Partitioning Perfect Graphs into Stars

René van Bevern, Robert Bredereck, Laurent Bulteau, Jiehua Chen, Vincent Froese, Rolf Niedermeier, Gerhard J. Woeginger

Partitioning Perfect Graphs into Stars*

René van Bevern[1,2], Robert Bredereck[2], Laurent Bulteau[3], Jiehua Chen[2], Vincent Froese[2], Rolf Niedermeier[2], and Gerhard J. Woeginger[4]

[1]Novosibirsk State University, Novosibirsk, Russian Federation, `rvb@nsu.ru`
[2]Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Germany, {`robert.bredereck,jiehua.chen,vincent.froese,rolf.niedermeier`}`@tu-berlin.de`
[3]Institut Gaspard-Monge, Université Paris-Est Marne-la-Vallée, France
`l.bulteau@gmail.com`
[4]Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands,
`gwoegi@win.tue.nl`

March 23, 2017

Abstract

The partition of graphs into "nice" subgraphs is a central algorithmic problem with strong ties to matching theory. We study the partitioning of undirected graphs into same-size stars, a problem known to be NP-complete even for the case of stars on three vertices. We perform a thorough computational complexity study of the problem on subclasses of perfect graphs and identify several polynomial-time solvable cases, for example, on interval graphs and bipartite permutation graphs, and also NP-complete cases, for example, on grid graphs and chordal graphs.

1 Introduction

We study the computational complexity (tractable versus intractable cases) of the following basic graph problem.

STAR PARTITION
Input: An undirected n-vertex graph $G = (V, E)$ and an integer $s \in \mathbb{N}$.
Question: Can the vertex set V be partitioned into $k := \lceil n/(s+1) \rceil$ mutually disjoint vertex subsets $V_1, V_2, \ldots, V_k$, such that each subgraph $G[V_i]$ contains an s-star (a $K_{1,s}$)?

Two prominent special cases of STAR PARTITION are the case $s = 1$ (finding a perfect matching) and the case $s = 2$ (finding a partition into connected

*An extended abstract of this work appeared under the title "Star Partitions of Perfect Graphs" in Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP'14), Part I, LNCS 8572, pp. 174–185, Springer, 2014.

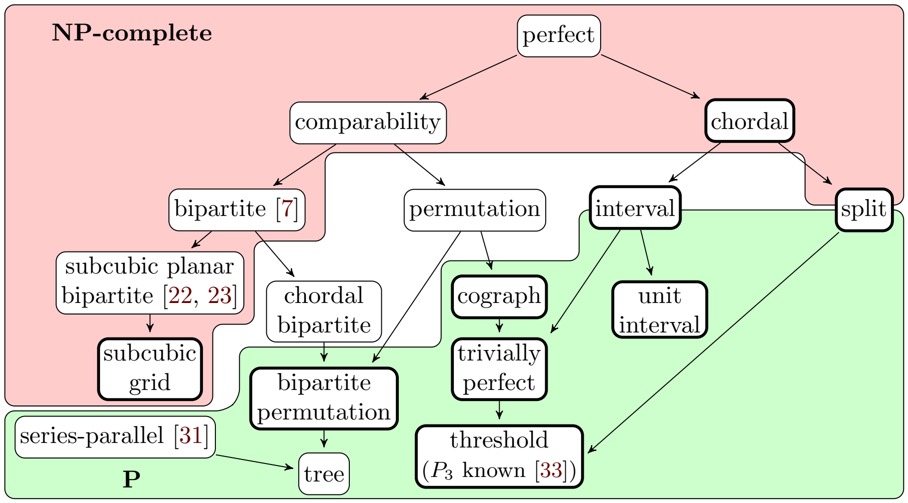


Figure 1: Complexity classification of Star Partition. Bold borders indicate results of this paper. An arrow from a class A to a class B indicates that A contains B. In most classes, NP-completeness results hold for $s = 2$ (that is, for P_3-Partition). However, on split graphs, Star Partition is polynomial-time solvable for $s \leq 2$, while it is NP-complete for $s \geq 3$. P_3-Partition is solvable on interval graphs in quasilinear time. We are not aware of any result for permutation graphs, chordal bipartite graphs, interval graphs (for $s \geq 3$), or grid graphs (for $s = 3$).

triples). Perfect matchings ($s = 1$), of course, can be found in polynomial time. Partitions into connected triples (the case $s = 2$), however, are hard to find; this problem, denoted P_3-Partition, was proven to be NP-complete by Kirkpatrick and Hell [18].

Our goal in this paper is to achieve a better understanding of star partitions for certain subclasses of perfect graphs. We provide a fairly complete classification in terms of polynomial-time solvability versus NP-completeness on the most prominent subclasses of perfect graphs, leaving a few potentially challenging cases open; see Figure 1 for an overview of our results.

Motivation. The literature in algorithmic graph theory is full of packing and partitioning problems (packing is an optimization variant of partitioning, where one tries to maximize the number of disjoint vertex subsets). Concerning practical relevance, note that P_3-Packing and P_3-Partition find applications in dividing distributed systems into subsystems [20] as well as in the Test Cover problem arising in bioinformatics [13]. In particular, the application in distributed systems explicitly motivates the consideration of very restricted (perfect) graph classes such as grid-like structures. Star Partition on grid graphs naturally occurs in political redistricting problems [4]. We show that Star Partition remains NP-hard on subcubic grid graphs.

Interval graphs are a further famous class of perfect graphs. Here, Star Partition can be considered a team formation problem: Assume that we have a number of agents, each being active during a certain time interval. Our goal is to form teams, all of the same size, such that each team contains at least one agent sharing time with every other team member. This specific team member becomes the team leader, since he or she can act as an information hub. Forming such teams is nothing else than solving Star Partition on interval graphs. We present efficient algorithms for Star Partition on unit interval graphs (that is, for the case when all agents are active for the same amount of time) and for P_3-Partition on general interval graphs.

Related work. Packing and partitioning problems are central problems in algorithmic graph theory with many applications and with close connections to matching theory [35]. In the case of packing, one wants to maximize the number of graph vertices that are "covered" by vertex-disjoint copies of some fixed pattern graph H. In the case of partitioning, one wants to cover *all* vertices in the graph. We focus on the partitioning problem, which is also called H-Factor in the literature. In this work, we always refer to it as H-Partition. Since Kirkpatrick and Hell [18] established the NP-completeness of H-Partition on general graphs for every connected pattern H with at least three vertices, one branch of research has turned to the investigation of classes of specially structured graphs. For instance, on the upside, H-Partition has been shown to be polynomial-time solvable on trees and series-parallel graphs [31] and on graphs of maximum degree two [23]. On the downside, P_k-Partition (for each fixed $k \geq 3$) remains NP-complete on planar bipartite graphs [14]; this hardness result is generalized to H-Partition on planar graphs for any outerplanar pattern H with at least three vertices [2]. For every fixed $s \geq 2$, Star Partition is NP-complete on bipartite graphs [7]. Partitioning into triangles (K_3), that is, K_3-Partition, is polynomial-time solvable on chordal graphs [12] and linear-time solvable on graphs of maximum degree three [25].

An optimization version of P_k-Partition, called Min P_k-Partition, has also received considerable interest in the literature. This version asks for a partition of a given graph into a minimum number of paths, each of length *at most* k. Clearly, all hardness results for P_k-Partition carry over to this minimization version. If k is *part of* the input, then Min P_k-Partition is hard for cographs [29] and chordal bipartite graphs [30]. In fact, Min P_k-Partition is NP-complete even on convex graphs and trivially perfect graphs (also known as quasi-threshold graphs), and hence on interval and chordal graphs [1]. Min P_k-Partition is solvable in polynomial time on trees [34], threshold graphs, cographs (for fixed k) [29] and bipartite permutation graphs [30].

While in this work we study the H-Partition problem, which partitions the vertex set of a graph into mutually vertex-disjoint copies of some fixed pattern graph H, the literature also studies the H-Decomposition problem, which partitions the *edge set* of a graph into mutually *edge*-disjoint copies of a pattern H. In general, H-Decomposition is NP-hard [8], yet easy to solve on highly-connected graphs if H is a k-star: Thomassen [32] shows that every $(k^2 + k)$-edge-connected graph has a k-star decomposition provided its number of edges is a multiple of k. Lovász et al. [21] strengthen this result to $(3k - 3)$-edge-connected graphs for odd $k \geq 3$. However, since a graph may have a k-star decomposition without having a k-star partition and vice versa, the results on H-Decomposition are not applicable to the Star Partition problem considered in our work.

Our contributions. So far, surprisingly little was known about the complexity of Star Partition for subclasses of perfect graphs. We provide a detailed picture of the corresponding complexity landscape for classes of perfect graphs; see Figure 1 for an overview. Let us briefly summarize our major findings. (Note that all problem variants we consider are clearly contained in NP, which means that our NP-hardness results in fact imply NP-completeness.)

As a central result, we provide a quasilinear-time algorithm for P_3-Partition (which is Star Partition with $s = 2$) on interval graphs; the complexity of Star Partition for $s \geq 3$ remains open. But if we restrict the input graphs to be unit interval graphs or trivially perfect graphs, we can solve Star Partition even in linear time. Furthermore, we develop a polynomial-time algorithm for Star Partition on cographs and on bipartite permutation graphs. Most of our polynomial-time algorithms are simple to describe: they are based on dynamic programming or even on greedy approaches, and hence should work well in implementations. Their correctness proofs, however, are intricate.

On the boundary of NP-completeness, we strengthen a result of Małafiejski and Żyliński [22] and Monnot and Toulouse [23] by showing that P_3-Partition is NP-hard on grid graphs with maximum degree three. Note that in strong contrast to this, K_3-Partition is linear-time solvable on graphs with maximum degree three [25]. Furthermore, we show P_3-Partition to be NP-hard on chordal graphs, while K_3-Partition is known to be polynomial-time solvable in this case [12]. Note that NP-hardness for $s = 2$ does not directly imply NP-hardness for all values $s \geq 2$ (for example, the case $s = 5$ is trivially solvable on grid graphs since they have maximum degree four). We observe that P_3-Partition is typically not easier than Star Partition for $s \geq 3$. An exception to this rule is the class of split graphs (which are chordal), where P_3-Partition is polynomial-time solvable but Star Partition is NP-hard for any constant value $s \geq 3$.

Preliminaries. We assume basic familiarity with standard graph classes [6, 17]. Definitions of the graph classes are provided when first studied in this paper. We call the complete bipartite graph $K_{1,s}$ an *s-star*. For a graph $G = (V, E)$, an *s-star partition* is a set of $k := |V|/(s+1)$ pairwise disjoint vertex subsets $V_1, V_2, \ldots, V_k \subseteq V$ with $\bigcup_{1 \leq i \leq k} V_i = V$ such that each induced subgraph $G[V_i]$ contains an s-star as a (not necessarily induced) subgraph. We refer to the vertex sets V_i as *stars*, even though the correct description of a star would be an *arbitrary $K_{1,s}$-subgraph of $G[V_i]$*. P_3-Partition is the special case of Star Partition with

$s = 2$. Without loss of generality, we assume throughout the paper that the input graph G is connected (otherwise, we can solve the partition problem separately for each connected component of G). We denote by $n := |V|$ the number of vertices and by $m := |E|$ the number of edges in a graph $G = (V, E)$. For a vertex $v \in V$, we denote by $N[v] := \{u \in V \mid \{u, v\} \in E\} \cup \{v\}$ the *closed neighborhood* of v.

Article outline. The article is structured into one section per graph class. Herein, we first present the results on graph classes with polynomial-time algorithms and then head over to the graph classes with NP-hardness results. Each section gives a formal definition of the graph class it considers. Section 2 considers interval graphs and their subclasses unit interval graphs and trivially perfect graphs. Section 3 provides a polynomial-time algorithm for cographs, Section 4 for bipartite permutation graphs. Section 5 marks the boundary between tractability and NP-hardness: it shows that P_3-Partition is polynomial-time solvable on split graphs, while Star Partition is NP-hard. Section 6 shows that P_3-Partition is NP-hard on grid graphs and, finally, Section 7 shows it for chordal graphs.

2 Interval graphs

In this section, we present algorithms that solve Star Partition on unit interval graphs and on trivially perfect graphs in linear time, and a simple greedy algorithm that solves P_3-Partition on interval graphs in quasilinear time.

An *interval graph* is a graph whose vertices one-to-one correspond to intervals on the real line such that there is an edge between two vertices if and only if their representing intervals intersect. Interval graphs naturally occur in many scheduling applications [5, 19]. In a *unit interval graph*, all representing intervals are open and have the same length, while in a *trivially perfect graph*, any two representing intervals are either disjoint or one is properly contained in the other.

2.1 Star Partition on unit interval graphs

The restricted structure of unit interval graphs allows us to solve Star Partition using a simple greedy approach, which yields the following result.

