u and v.

We existentially quantify an edge set $T \subseteq E$, intended to represent the Steiner tree, and require that T connects all terminals, is acyclic, and respects the degree bound.

Formally, the MSO_2 sentence is

$$\exists T \subseteq E \left(\mathsf{TerminalSpanning}(T) \wedge \mathsf{TerminalConnected}(T) \wedge \mathsf{Acyclic}(T) \wedge \mathsf{DegreeBound}(T)\right),$$

where the predicates are defined as follows.

Terminal spanning.

$$\mathsf{TerminalSpanning}(T) := \forall v \in R : \ \exists e \in T \text{ such that } \mathsf{inc}(v, e).$$

Terminal connectivity.

$$\mathsf{TerminalConnected}(T) := \forall X \subset V \left(X \cap R \neq \emptyset \wedge R \setminus X \neq \emptyset \ \rightarrow \ \exists e \in T \ \exists u \in X \ \exists v \in V \setminus X : \mathsf{Ends}(u, v, e)\right).$$

$$\mathsf{Used}(v, T) := \exists e \in T : \ \mathsf{inc}(v, e).$$

Acyclicity (corrected).

$$\mathsf{Acyclic}(T) := \neg\exists C \subseteq V \ \Big(\forall v \in C : \ \mathsf{Used}(v, T) \wedge \exists u, w \in C\big(u \neq w \wedge \exists e_1, e_2 \in T : \mathsf{Ends}(v, u, e_1) \wedge \mathsf{Ends}(v, w, e_2)\big)\Big)$$

Degree bound (corrected).

$$\mathsf{DegreeBound}(T) := \forall v \in V : \ \mathsf{Used}(v, T) \ \rightarrow \ \neg\exists u_1, \ldots, u_{k+2} \in V\Big(\bigwedge_{i \neq j} u_i \neq u_j \wedge \bigwedge_{i=1}^{k+2} \exists e_i \in T : \mathsf{Ends}(v, u_i, e_i)\Big).$$

Algorithmic Consequences. The MSO_2 formulations of the k-ary Spanning Tree and the k-ary Steiner Tree problems yield immediate algorithmic implications. By Courcelle's Theorem, for every fixed integer $k \geq 1$, there exists a computable function f such that both problems can be solved in time $f(\mathrm{tw}(G)) \cdot |V(G)|$, where $\mathrm{tw}(G)$ denotes the treewidth of the input graph.

Consequently, on graph classes of bounded treewidth, both the k-ary Spanning Tree and the k-ary Steiner Tree problems admit linear-time algorithms. Moreover, the logical characterizations imply that these problems can be handled uniformly for all fixed values of k using standard dynamic programming techniques over tree decompositions, without requiring problem-specific structural insights.

Proof of Theorem 12

Proof. Both problems are expressible in MSO_2. In particular, there exist MSO_2 formulas that define, respectively, a spanning tree and a Steiner tree whose maximum vertex degree is bounded by $k + 1$. By Courcelle's Theorem, every problem definable in MSO_2 can be solved in time $f(\mathrm{tw}(G)) \cdot |V(G)|$ on a graph G, for some computable function f depending only on the formula. Hence, for each fixed k, both the k-ary Spanning Tree and the k-ary Steiner Tree problems are fixed-parameter tractable when parameterized by treewidth. □