Theorem 1. Star Partition *is solvable in $O(n + m)$ time on unit interval graphs.*

The general idea behind the algorithm for Theorem 1 is to order the vertices in such a way that we can repeatedly select the $s + 1$ leftmost vertices to form an s-star and then delete them. If, at some point, the $s + 1$ leftmost vertices do not contain an s-star, then it can be shown that the graph cannot be partitioned into s-stars. We order the vertices according to a so-called bicompatible elimination order:

Definition 1 ([24]). For a graph $G = (V, E)$, a *bicompatible elimination order* is an ordering $\sigma\colon V \to \{1, \dots, n\}$ such that, for each vertex $v \in V$,

$$\text{the set } N_l[v] \coloneqq \{u \in N[v] \mid \sigma(u) \leq \sigma(v)\} \text{ of its left neighbors and}$$
$$\text{the set } N_r[v] \coloneqq \{u \in N[v] \mid \sigma(u) \geq \sigma(v)\} \text{ of its right neighbors}$$

each form a clique in G.

A graph is a unit interval graph if and only if it allows for a bicompatible elimination order [24]. Our algorithm will exploit the following property of bicompatible elimination orders:

Lemma 1 ([3]). *Let $G = (V, E)$ be a connected unit interval graph and σ be a bicompatible elimination order for G. Then, for all $\{u, v\} \in E$ with $\sigma(u) < \sigma(v)$, the set $\{w \in V \mid \sigma(u) \leq \sigma(w) \leq \sigma(v)\}$ induces a clique in G.*

We are now ready to prove Theorem 1.

Proof of Theorem 1. Given a unit interval graph $G = (V, E)$ with $n \coloneqq |V|$ and $m \coloneqq |E|$, we can compute in linear time a bicompatible elimination order σ [24]. Moreover, we can assume G to be connected, thus making Lemma 1 applicable. For a subset $V' \subseteq V$ let $\mathrm{r}(V') \coloneqq \arg\max_{v \in V'} \sigma(v)$ denote the rightmost vertex in V' with respect to σ.

Now, we greedily partition G into s-stars starting with the first (with respect to σ) $s+1$ vertices $v_1, \dots, v_{s+1}$ with $\sigma(v_1) < \dots < \sigma(v_{s+1})$. If $G[\{v_1, \dots, v_{s+1}\}]$ does not contain an s-star, then we answer "no". Otherwise, we delete $v_1, \dots, v_{s+1}$ from G and continue on the remaining graph. If we end up with the empty graph, then we have found a partition of G into s-stars and answer "yes".

Obviously, the algorithm requires $O(n + m)$ time since checking whether an induced subgraph $G[V']$ with $|V'| = s$ vertices contains an s-star runs in $O(|G[V']|)$ time and after the check we delete the set V' from the graph. In this way, we touch each vertex at most once and each edge at most twice.

It remains to show that this procedure is correct. To this end, we show that if G admits an s-star partition, then G also admits an s-star partition P' with $S \coloneqq \{v_1, \dots, v_{s+1}\} \in P'$ (note that $v_1, \dots, v_{s+1}$ are the first $s+1$ vertices). Let P be a partition of G into s-stars such that $\{v_1, \dots, v_{s+1}\} \notin P$, that is, the first $s+1$ vertices are not grouped into one star but distributed among several stars. Then, let $S_1, \dots, S_\ell \in P$, $2 \leq \ell \leq s+1$, be the stars that contain at least one vertex from S, that is, $S \subseteq \bigcup_{i=1}^{\ell} S_i$ and $S_i \cap \{v_1, \dots, v_{s+1}\} \neq \emptyset$ for $1 \leq i \leq \ell$, and assume that $v_1 \in S_1$. Further, let c_i denote the center vertex of S_i for $1 \leq i \leq \ell$. Note that $\sigma(\mathrm{r}(S_1)) > s+1$, which implies $S \subseteq N[c_1]$. Since $N_l[c_1]$ and $N_r[c_1]$ are cliques, it follows that $G[S]$ contains an s-star that could participate in an s-star partition if the remaining vertices in $S' \coloneqq \bigcup_{i=1}^{\ell} S_i \setminus S$ can also be partitioned into s-stars. To verify that this is possible, observe first that the number $|S'| = (\ell - 1)(s+1)$ of the remaining vertices is again divisible by $s+1$.

We now show that we can greedily partition S' into stars, because S' consists of two cliques such that there is a vertex of the first clique that is adjacent to

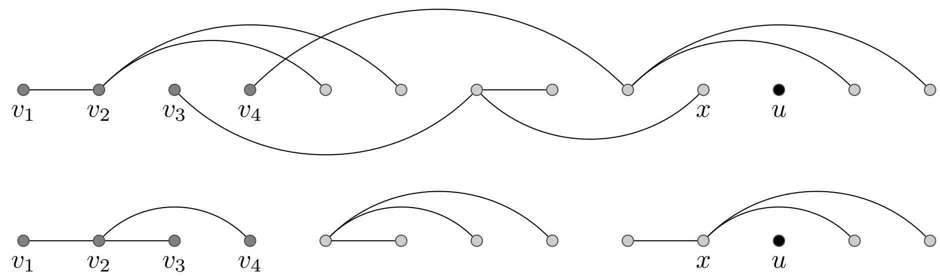


Figure 2: Example of a 3-star partition of a unit interval graph with vertices ordered according to a bicompatible elimination order from left to right. Only the edges and vertices of the first three stars as well as the rightmost neighbor $u := \mathrm{r}(N(v_4))$ of v_4 (black) are shown. Top: $v_1, \ldots, v_4$ are not grouped together into a star in the partition. Bottom: A possible rearrangement of the 3-stars as described in the proof of Theorem 1. It is always possible to group $v_1, \ldots, v_4$ into a 3-star.

all vertices of the second clique. To show this, we utilize the following claim, which describes the relative position of the center c_i of star S_i and the rightmost neighbor of v_{s+1}:

Claim 1. *For all $1 \leq i \leq \ell$, the center c_i of star S_i satisfies that $\sigma(c_i) \leq \sigma(\mathrm{r}(N[v_{s+1}]))$.*

Proof of Claim 1. Suppose towards a contradiction that $\sigma(c_i) > \sigma(\mathrm{r}(N[v_{s+1}]))$. Then $c_i \neq \mathrm{r}(N[v_{s+1}])$ and thus, $\{c_i, v_{s+1}\} \in E$ since c_i is adjacent to at least one vertex from $v_1, \ldots, v_{s+1}$ and Lemma 1 holds. Hence, $c_i \in N_r[v_{s+1}]$, which contradicts $\sigma(c_i) > \sigma(\mathrm{r}(N[v_{s+1}]))$. (of Claim 1) $\diamond$

Now, let $u := \mathrm{r}(N[v_{s+1}])$ denote the rightmost neighbor of v_{s+1}. It holds that $S' \subseteq N[u]$. This can be seen as follows: For a vertex $v' \in S'$, either $s + 1 < \sigma(v') \leq \sigma(u)$ or $\sigma(u) < \sigma(v')$ holds. In the first case, Lemma 1 implies that $\{v', u\} \in E$ since $\{v_{s+1}, u\} \in E$. For the second case, let S_i, $1 \leq i \leq \ell$ be the star containing v'. Then, by Claim 1, it follows that S_i's center c_i satisfies $\sigma(c_i) \leq \sigma(u)$. If $\sigma(u) < \sigma(v')$, then Lemma 1 implies $\{u, v'\} \in E$ since $\{c_i, v'\} \in E$.

Now, consider the vertex $x := \mathrm{r}(S' \cap N_l[u])$, that is, the rightmost vertex in S' that is a left neighbor of u. Clearly, from Claim 1 it follows that $\sigma(c_i) \leq \sigma(x)$ holds for every star center c_i, $1 \leq i \leq \ell$, since otherwise c_i were to be ordered between x and u, and is hence, a left neighbor of u—a contradiction to x being the rightmost left neighbor of u in S'. Thus, x is adjacent to all vertices in $S' \cap N_r[u]$ due to Lemma 1. The vertices in $S' \cap N_l[u]$ are also adjacent to x as they induce a clique which includes x. Moreover, $S' \cap N_r[u]$ also induces a clique. Therefore, we simply partition the vertices in S' from right to left (with respect to σ) into s-stars. This is always possible since x is connected to all vertices in both cliques $S' \cap N_r[u]$ and $S' \cap N_l[u]$. Figure 2 depicts an example of the rearranged partition. $\square$

2.2 Star Partition on trivially perfect graphs

Recall that an *interval graph* is a graph whose vertices correspond directly to intervals on the real line, and there is an edge between two vertices if their intervals intersect. A *trivially perfect* (also known as *quasi-threshold*) graph is an interval graph representable such that any two intervals are either disjoint or one is properly contained in the other.

In order to solve Star Partition in linear time on trivially perfect graphs, we will make use of the linear-time computable *(rooted) tree representation* of connected trivially perfect graphs [33]:

Definition 2 (Rooted tree representation)**.** Let $G = (V, E)$ be a connected trivially perfect graph. Let $T(G)$ be the directed graph on the vertex set V that contains an arc (v, w) if and only if a) the interval representing v contains the interval representing w, and b) there is no other vertex u such that its representing interval contains the interval representing w and is contained in the interval representing v.

By definition of trivially perfect graphs, $T(G)$ is a directed tree having a unique vertex, the *root*, with in-degree zero. We call $T(G)$ the *rooted tree representation* of G.

If, in $T(G)$, a vertex u lies on the directed path from the root to a vertex v, or equivalently, if there is a directed path from u to v, we call u *ancestor of* v and v *descendant of* u. The *depth* of a vertex is the length of the path from the root to this vertex.

Definition 2 is illustrated in Figure 3. It is crucial to observe the equivalence of the adjacency of two vertices and their ancestor-descendant relation:

Observation 1. The graph G contains an edge $\{p, q\}$ if and only if p is either an ancestor or a descendant of q in $T(G)$.

Proof. Since G is a trivially perfect graph, G contains an edge $\{p, q\}$ if and only if either $p' \subset q'$ or $q' \subset p'$ where p' and q' are the representing intervals of p and q, respectively. If $p' \subset q'$, then there is a directed path from q to p in $T(G)$. Conversely, $q' \subset p'$ implies that there is a directed path from p to q in $T(G)$. By the definition of ancestors and descendants, p is either an ancestor or a descendant of q. $\square$

Also the following is easy to observe:

Observation 2. If there are three vertices p, q, r such that G contains an edge $\{q, r\}$ and p is an ancestor of q in $T(G)$, then G also contains an edge $\{p, r\}$.

Proof. By Observation 1, $\{q, r\}$ being an edge in G implies that q is either an ancestor or a descendant of r. If q is an ancestor of r, then p is also an

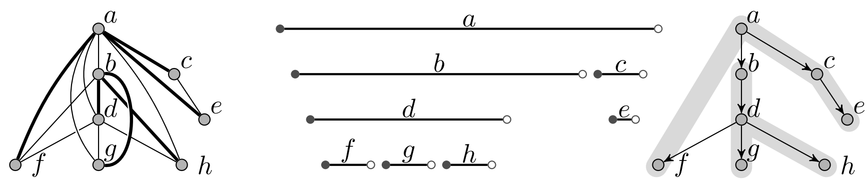


Figure 3: An example of a trivially perfect graph and its partition into stars $K_{1,3}$. Left: The trivially perfect graph with eight vertices partitioned into stars $K_{1,3}$ (bold). Middle: The interval representation. Right: The rooted tree representation with the corresponding partition in shaded gray.

ancestor of r, implying that G contains the edge $\{p, r\}$. Otherwise, the interval q' representing q is properly contained in the interval r' representing r. Since q' is also properly contained in the interval p' representing p (p is an ancestor of q), we obtain that p' and r' are not disjoint. By the definition of trivially perfect graphs, $\{p, r\}$ is contained in G. □

Before presenting our algorithm, we show that we may assume that star partitions of G have a very restricted structure with respect to $T(G)$. First of all, we can assume that the center of a star is an ancestor of all its leaves:

Observation 3. Let G be a trivially perfect graph with n vertices. If G allows for an s-star partition $\{V_1, V_2, \ldots, V_{n/(s+1)}\}$, then each $G[V_i]$, $1 \leq i \leq n/(s+1)$, contains an s-star whose center vertex c_i is an ancestor of all vertices $V_i \setminus \{c_i\}$ in $T(G)$.

Proof. Let c_i be the center of an s-star in $G[V_i]$. By Observation 1, for each vertex $u \in V_i \setminus \{c_i\}$, c_i is either an ancestor or a descendant of u. If c_i is not an ancestor of all vertices in $V_i \setminus \{c_i\}$, then let a be an ancestor of c_i in V_i with smallest depth. Clearly, since a is an ancestor of the center c_i, a is adjacent to all vertices of V_i in G by Observation 2. It remains to show that a is also an ancestor of all vertices in $V_i \setminus \{a\}$. Suppose, towards a contradiction, that there is a vertex $u \in V_i \setminus \{a\}$ such that a is not an ancestor of u. By Observation 1, u is an ancestor of a and is, hence, an ancestor of c_i with a smaller depth than a—a contradiction. □

Our next observation is that we can assume that no star center is contained in a subtree T' of a rooted tree representation $T(G)$ if T' contains "too few" vertices. Therefore, of special interest to use are subtrees of $T(G)$ that can contain s-stars but of which no subtree can:

Definition 3 (Center barrier)**.** A subtree X of a rooted tree representation $T(G)$ is a *center barrier for s-stars $K_{1,s}$* if X has at least $s+1$ vertices and each proper subtree of X has at most s vertices.

The term "center barrier" is chosen since we can assume that no subtree of the center barrier contains an s-star center. Note that any (connected) rooted tree representation with at least $s+1$ vertices contains a center barrier.

Observation 4. Let X be a center barrier for s-stars in a rooted tree representation $T(G)$, and P be an s-star partition of G. Then, for any $V_i \in P$ that shares a vertex with X, the graph $G[V_i]$ contains a star whose center is the root of X or an ancestor of that root.

Proof. By the definition of center barriers, X is a subtree of $T(G)$. Let x be its root. By Observation 3, $G[V_i]$ contains a star whose center c is the ancestor of all vertices in $V_i \setminus \{c\}$. If $x \neq c$, then let $w \in V_i \cap X$, which exists by assumption. Observe that x is an ancestor of w. Since G contains the edge $\{w, c\}$ (c is a center for $V_i \ni w$), by Observation 2 and $\{c, w\}$ being an edge in G, G also contains an edge $\{x, c\}$. Then, by Observation 1, c is either an ancestor or a descendant of x. If c is a descendant of x, then the subtree of $T(G)$ rooted at c (which contains V_i) is a proper subtree of X. This is impossible since the proper subtrees of X have at most s vertices. Hence, $x \neq c$ implies that c is an ancestor of x. ☐

Finally, we show that there exists a feasible star partition where each star consists only of vertices from center barriers.

Lemma 2. *Let G be a trivially perfect graph allowing for an s-star partition, let X be a center barrier for s-stars in the rooted tree representation $T(G)$, and let x be the root of X. Then, G admits a star partition $\mathcal{P}$ with $S \cup \{x\} \in \mathcal{P}$, where S consists of s arbitrary vertices of $X \setminus \{x\}$.*

Proof. Let $\mathcal{Q}$ be a star partition of G and let S consist of s arbitrary vertices of $X \setminus \{x\}$. By Observation 3, we can assume the center c of a vertex subset $V_i \in \mathcal{Q}$ to be the one being the ancestor of all other vertices in $V_i \setminus \{c\}$. If $S \cup \{x\} \in \mathcal{Q}$, then the partition we are searching for is $\mathcal{P} := \mathcal{Q}$. Otherwise, we show how to transform the partition $\mathcal{Q}$ into a new partition $\mathcal{P}$ containing $S \cup \{x\}$. We repeatedly exchange the vertices of two vertex sets in $\mathcal{Q}$ until

$$\text{the modified partition } \mathcal{Q}' \text{ contains a set } V_w \text{ such that } S \subsetneq V_w. \tag{1}$$

Finally, we set $\mathcal{P} := (\mathcal{Q}' \setminus \{V_w, V_u\}) \cup \{(V_u \setminus \{x\}) \cup \{w\}, (V_w \setminus \{w\}) \cup \{x\}\}$, where w is the center of V_w and V_u is the vertex set with center u in $\mathcal{Q}'$ such that $x \in V_u$. One can verify that both $(V_w \setminus \{w\}) \cup \{x\}$ and $(V_u \setminus \{x\}) \cup \{w\}$ contain an s-star, implying that $\mathcal{P}$ is indeed an s-star partition for G: on the one hand, it is easy to see that $G[(V_w \setminus \{w\}) \cup \{x\}]$ contains an s-star with center x since x is the ancestor of all vertices in $V_w \setminus \{w\} = S$. On the other hand, the fact that w and x are both ancestors of all vertices in S (for w, this holds since $S \subsetneq V_w$ and by Observation 3) implies that w and x are adjacent in G (Observations 1 and 2). Since u is an ancestor of x, by Observation 2, we have that u and w are adjacent in G. This implies that u is either an ancestor or a descendant of w (Observations 1 and 2). In any case, $G[(V_u \setminus \{x\}) \cup \{w\}]$ contains an s-star with star center either u or w.

Now, in the remainder of the proof, we aim at transforming the partition $\mathcal{Q}$ into a new partition $\mathcal{Q}'$ fulfilling Property (1). To this end, among all vertex subsets $V_i \in \mathcal{Q}$ with $V_i \cap S \neq \emptyset$, we let V_y be the one with the center y closest to x

with respect to $T(G)$ (possibly, $y = x$). By assumption, $|S \setminus V_y| \geq 1$. Thus, let $V_z \in \mathcal{Q}$ be another subset with center z that contains at least one vertex from S. By Observation 4, z is an ancestor of x. By the selection of y, z is also an ancestor of y. Thus, in the graph G, z is adjacent to every vertex in V_y and y is adjacent to all vertices in $V_z \cap S$ since y is either x or an ancestor of x (Observations 1 and 4). Thus, by setting $V'_y \coloneqq (V_y \setminus Y) \cup (V_z \cap S)$ and $V'_z \coloneqq (V_z \setminus S) \cup Y$, where $Y \subsetneq V_y \setminus S$ is an arbitrary size-($|V_z \cap S|$) subset, we obtain a new valid partition $(\mathcal{Q} \setminus \{V_y, V_z\}) \cup \{V'_y, V'_z\}$ such that V'_y shares more vertices with S than V_y does. Note that Y exists since $|S| + 1 = |V_y| = |V_z|$. Repeating the above procedure at most $s - 1$ times results in a partition satisfying (1). □

Based on Lemma 2, we now give a linear-time algorithm computing an s-star partition (if existent) of a given trivially perfect graph.

Theorem 2. Star Partition *can be solved in $O(n + m)$ time on trivially perfect graphs.*

Proof. Let G be a connected trivially perfect graph. Construct a tree representation $T(G)$ of G in linear time [33]. Furthermore, construct in linear time a directed acyclic graph $D(G)$ from G which has the same vertex set as G, and for each edge $\{u, v\} \in E(G)$, there is an arc (u, v) in $D(G)$ if and only if the degree of u is larger than the degree of v in G.

Due to Lemma 2, if G admits an s-star partition, then G also admits an s-star partition $\mathcal{P} \coloneqq \{V_1, V_2, \ldots, V_{|V|/(s+1)}\}$ such that for each $i \in \{1, 2, \ldots, |V|/(s + 1)\}$, V_i is contained in a center barrier of the rooted tree representation for the graph $G[V \setminus (\bigcup_{j=1}^{i-1} V_j)]$ resulting from G by deleting the vertices in $\bigcup_{j=1}^{i-1} V_j$. Hence, it is sufficient to recursively search for a center barrier X for s-stars, and delete the root of X and s arbitrary remaining vertices from X (these deleted vertices form a subset in the s-star partition). If, at some point, there is no center barrier, then there are less than $s + 1$ remaining vertices; hence, G cannot allow for an s-star partition.

To realize the above algorithm in linear time, we traverse $T(G)$ in a depth-first post-order way. If, in $D(G)$, the current vertex u has at least s non-marked (out-going) neighbors, then we *mark* this vertex u and s (arbitrary) non-marked (out-going) neighbors of u. Otherwise, we do nothing. We answer yes if all vertices in $D(G)$ are marked after traversing the whole tree, and no otherwise. Since we mark, in $D(G)$, each vertex and each of its out-going neighbor at most once, and since we traverse each vertex in $T(G)$ at most once, by the construction of $D(G)$ and $T(G)$, the total running time is $O(|V(G)| + |E(G)|)$. □

2.3 P_3-Partition on interval graphs

While it might not come as a surprise that Star Partition can be solved efficiently on unit interval graphs using a greedy strategy, this is far from obvious for general interval graphs even when $s = 2$. The obstacle here is that two intervals arbitrarily far apart from each other may eventually be required to form a P_3 in the solution. Indeed, the greedy strategy we propose to overcome this

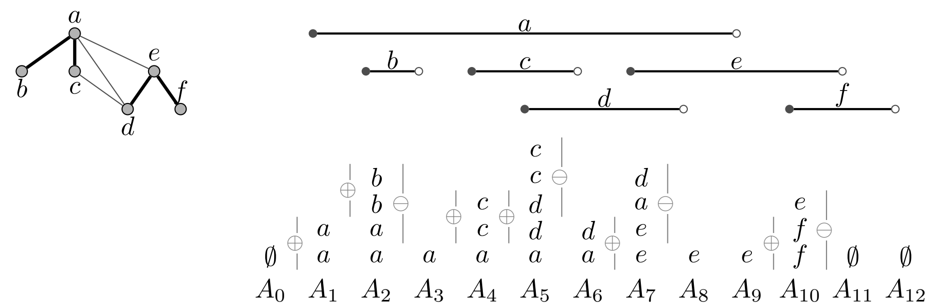


Figure 4: Left: An interval graph with six vertices and a P_3-partition $\mathcal{P}$ (bold). Right: Interval representation of this graph and successive token lists $A_0, \ldots, A_{12}$ computed by Algorithm 1 (additions and deletions are marked with $\oplus$ and $\ominus$).

obstacle is naive in the sense of allowing wrong choices that can be corrected later. Note that, while we can solve the more general Star Partition in polynomial time on subclasses of interval graphs like unit interval graphs and trivially perfect graphs (see previous subsections), we are not aware of a polynomial-time algorithm for Star Partition with $s \geq 3$ on interval graphs.

Overview of the algorithm. The algorithm is based on the following analysis of a P_3-partition of an interval graph. Each P_3 contains a *center* and two *leaves* connected to the center via their incident edges called *links*. We associate with each interval two so-called *tokens*. We require that the link between a leaf and a center *consumes* both of the leaf's tokens (such that a leaf can be associated to only one link) and one token of the center (which can thus be linked to two leaves).

The algorithm examines the *event points* (start and end points of intervals) of an interval representation in increasing order. We consider that a link $\{x, y\}$ consumes three tokens of x and y as soon as one of the two intervals ends. Intuitively, a graph is a no-instance if, at some point, an interval with one or two remaining tokens ends, but there are not enough tokens of other adjacent intervals to *create* a link. Note that a link consumes three tokens. A graph is a yes-instance if the number of tokens is always sufficient.

The algorithm works according to the following two rules: when an interval starts, its two tokens are added to a list; when an interval with remaining tokens ends, then three tokens are deleted from this list. Only tokens of the earliest-ending intervals will be deleted (this choice may not directly translate into a "sane" solution, with each link consuming tokens from only two intervals, but it turns out not to be a problem). The algorithm is sketched in Algorithm 1. Figure 4 shows an example instance and the list of tokens maintained by the algorithm. Note that a token of an interval x is simply represented by a copy of interval x itself. We now introduce the necessary formal definitions.

Definition 4. Let $G = (V, E)$ be a fixed interval graph. We assume that any vertex $u \in V$ represents a right-open interval $u = [\mathrm{start}(u), \mathrm{end}(u)[$ with integer

Algorithm 1: P_3-partition of an interval graph

Input: An interval representation of an interval graph with pairwise distinct event points in $\{1, \dots, 2n\}$.

Output: `true` if the graph allows for a P_3-partition, otherwise `false`.

$A_0 \leftarrow$ empty token list $\emptyset$;
for $t \leftarrow 1$ *to* $2n$ **do**
    if $t = \text{start}(x)$ **then** $A_t \leftarrow A_{t-1} \oplus (x, x)$;
    if $t = \text{end}(x)$ **then**
        if $x \notin A_{t-1}$ **then** $A_t \leftarrow A_{t-1}$;
        else if $\|A_{t-1}\| < 3$ **then return** *false*;
        else
            $(x, y, z) \leftarrow$ top three elements of A_{t-1} (intervals ending first);
            $A_t \leftarrow A_{t-1} \ominus (x, y, z)$;
        end
    end
end
return *true*;

end points $\text{start}(u) < \text{end}(u)$. Moreover, without loss of generality, each position in $\{1, \dots, 2n\}$ corresponds to exactly one event.

Let $\mathcal{P}$ be a P_3-partition and $P = \{x, y, z\} \in \mathcal{P}$ with $\text{end}(x) < \text{end}(y) < \text{end}(z)$, we write $\text{rank}_{\mathcal{P}}(x) = 1$, $\text{rank}_{\mathcal{P}}(y) = 2$, and $\text{rank}_{\mathcal{P}}(z) = 3$ (we omit the subscript when there is no ambiguity). Moreover, we call the element among $\{y, z\}$ having the earliest start point the *center* of P. The other two elements of P are called *leaves*. Note that the center of P intersects both leaves.

A *token list* Q is a list of intervals $(q_1, \dots, q_k)$ sorted in decreasing order of their end points ($\text{end}(q_i) \geq \text{end}(q_j)$ for $1 \leq i \leq j \leq k$). To avoid confusion with the left-to-right sequence of event points, we consider the list to be written vertically, with the earliest-ending interval on top. We write $\|Q\|$ for the length of Q, $\emptyset$ for the empty token list, and $x \in Q$ if interval x appears in Q. We now define *insertion* $\oplus$, *deletion* $\ominus$, and *comparison* $\preccurlyeq$ of token lists: $Q \oplus (x_1, \dots, x_l)$ is the token list obtained from Q by inserting intervals $x_1 \dots, x_l$ so that the list remains sorted. For $x \in Q$, the list $Q \ominus x$ is obtained by deleting one copy of x from Q (otherwise, $Q \ominus x = Q$); and $Q \ominus (x_1, \dots, x_l) = Q \ominus x_1 \ominus \dots \ominus x_l$. We write $(q_1, \dots, q_k) \preccurlyeq (q'_1, \dots, q'_{k'})$ if $k \leq k'$ and $\forall i \in \{1, \dots k\} : \text{end}(q_i) \leq \text{end}(q'_i)$.

Let $\mathcal{P}$ be a P_3-partition. We define tokens($\mathcal{P}$) as a tuple $(T_0, T_1, \dots, T_{2n})$ of $2n + 1$ token lists such that $T_0 \coloneqq \emptyset$ and for $t \in \{1, \dots, 2n + 1\}$,

- if $t = \text{start}(x)$, then $T_t \coloneqq T_{t-1} \oplus (x, x)$,
- if $t = \text{end}(x)$, then let $P \coloneqq \{x, y, z\}$ be the P_3 in $\mathcal{P}$ containing x and
  - if $\text{rank}(x) = 1$, then $T_t \coloneqq T_{t-1} \ominus (x, x, c)$ where c is the center of P,
  - if $\text{rank}(x) = 2$, then $T_t \coloneqq T_{t-1} \ominus (x, x, y, y, z, z)$,
  - if $\text{rank}(x) = 3$, then $T_t \coloneqq T_{t-1}$.

Note that in Figure 4, each token list T_t for $\mathcal{P}$ is equal to the respective A_t, except for $T_6 = (d, d)$ and $T_7 = (e, e, d, d)$.

To compare the token lists generated by Algorithm 1 to those induced by a P_3-partition, we show a few properties for both types of lists.

Property 1. Let $\mathcal{P}$ be a P_3-partition with $\text{tokens}(\mathcal{P}) = (T_0, T_1, \ldots, T_{2n})$ and let x be an interval with $t \coloneqq \text{end}(x)$. Then, one of the following is true:

i) $x \in T_{t-1}$, $\|T_{t-1}\| \geq 3$, and $\|T_t\| = \|T_{t-1}\| - 3$ or

ii) $x \notin T_{t-1}$ and $T_t = T_{t-1}$.

Moreover, in both cases, $x \notin T_t$.

Proof. Let $P \in \mathcal{P}$ be the P_3 containing x. Depending on the rank of x, we prove that either case (i) or (ii) applies.

If $\text{rank}(x) = 1$, then x is not the center of P. Let c be the center of P. Since c is adjacent to x, it follows that $\text{start}(c) < t$. Since x ranks first, T_{t-1} contains twice both elements x and c. Hence, $\|T_{t-1}\| \geq 4$, and from the definition of $T_t = T_{t-1} \ominus (x, x, c)$ it follows that $\|T_t\| = \|T_{t-1}\| - 3$, we are thus in case (i). Moreover, only one copy of c remains in T_t.

If $\text{rank}(x) = 2$ and x is not the center, then let c be the center of P and y be the interval of the first rank in P. From the reasoning above, it follows that T_{t-1} contains once c and twice x but no y, implying that $\|T_{t-1}\| \geq 3$. As $T_t = T_{t-1} \ominus (x, x, c, c, y, y)$, this implies that $\|T_t\| = \|T_{t-1}\| - 3$: we are in case (i).

If $\text{rank}(x) = 2$ and x is the center, then let $P = \{x, y, z\}$ such that y ranks first and z ranks third. Using the same reasoning as before it follows that T_{t-1} contains x once and z twice, but not y, implying that $\|T_{t-1}\| \geq 3$. As $T_t = T_{t-1} \ominus (x, x, y, y, z, z)$, this implies that $\|T_t\| = \|T_{t-1}\| - 3$: we are in case (i).

Finally, if $\text{rank}(x) = 3$, then T_{t-1} does not contain x (the last copies have been removed when the rank-2-interval ended), and $T_t = T_{t-1}$: we are in case (ii).

The fact that $x \notin T_t$ is clear in each case (all copies are removed when $x \in T_{t-1}$, none is added). □

Property 2. For any A_t defined by Algorithm 1 and $x \in A_t$, it holds that $\text{start}(x) \leq t < \text{end}(x)$. For any P_3-partition $\mathcal{P}$ with $\text{tokens}(\mathcal{P}) = (T_0, T_1, \ldots, T_{2n})$ and $x \in T_t$, it holds that $\text{start}(x) \leq t < \text{end}(x)$.

Proof. An element x is only added to a token list A_t or T_t when $t = \text{start}(x)$, so the inequality $\text{start}(x) \leq t$ is trivial in both cases. Consider now an interval x and $t \coloneqq \text{end}(x)$. We show that neither A_t nor T_t contain x, which suffices to complete the proof.

The fact that $x \notin T_t$ is already proven in Property 1. Moreover, if $x \notin A_{t-1}$, then $x \notin A_t$ follows obviously.

Now, assume that $x \in A_{t-1}$. We inductively apply Property 2 to obtain that for any $y \in A_{t-1}$, we have $t - 1 < \text{end}(y)$ (note that the property is trivial for A_0). Hence, x is the interval with the earliest end point in A_{t-1} (i. e., the interval on top) and all of its copies(at most two) are removed from A_{t-1} to obtain A_t in line 9 of Algorithm 1. It follows that $x \notin A_t$. □

Property 3. Let $Q = (q_1, \ldots, q_k)$ and $Q' = (q'_1, \ldots, q'_{k'})$ be two token lists such that $Q \preccurlyeq Q'$. Then for any $q_i \in Q$, $Q \ominus q_i \preccurlyeq Q' \ominus q'_{k'}$ and for any interval x, $Q \oplus x \preccurlyeq Q' \oplus x$.

Proof. For both insertion and deletion, the size constraint is clearly maintained (both list lengths respectively increase or decrease by 1). It remains to compare pairs of elements with the same index in both lists (such pairs are said to be *aligned*).

For the deletion case, q_i is removed from Q. For any $j \neq i$, q_j is now aligned with either q'_j (if $j < i$) or q'_{j-1} (if $j > i$). We have $\mathrm{end}(q_j) \leq \mathrm{end}(q'_j)$ since $Q \preccurlyeq Q'$ and $\mathrm{end}(q'_j) \leq \mathrm{end}(q'_{j-1})$ since Q' is sorted. Hence, q_j is aligned with an interval in Q' ending no later than q_j itself.

We now prove the property for the insertion of x in both Q and Q'. An element q of Q or Q' is said to be *shifted* if it is higher than the insertion point of x (assuming that already-presentcopies of x in Q or Q' are not shifted), this is equivalent to $\mathrm{end}(q) < \mathrm{end}(x)$. Note that if some q'_i is shifted but q_i is not, then $\mathrm{end}(q'_i) < \mathrm{end}(x) \leq \mathrm{end}(q_i)$, a contradiction to $\mathrm{end}(q'_i) \geq \mathrm{end}(q_i)$. This implies that the insertion point of x in Q' is not lower than the insertionpoint in Q.

Let q be an interval of $Q \oplus x$, now aligned with some q' in $Q' \oplus x$. We prove that $\mathrm{end}(q') \geq \mathrm{end}(q)$. Assume first that $q = x$, then either $q' = x$, in which case trivially $\mathrm{end}(q') \geq \mathrm{end}(q)$, or $q' \neq x$. Then, q' cannot be shifted (since x's insertion point is not lower in Q' thanin Q), and $\mathrm{end}(q') \geq \mathrm{end}(x) = \mathrm{end}(q)$.

Assume now that $q \neq x$. Then, $q = q_i$ for some i. With $Q \preccurlyeq Q'$, we have $\mathrm{end}(q) \leq \mathrm{end}(q'_i)$. If $q' = q'_i$ then we directly have $\mathrm{end}(q) \leq \mathrm{end}(q')$. Otherwise, exactly one of q_i and q'_i must be shifted. It cannot be q'_i (the insertion point of x is not lower in Q' than in Q), henceq_i is shifted and q'_i is not. In Q we have $\mathrm{end}(q_i) < \mathrm{end}(x)$, and in Q' interval q'_i must be placed directly below q' and both cannot occur higher than x (note that $q' = x$ ispossible), thus we have $\mathrm{end}(q'_i) \geq \mathrm{end}(q') \geq \mathrm{end}(x)$. Overall, we indeed have $\mathrm{end}(q) \leq \mathrm{end}(q')$. □

Using the proven properties, we can put the token lists defined by a P_3-partition into relation with the token lists generated by Algorithm 1.

The following two lemmas state that, on the one hand, if there is a P_3-partition, then each token list created by Algorithm 1 is comparable with the corresponding T_t, hence it always contains enough tokens to create the next list, up to A_{2n}, and answer "`true`" in the end. On the other hand, if the algorithm returns "`true`", then it is indeed possible to construct a P_3-partition using (indirectly) the triples of intervals removed from the token list to create the links.

Lemma 3. *If an interval graph G has a P_3-partition $\mathcal{P}$, then, for all $0 \leq t \leq 2n$, Algorithm 1 defines list A_t with $T_t \preccurlyeq A_t$ and $\|T_t\| - \|A_t\| \equiv 0 \pmod 3$, where* $\mathrm{tokens}(\mathcal{P}) = (T_0, T_1, \ldots, T_{2n})$.

Proof. We show by induction that for any position t, $0 \leq t \leq 2n$, the algorithm defines a list A_t with $T_t \preccurlyeq A_t$ and $\|T_t\| - \|A_t\| \equiv 0 \pmod 3$.

For $t = 0$, Algorithm 1 defines list $A_0 = \emptyset$, and $T_0 = \emptyset \preccurlyeq A_0$. Consider now some $0 < t \leq 2n$, and assume that the induction property is proven for $t - 1$.

If an interval x starts at position t, then $x \notin T_{t-1}$, $x \notin A_{t-1}$, $T_t = T_{t-1} \oplus (x, x)$, and Algorithm 1 defines $A_t \coloneqq A_{t-1} \oplus (x, x)$. Then property $\|T_t\| - \|A_t\| \equiv 0 \pmod 3$ is trivially preserved, and Property 3 implies $T_t \preccurlyeq A_t$.

If an interval x ends at t, then we first show that Algorithm 1 defines A_t. Towards a contradiction, suppose that A_t is not defined. This means that $x \in A_{t-1}$ and $\|A_{t-1}\| \leq 2$. Then, $\|T_{t-1}\| \leq 2$ since $T_{t-1} \preccurlyeq A_{t-1}$, which implies $\|T_{t-1}\| = \|A_{t-1}\|$ (since $\|T_{t-1}\| - \|A_{t-1}\| \equiv 0 \pmod 3$). By Property 1, we must have $x \notin T_{t-1}$ and $T_t = T_{t-1}$ (the second case). Also, with $T_{t-1} \preccurlyeq A_{t-1}$, the top element x' in T_{t-1} must have $\mathrm{end}(x') \leq \mathrm{end}(x) = t$. By Property 2 and due to $x' \in T_{t-1}$, $\mathrm{end}(x') > t - 1$, i. e., $\mathrm{end}(x') = \mathrm{end}(x)$, and $x' = x$: a contradiction since $x \notin T_{t-1} = A_{t-1}$.

We have shown that Algorithm 1 defines A_t. Moreover, we have $\|T_t\| - \|T_{t-1}\| \in \{0, -3\}$ (Property 1), and $\|A_t\| - \|A_{t-1}\| \in \{0, -3\}$ (Algorithm 1), hence $\|T_t\| - \|A_t\| \equiv 0 \pmod 3$. Note also that $T_t \preccurlyeq T_{t-1}$ and $A_t \preccurlyeq A_{t-1}$.

If $x \notin A_{t-1}$, then $A_t = A_{t-1}$, and $T_t \preccurlyeq T_{t-1} \preccurlyeq A_{t-1} = A_t$.

If $x \in A_{t-1}$ and $x \notin T_{t-1}$, then $T_t = T_{t-1}$ (Property 1). Observe that $T_{t-1} \preccurlyeq A_{t-1}$ and, therefore, $\|T_{t-1}\| = \|A_{t-1}\|$ would imply the existence of an interval $u \in T_{t-1}$ with $\mathrm{end}(u) \leq \mathrm{end}(x)$. This is impossible since for all $u \in T_t = T_{t-1}$, by Property 2, $\mathrm{end}(u) > t = \mathrm{end}(x)$. Thus, one has $\|T_{t-1}\| < \|A_{t-1}\|$, which implies $\|T_{t-1}\| \leq \|A_{t-1}\| - 3$. Hence, since the *top* three elements of A_{t-1} are removed to obtainA_t, from $T_{t-1} \preccurlyeq A_{t-1}$ we conclude $T_{t-1} \preccurlyeq A_t$ and, in turn, $T_t \preccurlyeq A_t$.

If $x \in A_{t-1}$ and $x \in T_{t-1}$, then let (u, v, w) be the three deleted intervals from T_{t-1}, and (u', v', w') the topthree elements of A_{t-1} (which are removed to obtain A_t). Then, applying Property 3 three times, we obtain $T_{t-1} \ominus (u, v, w) \preccurlyeq A_{t-1} \ominus (u', v', w')$, i. e., $T_t \preccurlyeq A_t$. ☐

Before we move on to our last lemma for the interval graph algorithm, we introduce further notions necessary for constructing a P_3-partition from the list A_{2n} that our algorithm produces at step $2n$.

Definition 5 (Partial partition)**.** Given an interval x and a token list Q, we write $|Q|_x$ for the number of occurrences of interval x in Q. For $0 \leq t \leq 2n$, let $\mathcal{P} = \{V_1, \ldots, V_k\}$ be a partition of $\{u \in V \mid \mathrm{start}(u) \leq t\}$. Then $\mathcal{P}$ is called a *partial partition at* t if each V_j is either

- a singleton $\{x\}$, in which case $\mathrm{end}(x) > t$,
- an edge $\{x, y\}$, in which case $\max\{\mathrm{end}(x), \mathrm{end}(y)\} > t$,
- a triple $\{x, y, z\}$ containing a P_3.

Note that a P_3-partition of an interval graph corresponds to a partial partition at $t = 2n$. A partial solution $\mathcal{P}$ at t *satisfies* A_t if

- for any singleton $\{x\} \in \mathcal{P}$ we have $|A_t|_x = 2$,
- for any edge $\{x, y\} \in \mathcal{P}$ with $\mathrm{end}(x) < \mathrm{end}(y)$ we have $|A_t|_x = 0$ and $|A_t|_y = 1$, and
- for any triple $\{x, y, z\} \in \mathcal{P}$ we have $|A_t|_x = |A_t|_y = |A_t|_z = 0$.

Note that, for any $x \in A_t$, since $\mathrm{start}(x) \leq t < \mathrm{end}(x)$ (Property 2), it follows that x must be in a singleton or in an edge of any partial solution satisfying A_t. Moreover, for any t and $x, y \in A_t$ with $x \neq y$, intervals x and y intersect (there is an edge between them in the interval graph).

Lemma 4. *Let G be an interval graph such that Algorithm 1 returns* `true` *on G. Then G admits a P_3-partition.*

Proof. We prove by induction that for any t such that Algorithm 1 defines A_t, there exists a partial solution at t satisfying A_t.

For $t = 0$, the partial solution $\emptyset$ satisfies A_0. Assume now that for some $t \leq 2n$, Algorithm 1 defines A_t, and that there exists a partial solution $\mathcal{P}$ at $t-1$ satisfying A_{t-1}.

First, if $t = \mathrm{start}(x)$ for some interval x, then let $\mathcal{P}' := \mathcal{P} \cup \{\{x\}\}$. Thus, $\mathcal{P}'$ is now a partial solution at t (it partitions every interval with earlier starting point into singletons, edges and P_3s) which satisfies A_t since by construction of A_t by Algorithm 1, $|A_t|_x = 2$.

Now assume that $t = \mathrm{end}(x)$ with $x \notin A_{t-1}$. Then, in $\mathcal{P}$, either x is part of an edge $\{x, y\}$ with $\mathrm{end}(y) > t$, or x is part of a P_3. In both cases, $\mathcal{P}' := \mathcal{P}$ is a partial solution at t which satisfies $A_t = A_{t-1}$.

We now explore the case where $t = \mathrm{end}(x)$ with $x \in A_{t-1}$. Then, the top element of A_{t-1} must be x (no other interval $u \in A_{t-1}$ can have $t-1 < \mathrm{end}(u) \leq \mathrm{end}(x)$). Let y and z be the two elements below x in A_{t-1}. Then, by construction, $A_t = A_{t-1} \ominus (x, y, z)$ and $\mathrm{end}(x) \leq \mathrm{end}(y) \leq \mathrm{end}(z) \leq \mathrm{end}(u)$ for all $u \in A_t$. We create a partial solution $\mathcal{P}'$ at t depending on the number of occurrences of x, y, and z in A_{t-1}.

If $x = y$ (hence, $|A_{t-1}|_x = 2$) and $|A_{t-1}|_z = 2$, then $\mathcal{P}$ contains two singletons $\{x\}$ and $\{z\}$. Let $\mathcal{P}' := (\mathcal{P} \setminus \{\{x\}, \{z\}\}) \cup \{\{x, z\}\}$. Then, $\mathcal{P}'$ is indeed a partial solution at t (since $\{x, z\}$ is an edge with $\mathrm{end}(z) > t$) that satisfies A_t, since $|A_t|_x = 0$ and $|A_t|_z = 1$.

If $x = y$ (hence, $|A_{t-1}|_x = 2$) and $|A_{t-1}|_z = 1$, then $\mathcal{P}$ contains a singleton $\{x\}$ and an edge $\{z, u\}$. Also, note that $|A_{t-1}|_u = 0$, that is, $u \notin A_{t-1}$. Because there is an edge $\{x, z\}$, the triple $\{x, z, u\}$ contains a P_3. Let $\mathcal{P}' := (\mathcal{P} \setminus \{\{x\}, \{z, u\}\}) \cup \{\{x, z, u\}\}$. Then $\mathcal{P}'$ is a partial solution at t that satisfies A_t, since $|A_t|_x = |A_t|_z = |A_t|_u = 0$.

If $z = y$ (hence, $|A_{t-1}|_x = 1$ and $|A_{t-1}|_z = 2$), then similarly $\mathcal{P}$ contains an edge $\{x, u\}$ and a singleton $\{z\}$: $\mathcal{P}' := (\mathcal{P} \setminus \{\{x, u\}, \{z\}\}) \cup \{\{x, z, u\}\}$ is a partial solution at t that satisfies A_t.

If $y \neq x$ and $y \neq z$ (hence, $|A_{t-1}|_x = 1$ and $|A_{t-1}|_y = 1$), and $|A_{t-1}|_z = 2$, then $\mathcal{P}$ contains two edges $\{x, u\}$ and $\{y, v\}$ and a singleton $\{z\}$. Recall that $v, u \notin A_{t-1}$. Assume first that $\mathrm{start}(y) < \mathrm{start}(x)$, then interval u intersects

y, and $\{y, u, v\}$ contains a P_3. Also, $\{x, z\}$ forms an edge with $|A_t|_z = 1$: define $\mathcal{P}' \coloneqq (\mathcal{P} \setminus \{\{x, u\}, \{y, v\}, \{z\}\}) \cup \{\{y, u, v\}, \{x, z\}\}$. In the case where $\text{start}(x) < \text{start}(y)$, $\{x, u, v\}$ contains a P_3 and $\mathcal{P}' \coloneqq (\mathcal{P} \setminus \{\{x, u\}, \{y, v\}, \{z\}\}) \cup \{\{x, u, v\}, \{y, z\}\}$ is a partial solution at t that satisfies A_t.

Finally, we have a similar situation when $y \neq x$, $y \neq z$ and $|A_{t-1}|_z = 1$: then, $\mathcal{P}$ contains three edges $\{x, u\}$, $\{y, v\}$ and $\{z, w\}$. If $\text{start}(y) < \text{start}(x)$, then both $\{y, u, v\}$ and $\{x, z, w\}$ contain P_3s. Otherwise, $\{x, u, v\}$ and $\{y, z, w\}$ contain P_3s. Thus, we define $\mathcal{P}' \coloneqq (\mathcal{P} \setminus \{\{x, u\}, \{y, v\}, \{z, w\}\}) \cup \{\{y, u, v\}, \{x, z, w\}\}$ and $\mathcal{P}' \coloneqq (\mathcal{P} \setminus \{\{x, u\}, \{y, v\}, \{z, w\}\}) \cup \{\{x, u, v\}, \{y, z, w\}\}$ respectively. In both cases, $\mathcal{P}'$ is a partial solution at t that satisfies A_t.

Overall, if Algorithm 1 returns `true`, then it defines A_{2n}. According to the property we have proven, there exists a partial solution at $t = 2n$, hence G has a P_3-partition. $\square$

The above lemmas allow us to conclude the correctness of Algorithm 1.

Theorem 3. *P_3-Partition on interval graphs is solvable in $O(n \log n + m)$ time.*

Proof. Let G be an interval graph. To prove the theorem, we show that Algorithm 1 returns `true` on G if and only if G has a P_3-partition. The "only if" part is the statement of Lemma 4. For the "if" part, suppose that G has a P_3-partition $\mathcal{P}$. Then Lemma 3 implies that Algorithm 1 defines list A_t at position $t = 2n$, which means it returns `true`.

It remains to prove the running time bound. We first preprocess the input as follows: in $O(n + m)$ time, we can get an interval representation of an interval graph with n intervals that use start and end points in $\{1, \ldots, n\}$ [10, Section 8]. We modify this representation so that each position is the start or end point of at most one interval: first, for each interval, we add its start point to the beginning of a list L and its end point to the end of L. We sort L using a stable sorting algorithm like counting sort in $O(n)$ time. The result is a sorted list L that, for each position, contains first the start points and then the end points. Now, in $O(n)$ time, we iterate over L and reassign each event points to its own position in $\{1, \ldots, 2n\}$ in the order of its appearance in L. At the same time, we build an $2n$-element array B such that $B[i]$ holds a pointer to the interval starting or ending at event point i (there is at most one such interval). It follows that all preprocessing works in $O(n + m)$ time.

After this preprocessing, each of the $O(n)$ iterations for some $t \in \{1, \ldots, 2n\}$ of the loop in line 2 of Algorithm 1 is executed in $O(\log n)$ time: in constant time, we get the interval $B[t]$ starting or ending at t and each operation on the token list can be executed in $O(\log n)$ time if it is implemented as a balanced binary tree (note that only the current value of A_t need to be kept at each point, hence it is never necessary for the algorithm to make a copy of the whole token list). $\square$

3 Cographs

A cograph is a graph that does not contain a P_4 (path on four vertices) as an induced subgraph. Cographs allow for a so-called *cotree* to be computed in linear time [9].

Definition 6. A cotree $\cot(G)$ of a cograph $G = (V, E)$ is a rooted binary tree $T = (V_T, E_T, r), r \in V_T$, where each internal node is assigned a label in $\{\oplus, \otimes\}$ and the set of leaves corresponds to the original set V of vertices such that:

- A subtree consisting of a single *leaf node* corresponds to an induced subgraph with a single vertex.
- A subtree rooted at a *union node*, labeled "$\oplus$", corresponds to the disjoint union of the subgraphs defined by the two children of that node.
- A subtree rooted at a *join node*, labeled "$\otimes$", corresponds to the *join* of the subgraphs defined by the two children of that node; that is, the union of the two subgraphs with additional edges between every two vertices corresponding to leaves in different subtrees.

Consequently, the subtree rooted at the root r of $\cot(G)$ corresponds to G.

Using a dynamic programming approach on the cotree representation of the cograph, we can solve Star Partition in polynomial time.

Theorem 4. Star Partition *can be solved in $O(kn^2)$ time on cographs.*

Proof. Let $(G = (V, E), s)$ be a Star Partition instance with G being a cograph. Let $T = (V_T, E_T, r) = \cot(G)$ denote the cotree of G. Furthermore, for any node $x \in V_T$, let $T[x]$ denote the subgraph of G that corresponds to the subtree of T rooted at x.

We define a dynamic programming table L as follows. For every node $x \in V_T$ and every non-negative integer $c \leq k$, the table entry $L[x, c]$ denotes the maximum number of leaves in $T[x]$ that are covered by a center in $T[x]$ when c vertices in $T[x]$ are centers. Consequently, (G, s) is a yes-instance if and only if $L[r, k] = ks$. Now, let us describe how to compute L processing the cotree T bottom up.

Leaf nodes. For a leaf node x, either the only vertex v from $T[x]$ is a center or not. In both cases no leaf in $T[x]$ is covered by v. Thus, $L[x, 0] = L[x, 1] = 0$ and $\forall c > 1 : L[x, c] = -\infty$.

Union nodes. Let x be a node labeled with "$\oplus$" and let x_1 and x_2 be its children. Note that there is no edge between a vertex from $T[x_1]$ and a vertex from $T[x_2]$, neither in $T[x]$ nor in any other subgraph of G corresponding to any $T[x']$, $x' \in V_T$. Thus, for every leaf v in $T[x]$ that is covered by a center v' from $T[x]$, it holds that either both v and v' are in $T[x_1]$ or both are in $T[x_2]$. Hence, it follows $L[x, c] = \max_{c_1 + c_2 = c}(L[x_1, c_1] + L[x_2, c_2])$.

Join nodes. Let x be a node labeled with "$\otimes$" and let x_1 and x_2 be its children. Join nodes are more complicated than leaf or union nodes for computing the table entries, because these nodes actually introduce the edges. However, they always introduce *all possible* edges between vertices from $T[x_1]$ and $T[x_2]$ which has some nice consequences. The idea is that the maximum number of leaves in $T[x]$ that are covered by centers in $T[x]$ is achieved by maximizing the number of leaves from $T[x_2]$ that are covered by centers from $T[x_1]$ and vice versa.

To compute $L[x, c]$, we introduce an auxiliary table A as follows. For every pair c_1, c_2 of non-negative integers with $c_1 + c_2 = c$, the table entry $A[c_1, c_2]$ denotes the maximum number of leaves in $T[x]$ that are covered by a center in $T[x]$ when c_1 vertices in $T[x_1]$ are centers and c_2 vertices in $T[x_2]$ are centers. To this end, let ℓ_i, $i \in \{1, 2\}$, be the number of leaves in the desired s-star partition being in $T[x_i]$. (Note that in every solution every vertex is either a center or a leaf and a leaf is not necessarily already covered within the current $T[x]$. That is, ℓ_i can be larger than the number of leaves covered by a center in $T[x]$.) Moreover, $\ell_i = |V(T[x_i])| - c_i$, where $V(T[x_i])$ is the set of vertices in $T[x_i]$. To compute the auxiliary table A, we consider three cases:

Case 1: $(c_1 s > \ell_2) \wedge (c_2 s > \ell_1)$. In this case, we can cover all leaves in $T[x]$ by covering the leaves from $T[x_1]$ with centers from $T[x_2]$ and vice versa. Thus, $A[c_1, c_2] = \ell_1 + \ell_2$.

Case 2: $(c_1 s \leq \ell_2) \wedge (c_2 s \leq \ell_1)$. In this case, we can cover cs leaves in $T[x]$ by covering $c_1 s$ leaves from $T[x_2]$ by centers from $T[x_1]$ and $c_2 s$ leaves from $T[x_1]$ by centers from $T[x_2]$. This is obviously the best one can do. Thus, $A[c_1, c_2] = cs$.

Case 3: $(c_1 s > \ell_2) \wedge (c_2 s \leq \ell_1)$ or $(c_1 s \leq \ell_2) \wedge (c_2 s > \ell_1)$. In this case it is also optimal to greedily maximize the number of leaves from $T[x_2]$ that are covered by centers from $T[x_1]$ and vice versa. To see this, let $y_i, i \in \{1, 2\}$, denote the number of leaves from $T[x_i]$ that are covered by a center from $T[x_i]$. More precisely, assume that y_1 and y_2 are both greater than zero. Then, repeatedly take one center from $T[x_1]$ covering a leaf in $T[x_2]$ and one center from $T[x_2]$ covering a leaf in $T[x_1]$ and exchange their leaves until either y_1 or y_2 is zero (if both become zero, we would be in Case 2).

Without loss of generality, let $y_1 > 0$ and $y_2 = 0$. Note that this corresponds to the first subcase, i. e., $(c_1 s > \ell_2) \wedge (c_2 s \leq \ell_1)$—the other subcase works analogously. As $y_2 = 0$ and $c_2 s \leq \ell_1$, we can assume that c_2 centers from $T[x_2]$ cover altogether $c_2 s$ leaves from $T[x_1]$. Furthermore, all ℓ_2 leaves from $T[x_2]$ are covered by centers in $T[x_1]$. Since $c_1 s > \ell_2$, the centers in $T[x_1]$ might additionally cover some number ℓ' of leaves from $T[x_1]$. We thus have $A[c_1, c_2] = c_2 s + \ell_2 + \ell'$. We now compute the maximum possible value of ℓ'. Clearly:

- ℓ' is at most $c_1 s - \ell_2$, the maximum number of leaves that can be covered by c_1 centers after ℓ_2 leaves are covered in $T[x_2]$,

- ℓ' is at most $\ell_1 - c_2 s$, the maximum number of leaves that are not already covered by centers from $T[x_2]$, and
- ℓ' is at most $L[x_1, c_1]$, the maximum number of leaves from $T[x_1]$ that can be covered by centers from $T[x_1]$.

Hence, $\ell' \leq \min(c_1 s - \ell_2, \ell_1 - c_2 s, L[x_1, c_1])$

Conversely, for $\ell'' = \min(c_1 s - \ell_2, \ell_1 - c_2 s, L[x_1, c_1])$, it is possible for c_1 centers in $T[x_1]$ to cover ℓ'' leaves in $T[x_1]$ and ℓ_2 leaves in $T[x_2]$, and for c_2 centers in $T[x_2]$ to cover $c_2 s$ leaves in $T[x_1]$. Here, the property that a join node introduces all possible edges between the two subgraphs is crucial, because we can therefore simply cover leaves from $T[x_1]$ by centers from $T[x_1]$ in an optimal way. (*Each* center from $T[x_1]$ can cover *each* leaf from $T[x_2]$ and vice versa.) So $\ell' \geq \ell'' = \min(c_2 s - \ell_1, \ell_2 - c_1 s, L[x_2, c_2])$. Overall,

$$A[c_1, c_2] = \begin{cases} \ell_1 + \ell_2 & \text{if } (c_1 s > \ell_2) \wedge (c_2 s > \ell_1) \\ cs & \text{if } (c_1 s \leq \ell_2) \wedge (c_2 s \leq \ell_1) \\ c_2 s + \ell_2 + \min(c_1 s - \ell_2, \ell_1 - c_2 s, L[x_1, c_1]) & \text{if } (c_1 s > \ell_2) \wedge (c_2 s \leq \ell_1) \\ c_1 s + \ell_1 + \min(c_2 s - \ell_1, \ell_2 - c_1 s, L[x_2, c_2]) & \text{if } (c_1 s \leq \ell_2) \wedge (c_2 s > \ell_1). \end{cases}$$

Finally, we compute $L[x, c]$ by considering the auxiliary table, that is,

$$L[x, c] = \max_{c_1 + c_2 = c} (A[c_1, c_2]).$$

The $O(kn^2)$ running time of this algorithm can be seen as follows: Computing the cotree representation runs in linear time [9]. The table size of the dynamic program is bounded by $O(kn)$—there are $O(n)$ nodes in the cotree and $c \leq k$. Since $V(T[x_i])$ corresponds to the set of leaf nodes of the subtree of T rooted in x_i, the sizes $|V(T[x_i])|$ can be precomputed in linear time for each node x_i of the cotree. Hence, computing a table entry costs at most $O(n)$. □

4 Bipartite permutation graphs

In this section, we show that Star Partition can be solved in $O(n^2)$ time on bipartite permutation graphs. The class of bipartite permutation graphs is the intersection of the class of bipartite graphs and the class of permutation graphs. An alternative characterization of bipartite permutation graphs can be given using *strong orderings* of the vertices of a bipartite graph:

Definition 7 (Spinrad et al. [28]). A *strong ordering* $\prec$ of the vertices of a bipartite graph $G = (U, W, E)$ is the union of a total order $\prec_U$ of U and a total order $\prec_W$ of W, such that, for all edges $\{u, w\}$, $\{u', w'\}$ in E with $u, u' \in U$ and $w, w' \in W$, $u \prec u'$ and $w' \prec w$ implies that there are edges $\{u, w'\}$ and $\{u', w\}$ in E.

A graph is a bipartite permutation graph if and only if it is bipartite and there is a strong ordering of its vertices; a strong ordering can be computed in linear time [28].

In a bipartite graph G with vertex set $U \cup W$, if the subgraph induced by a size-$(s+1)$ vertex subset $X \subseteq U \cup W$ contains an s-star, then this induced subgraph *is* a star—there is only one way to choose the star center. Thus, we refer to $G[X]$ as a star. We denote by $\operatorname{center}(X)$ the center of the star $G[X]$. Observe that the number k_U of star centers in U and the number k_W of star centers in W are uniquely determined by the sizes $|U|$ and $|W|$ of the two independent vertex sets and by the number s of leaves in a star, since

$$|U| = k_U + s \cdot k_W \qquad \text{and} \qquad |W| = k_W + s \cdot k_U$$

and therefore

$$k_U = \frac{|U| - |W| \cdot s}{1 - s^2} \qquad \text{and} \qquad k_W = \frac{|W| - |U| \cdot s}{1 - s^2}.$$

If these numbers are not positive integers, then G does not have an s-star partition. Thus, we assume throughout this section that k_U and k_W are positive integers.

Our key to obtain star partitions on bipartite permutation graphs is a structural result that only a certain "normal form" of star partitions has to be searched for. This paves the way to developing a dynamic programming algorithm exploiting these normal forms. We define these structural properties of an s-star partition of bipartite permutation graphs in the following.

Let (G, s) be a Star Partition instance, where $G = (U, W, E)$ is a bipartite permutation graph, $\prec$ is a strong ordering of the vertices, and $\preccurlyeq$ is the reflexive closure of $\prec$. For two vertex sets A, B, we also write $A \prec B$ if for all vertices $v \in A$ and $w \in B$, we have $v \prec w$.

Assume that G admits an s-star partition $\mathcal{P}$. Let $X \in \mathcal{P}$ form a star. By $\operatorname{lm}(X)$ (respectively by $\operatorname{rm}(X)$), we denote the leftmost (that is, the minimum), respectively the rightmost (that is, the maximum) leaf of X with respect to $\prec$. The *scope* of star X is the set $\operatorname{scope}(X) \coloneqq \{v \mid x_l \preccurlyeq v \preccurlyeq x_r\}$ containing all vertices from $x_l = \operatorname{lm}(X)$ to $x_r = \operatorname{rm}(X)$. The *width* of star X is the cardinality of its scope, that is, $\operatorname{width}(X) \coloneqq |\operatorname{scope}(X)|$. The *width* of $\mathcal{P}$, $\operatorname{width}(\mathcal{P})$, is the sum of $\operatorname{width}(X)$ over all $X \in \mathcal{P}$.

Let $e = \{u, w\}$ and $e' = \{u', w'\}$ be two edges. We say that e and e' *cross* each other if it holds that $u \prec u'$ and $w' \prec w$ or if it holds that $u' \prec u$ and $w \prec w'$. The *edge-crossing number* of two stars $X, Y \in \mathcal{P}$ is the number of pairs of crossing edges e, e' with respect to the given strong order $\prec$ where e is an edge of X and e' is an edge of Y. The edge-crossing number $\#\text{edge-crossings}(\mathcal{P})$ of $\mathcal{P}$ is the sum of the edge-crossing numbers over all pairs of stars $X \neq Y \in \mathcal{P}$.

We identify the possible configurations of two stars, depending on the relative positions of their leaves and centers, see Figure 5. Among those, the following two configurations are favorable: Given $X, Y \in \mathcal{P}$, we say that X and Y are

- *non-crossing* if their edge-crossing number is zero;

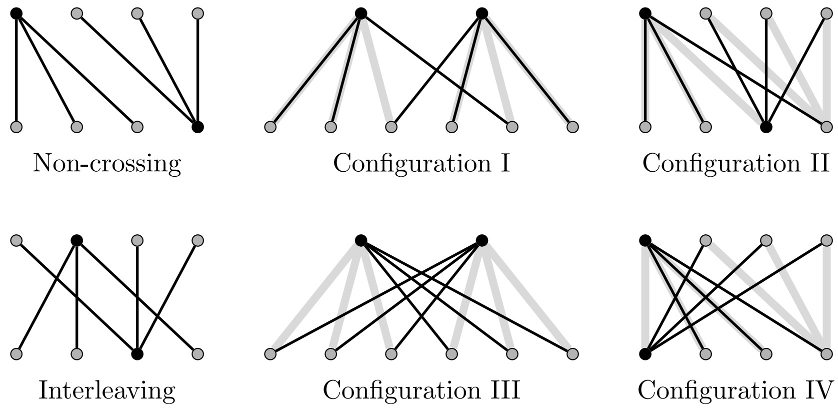


Figure 5: Possible interactions between two stars of a partition. Centers are drew black. The four possible configurations of star centers and scopes that are neither non-crossing nor interleaving are labeled I to IV. By Lemma 5, any partition containing one of the configurations I to IV can be edited to reduce the score (see the thick gray edges).

- *interleaving* if $\text{center}(X) \in \text{scope}(Y)$ and $\text{center}(Y) \in \text{scope}(X)$;

We say that $\mathcal{P}$ is *good* if any two stars $X \neq Y \in \mathcal{P}$ are either non-crossing or interleaving. We define the *score* of $\mathcal{P}$ as the tuple $(\text{width}(\mathcal{P}), \#\text{edge-crossings}(\mathcal{P}))$. We use the lexicographical order to compare scores.

These definitions allow us to observe the following property and show a normal form of star partitions in bipartite permutation graphs.

Property 4. Let $u_0 \prec u_1$ and $w_0 \prec w_1$ be four vertices such that edges $\{u_0, w_1\}$ and $\{u_1, w_0\}$ are in G. Then, G has edges $\{u_0, w_0\}$ and $\{u_1, w_1\}$ and, for any edge e crossing one (respectively both) edge(s) in $\{\{u_0, w_0\}, \{u_1, w_1\}\}$, e crosses one (respectively both) edge(s) in $\{\{u_0, w_1\}, \{u_1, w_0\}\}$.

Proof. The existence of the edges $\{u_0, w_0\}$ and $\{u_1, w_1\}$ is a direct consequence of Definition 7. Let $e = \{u, w\}$ be an edge crossing $\{u_0, w_0\}$ and/or $\{u_1, w_1\}$. We consider the cases where $u \prec u_0$ and where $u_0 \prec u \prec u_1$ (the case $u_1 \prec u$ being symmetrical to $u \prec u_0$).

If $u \prec u_0$, then $w_0 \prec w$, and e crosses both $\{u_0, w_0\}$ and $\{u_1, w_0\}$. Also, if e crosses $\{u_1, w_1\}$, then e also crosses $\{u_0, w_1\}$, which proves the property for this case.

If $u_0 \prec u \prec u_1$, then if e crosses $\{u_0, w_0\}$, then e also crosses $\{u_0, w_1\}$. If e crosses $\{u_1, w_1\}$, then e also crosses $\{u_1, w_0\}$. Overall, the property is thus proven for all cases. $\square$

Our main structural lemma now is the following.

Lemma 5. *Any s-star partition of a bipartite permutation graph G with minimum score is a good s-star partition, that is, any two stars are either non-crossing or interleaving.*

Proof. Let $\mathcal{P}$ be an s-star partition for G. First, we show that any two stars $X \neq Y \in \mathcal{P}$ are non-crossing, interleaving, or in one of the following four configurations (possibly after exchanging the roles of X and Y, see Figure 5 for an illustration):

Configuration I. $\mathrm{scope}(X) \cap \mathrm{scope}(Y) \neq \emptyset$;

Configuration II. $\mathrm{center}(Y) \in \mathrm{scope}(X)$ and $\mathrm{center}(X) \notin \mathrm{scope}(Y)$;

Configuration III. $\mathrm{center}(X) \prec \mathrm{center}(Y)$ and $\mathrm{scope}(Y) \prec \mathrm{scope}(X)$;

Configuration IV. $\mathrm{center}(X) \prec \mathrm{scope}(Y)$ and $\mathrm{center}(Y) \prec \mathrm{scope}(X)$ or, symmetrically, $\mathrm{scope}(Y) \prec \mathrm{center}(X)$ and $\mathrm{scope}(X) \prec \mathrm{center}(Y)$.

First, assume that $\mathrm{center}(X)$ and $\mathrm{center}(Y)$ are both either in U or in W. Furthermore, assume, without loss of generality, that $\mathrm{center}(X) \prec \mathrm{center}(Y)$. If X and Y are not in Configuration I, then either $\mathrm{scope}(X) \prec \mathrm{scope}(Y)$ or $\mathrm{scope}(Y) \prec \mathrm{scope}(X)$. If $\mathrm{scope}(X) \prec \mathrm{scope}(Y)$, then $G[X]$ and $G[Y]$ are non-crossing. Otherwise, $\mathrm{scope}(Y) \prec \mathrm{scope}(X)$ and, hence, Configuration III holds.

If $\mathrm{center}(X)$ and $\mathrm{center}(Y)$ are in different vertex sets and if X and Y are not in Configuration IV, then $\mathrm{center}(X) \in \mathrm{scope}(Y)$ and/or $\mathrm{center}(Y) \in \mathrm{scope}(X)$. If $\mathrm{center}(X) \in \mathrm{scope}(Y)$ and $G[X]$ and $G[Y]$ are not interleaving, then $\mathrm{center}(Y) \notin \mathrm{scope}(X)$ and we are in Configuration II. Otherwise, if $\mathrm{center}(Y) \in \mathrm{scope}(X)$ and, again, $G[X]$ and $G[Y]$ are not interleaving, then $\mathrm{center}(X) \notin \mathrm{scope}(Y)$ and we are again in Configuration II.

We now prove that a minimum-score s-star partition $\mathcal{P}$ does not contain any pair of stars $X \neq Y \in \mathcal{P}$ in Configurations I, II, III or IV (see Figure 5). For each such configuration, we construct an s-star partition $\mathcal{P}'$ with a score strictly smaller than $\mathcal{P}$.

Configuration I. Let X, Y be two stars of $\mathcal{P}$ in Configuration I, that is, $\mathrm{scope}(X) \cap \mathrm{scope}(Y) \neq \emptyset$. Write $x_c = \mathrm{center}(X)$ and $y_c = \mathrm{center}(Y)$. Then, x_c and y_c are either both in U or both in W. Without loss of generality, assume $x_c \prec y_c$. Write $\{z_1, z_2, \ldots, z_{2s}\}$ for the union of the leaves of X and Y, with indices taken such that $z_i \prec z_j$ for $1 \leq i < j \leq 2s$. Let $Z_l = \{z_1, \ldots, z_s\}$ and $Z_r = \{z_{s+1}, \ldots, z_{2s}\}$. We first show that both vertex sets $Z_l \cup \{x_c\}$ and $Z_r \cup \{y_c\}$ form a star in G.

Let k be the index such that $z_k = \mathrm{lm}(Y)$. Then, since the scopes of X and Y intersect, z_k cannot be to the right of all the leaves of $G[X]$, hence we have $k \leq s$, and $z_k \prec Z_r$. Consider now any $z \in Z_r$. If $z \in Y$, then there exists an edge $\{z, y_c\}$ in G. If $z \in X$, then there exists an edge $\{z, x_c\}$ in G that crosses $\{z_k, y_c\}$ (since $z_k \prec z$ and $x_c \prec y_c$). Thus, there also exists an edge $\{z, y_c\}$ in G by Definition 7. With a symmetrical argument, G has an edge $\{z, x_c\}$ for all $z \in Z_l$.

It follows that the vertex sets $X' = Z_l \cup \{x_c\}$ and $Y' = Z_r \cup \{y_c\}$ both form stars in G.

We now compare the widths of $G[X']$ and $G[Y']$ to the widths of the original stars $G[X]$ and $G[Y]$. Let w be the total number of elements between z_1 and z_{2s}, that is, the cardinality of the vertex set $\{u \mid z_1 \preccurlyeq u \preccurlyeq z_{2s}\} = \text{scope}(X) \cup \text{scope}(Y)$. Then, using the fact that the scopes of X' and Y' are disjoint and included in a size-w set, we have

$$\begin{aligned}\text{width}(X') + \text{width}(Y') &\leq w \\ &= |\,\text{scope}(X)| + |\,\text{scope}(Y)| - |\,\text{scope}(X) \cap \text{scope}(Y)| \\ &< \text{width}(X) + \text{width}(Y).\end{aligned}$$

We can thus construct an s-star partition $\mathcal{P}' = (\mathcal{P} \setminus \{X, Y\}) \cup \{X', Y'\}$ such that $\text{width}(\mathcal{P}') < \text{width}(\mathcal{P})$, that is, with strictly smaller score. Thus, no pair of stars in the minimum-score s-star partition $\mathcal{P}$ may be in Configuration I.

Configuration II. Let X, Y be two stars of $\mathcal{P}$ in Configuration II, i. e., $\text{center}(Y) \in \text{scope}(X)$ and $\text{center}(X) \notin \text{scope}(Y)$. Write $x_c = \text{center}(X)$ and $y_c = \text{center}(Y)$. Then $y_c \prec \text{rm}(X)$ and either $x_c \prec \text{scope}(Y)$ or $\text{scope}(Y) \prec x_c$. We only consider the case $x_c \prec \text{scope}(Y)$; the case $\text{scope}(Y) \prec x_c$ works analogously.

Let $v = \text{rm}(X)$ be the rightmost vertex of the leaves of the star $G[X]$. First, G contains the edge $\{x_c, y_c\}$ since the star $G[Y]$ has at least one leaf u with $x_c \prec u$ and $y_c \prec v$, and G contains the edges $\{x_c, v\}$ and $\{y_c, u\}$. Now, consider any vertex $u \in Y \setminus \{\text{center}(Y)\}$. Then, the edge $\{x_c, v\}$ crosses the edge $\{u, y_c\}$, since $x_c \prec u$ and $y_c \prec \text{rm}(X)$. The graph G contains the edges $\{x_c, y_c\}$ and $\{v, u\}$. Thus, the vertex sets $X' = (X \setminus \{v\}) \cup \{y_c\}$ and $Y' = (Y \setminus \{y_c\}) \cup \{v\}$ both form stars in G.

We now compare the widths of $G[X']$ and $G[Y']$ to the widths of the original stars $G[X]$ and $G[Y]$.

Since $y_c \prec v$, one has $\text{width}(X') \leq \text{width}(X) - 1$. Obviously, $\text{width}(Y) = \text{width}(Y')$. We can thus construct an s-star partition $\mathcal{P}' = (\mathcal{P} \setminus \{X, Y\}) \cup \{X', Y'\}$ with $\text{width}(\mathcal{P}') < \text{width}(\mathcal{P})$, that is, with strictly smaller score. Therefore, no pair of stars in the s-star partition $\mathcal{P}$ may be in Configuration II.

Configuration III. Let X, Y be two stars of $\mathcal{P}$ in Configuration III. Let $x_c \coloneqq \text{center}(X)$ and $y_c \coloneqq \text{center}(Y)$ and assume, without loss of generality, that $x_c \prec y_c$. Then, $\text{scope}(Y) \prec \text{scope}(X)$. Thus, all edges of $G[X]$ cross all edges of $G[Y]$. Hence, there exists an edge $\{x_c, y\}$ for each leaf y of $G[Y]$, and an edge $\{y_c, x\}$ for each leaf x of $G[X]$. Defining $X' = (X \setminus \{x_c\}) \cup \{y_c\}$ and $Y' = (Y \setminus \{x_c\}) \cup \{y_c\}$, we thus have two stars $G[X']$ and $G[Y']$ with the same width as $G[X]$ and $G[Y]$, respectively. Hence, the s-star partition $\mathcal{P}' = (\mathcal{P} \setminus \{X, Y\}) \cup \{X', Y'\}$ has the same width as $\mathcal{P}$.

We now show that $\#\text{edge-crossings}(\mathcal{P}') < \#\text{edge-crossings}(\mathcal{P})$. We write B_X (respectively B_Y, $B_{X'}$, and $B_{Y'}$) for the *branches* of the corresponding star, that

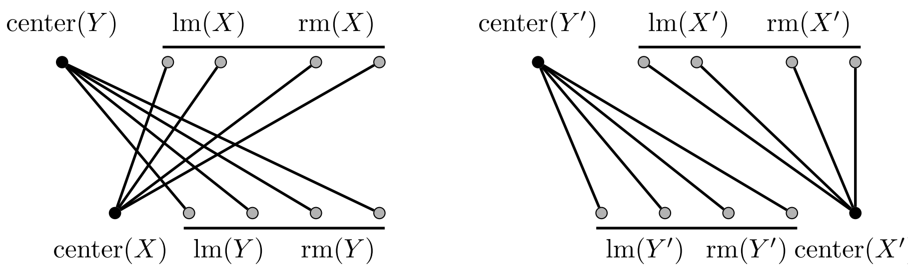


Figure 6: Left: Two stars X and Y in Configuration IV such that $\mathrm{d}(X,Y)$ is minimal. Right: Two stars X' and Y' obtained from X and Y, with equal width and fewer crossings.

is, for the set of edges of $G[X]$ (respectively of $G[Y]$, $G[X']$, and $G[Y']$), and $R_{X,Y}$ (respectively $R_{X',Y'}$) for the set of edges in $G[Z]$ for any $Z \in \mathcal{P} \setminus \{X,Y\}$ (respectively for any $Z \in \mathcal{P}' \setminus \{X',Y'\}$). Note that, by definition of $\mathcal{P}'$, $R_{X',Y'} = R_{X,Y}$. We thus simply denote this set by R. We write $\times_{b,b}$ (respectively $\times_{b',b'}$) for the number of crossings between branches of B_X and B_Y (respectively of $B_{X'}$ and $B_{Y'}$), $\times_{b,r}$ (respectively $\times_{b',r}$) for the number of crossings between a branch in $B_X \cup B_Y$ (respectively in $B_{X'} \cup B_{Y'}$) and an edge in R, and $\times_{r,r}$ for the number of crossings between two edges in R. Note that #edge-crossings($\mathcal{P}$) $= \times_{b,b} + \times_{b,r} + \times_{r,r}$ and that #edge-crossings($\mathcal{P}'$) $= \times_{b',b'} + \times_{b',r} + \times_{r,r}$.

It is easy to see that $\times_{b',b'} = 0$ (X' and Y' form non-crossing stars), and $\times_{b,b} > 0$. Let x_i (respectively y_i) be the i-th leaf of X (respectively of Y) in the order $\prec$. Then, by Property 4, any edge in R crossing one or two edges among $\{\{y_c, x_i\}, \{x_c, y_i\}\}$ also crosses at least as many edges among $\{\{x_c, x_i\}, \{y_c, y_i\}\}$. Summing over all branches and all crossing edges, we obtain $\times_{b',r} \leq \times_{b,r}$. Thus, overall, we indeed have #edge-crossings($\mathcal{P}'$) $<$ #edge-crossings($\mathcal{P}$).

Finally, we have constructed an s-star partition with the same width but fewer crossings, that is, with strictly smaller score. Thus, no pair of stars in the s-star partition $\mathcal{P}$ may be in Configuration III.

Configuration IV. Let X, Y be two stars of $\mathcal{P}$ in Configuration IV. Without loss of generality, we assume that center(X) $\prec$ scope(Y) and center(Y) $\prec$ scope(X). We moreover assume that X and Y are chosen so that the number of elements between center(X) and rm(Y), written $\mathrm{d}(X,Y)$, is minimal among all pairs in Configuration IV. The configuration is depicted in more detail in Figure 6 (left).

We first show that $\mathrm{d}(X,Y) = s-1$, which means that no vertex exists between center(X) and rm(Y), except for the $s-1$ other leaves of Y. Suppose, towards a contradiction, that there is a vertex $z \notin \{\text{center}(X)\} \cup \text{scope}(Y)$ such that center(X) $\prec z \prec$ rm(Y).

Assume first that z is the center of a star $G[X']$ with $X' \in \mathcal{P}$. Then, $\mathrm{scope}(X) \prec \mathrm{scope}(X')$, since X and X' cannot be in Configuration I or III. Moreover, we have $\mathrm{center}(X) \prec z \prec \mathrm{scope}(Y)$ since, otherwise, $z \in \mathrm{scope}(Y)$ and Y and X' would be in Configuration II. Hence, X' and Y are in configuration IV (with $\mathrm{center}(X') \prec \mathrm{scope}(Y)$, $\mathrm{center}(Y) \prec \mathrm{scope}(X')$), and $\mathrm{d}(X', Y) < \mathrm{d}(X, Y)$, which is a contradiction.

Now assume that z is a leaf of a star $G[Y']$ with $Y' \in \mathcal{P}$. First compare Y and Y': $\mathrm{scope}(Y') \cap \mathrm{scope}(Y) = \emptyset$ since, otherwise, Y and Y' would be in Configuration I. Using $z \prec \mathrm{rm}(Y)$, it follows that $\mathrm{scope}(Y') \prec \mathrm{scope}(Y)$. This implies that $\mathrm{center}(Y') \prec \mathrm{center}(Y) \prec \mathrm{scope}(X)$ since, otherwise, Y' and Y would be in Configuration III. We now compare X and Y'. We have already seen that $\mathrm{center}(Y') \prec \mathrm{scope}(X)$. Also, $\mathrm{center}(X) \notin \mathrm{scope}(Y')$ since, otherwise, Y' and X would be in Configuration II. Using $\mathrm{center}(X) \prec z$, we thus have $\mathrm{center}(X) \prec \mathrm{scope}(Y')$, which implies that X and Y' are in Configuration IV with $\mathrm{d}(X, Y') < \mathrm{d}(X, Y)$, which is a contradiction. We conclude that no vertex other than the leaves of Y may exist between $\mathrm{center}(X)$ and $\mathrm{rm}(Y)$.

We now construct an s-star partition with score strictly less than $\mathcal{P}$. To this end, let $X_0 = X \setminus \{\mathrm{center}(X)\}$ and $Y_0 = Y \setminus \{\mathrm{rm}(Y)\}$. First observe that G contains the edge $\{\mathrm{center}(X), \mathrm{center}(Y)\}$ since there is an edge in $G[X]$ and an edge in $G[Y]$ crossing each other. Hence, $Y' = Y_0 \cup \{\mathrm{center}(X)\}$ forms a star. Now, consider any vertex $u \in X_0$. The edge $\{\mathrm{center}(X), u\}$ crosses the edge $\{\mathrm{center}(Y), \mathrm{rm}(Y)\}$ and, therefore, G contains the edge $\{\mathrm{rm}(Y), u\}$. Thus, $X' = X_0 \cup \{\mathrm{rm}(Y)\}$ forms a star. For an illustration, see Figure 6 (right). Also, X' and Y' are non-crossing (X' is completely to the right of Y').

We now compare the widths of $G[X']$ and $G[Y']$ to the widths of the original stars $G[X]$ and $G[Y]$. Obviously, $\mathrm{width}(X') = \mathrm{width}(X)$. Moreover, since $\mathrm{d}(X, Y) = s - 1$, it follows that $\mathrm{width}(Y') = \mathrm{width}(Y) = s$. Hence, the s-star partition $\mathcal{P}' = (\mathcal{P} \setminus \{X, Y\}) \cup \{X', Y'\}$ has the same width as $\mathcal{P}$.

Since the widths have not changed, we have to show that $\#\text{edge-crossings}(\mathcal{P}') < \#\text{edge-crossings}(\mathcal{P})$. We introduce the same notations as in Configuration III: Let B_X (respectively B_Y, $B_{X'}$, and $B_{Y'}$) be the set of branches of the corresponding star, that is, the set of edges in $G[X]$ (respectively in $G[Y]$, $G[X']$, and $G[Y']$), and let $R_{X,Y}$ (respectively $R_{X',Y'}$) be the set of edges in $G[Z]$ for any $Z \in \mathcal{P} \setminus \{X, Y\}$ (respectively for any $Z \in \mathcal{P}' \setminus \{X', Y'\}$). Note that by definition of $\mathcal{P}'$, $R_{X',Y'} = R_{X,Y}$, and we thus simply denote this set by R. Furthermore, let $\times_{b,b}$ (respectively $\times_{b',b'}$) be the number of crossings between branches of B_X and B_Y (respectively between branches of $B_{X'}$ and $B_{Y'}$), let $\times_{b,r}$ (respectively $\times_{b',r}$) be the number of crossings between a branch in $B_X \cup B_Y$ (respectively in $B_{X'} \cup B_{Y'}$) and an edge in R, and let $\times_{r,r}$ be the number of crossings between two edges in R. Note that $\#\text{edge-crossings}(\mathcal{P}) = \times_{b,b} + \times_{b,r} + \times_{r,r}$ and that $\#\text{edge-crossings}(\mathcal{P}') = \times_{b',b'} + \times_{b',r} + \times_{r,r}$. Then, it is easy to see that $\times_{b',b'} = 0$ (X' and Y' form non-crossing stars), and $\times_{b,b} > 0$.

We now show that $\times_{b',r} \leq \times_{b,r}$. First recall that no edge in R has an end point between $\mathrm{center}(X)$ and $\mathrm{rm}(Y)$. We consider the branches in $B_{X'} \cup B_{Y'}$ and, for each, give a unique edge in $B_X \cup B_Y$ crossing the same edges of R. For any leaf x of X', any $r \in R$ crossing $\{\mathrm{center}(X'), x\}$ must also cross $\{\mathrm{center}(X), x\}$.

For the leftmost branch of Y', any $r \in R$ crossing $\{\text{center}(X), \text{center}(Y)\}$ must also cross $\{\text{rm}(Y), \text{center}(Y)\}$. For any other branch $b = \{\text{center}(Y), y\}$ of Y', any $r \in R$ must also cross the same branch b of Y'. Overall, we indeed have $\times_{b',r} \leq \times_{b,r}$, which implies $\#\text{edge-crossings}(\mathcal{P}') < \#\text{edge-crossings}(\mathcal{P})$.

Altogether, we have shown that a minimum-score s-star partition $\mathcal{P}$ containing pairs of stars in Configurations I to IV leads to a contradiction, since, in this case, we could find an s-star partition of lower score, which is a contradiction. □

As a consequence of Lemma 5, we obtain the following corollary.

Corollary 1. *Let $\mathcal{P}$ be an s-star partition of a bipartite permutation graph G with minimum score. Then, for each star $X \in \mathcal{P}$, there is at most one $Y \in \mathcal{P}$ such that X and Y are interleaving, and for all $Z \in \mathcal{P} \setminus \{X, Y\}$, X and Z are non-crossing.*

Proof. Since $\mathcal{P}$ has minimum score, for any $Y \in \mathcal{P} \setminus \{X\}$, $G[X]$ and $G[Y]$ are either interleaving or non-crossing.

Any star interleaving with $G[X]$ contains center(x) in its scope. If there exist at least two such stars in $\mathcal{P}$, then their scopes intersect and they are in Configuration I, which is impossible by Lemma 5. □

We now informally describe a dynamic programming algorithm for deciding whether a bipartite graph $G = (U, W, E)$ allows for a *good* s-star partition. It builds up a solution following the strong ordering of the graph from left to right. A partial solution can be extended in three ways only: either (i) a star is added with the center in U, or (ii) a star is added with the center in W, or (iii) two interleaving stars are added. The algorithm can thus compute, for any given number of centers in U and in W, whether it is possible to partition the leftmost vertices of U and W in one of the three ways (i)–(iii). This algorithm leads to the following result.

Theorem 5. Star Partition *can be solved in $O(n^2)$ time on bipartite permutation graphs.*

Proof. Let (G, s) denote a Star Partition instance, where $G = (U, W, E)$ is a bipartite permutation graph. Furthermore, let $U = \{u_1, u_2, \ldots, u_{k_U}\}$ and $W = \{w_1, w_2, \ldots, w_{k_W}\}$ such that $u_i \prec u_j$ (respectively $w_i \prec w_j$) implies $i < j$ for some fixed strong ordering $\prec$. We describe a dynamic programming algorithm that finds a good s-star partition $\mathcal{P}$. The idea is to use the fact that a star from $\mathcal{P}$ is either interleaving with exactly one other star from $\mathcal{P}$ or it does not cross any other star from $\mathcal{P}$ (see Lemma 5 and Corollary 1). In both cases, the part of the graph that lies entirely to the left of the star (of the two interleaving stars respectively) with respect to the strong ordering must have an s-star partition on its own. This is clearly also true for the part of the graph that lies entirely to the right, but we do not need this for the proof.

Informally, an entry $T(x, y)$ of our binary dynamic programming table T is true if and only if x stars with centers from U and y stars with centers from W can "consecutively cover" the correspondingly large part of the graph from the left side of the strong ordering. Formally, the binary dynamic programming table T is defined as

$$T(x,y) = \begin{cases} 1 & \text{if } G[\{u_1, u_2, \ldots, u_{x+s\cdot y}, w_1, w_2, \ldots, w_{y+s\cdot x}\}] \\ & \quad \text{has an } s\text{-star partition,} \\ 0 & \text{otherwise.} \end{cases}$$

Initialize the table T by:

$$T(0,1) = \begin{cases} 1 & \text{if } G[\{u_1, u_2, \ldots, u_s, w_1\}] \text{ contains an } s\text{-star,} \\ 0 & \text{otherwise,} \end{cases}$$

$$T(1,0) = \begin{cases} 1 & \text{if } G[\{u_1, w_1, w_2, \ldots, w_s\}] \text{ contains an } s\text{-star,} \\ 0 & \text{otherwise, and} \end{cases}$$

$$T(1,1) = \begin{cases} 1 & \text{if } G[\{u_1, u_2, \ldots, u_{s+1}, w_1, w_2, \ldots, w_{s+1}\}] \\ & \quad \text{contains disjoint } s\text{-stars,} \\ 0 & \text{otherwise.} \end{cases}$$

Update the table T for all $1 < x \leq k_U$ and $1 < y \leq k_W$ by

$$T(x,y) =$$
$$\begin{cases} 1 & \text{if one of the following holds:} \\ & \text{(a) } T(x, y-1) = 1 \text{ and } G[\{u_{x+s\cdot(y-1)+1}, u_{x+s\cdot(y-1)+2}, \ldots, u_{x+s\cdot(y-1)+s}, \\ & \quad w_{(y-1)+s\cdot x+1}\}] \text{ contains an } s\text{-star,} \\ & \text{(b) } T(x-1, y) = 1 \text{ and } G[\{u_{(x-1)+s\cdot y+1}, w_{y+s\cdot(x-1)+1}, w_{y+s\cdot(x-1)+2}, \ldots, \\ & \quad w_{y+s\cdot(x-1)+s}\}] \text{ contains an } s\text{-star,} \\ & \text{(c) } T(x-1, y-1) = 1 \text{ and } G[\{u_{(x-1)+s\cdot(y-1)+1}, \ldots, u_{(x-1)+s\cdot(y-1)+s+1}, \\ & \quad w_{(y-1)+s\cdot(x-1)+1}, \ldots, w_{(y-1)+s\cdot(x-1)+s+1}\}] \text{ contains disjoint } s\text{-stars.} \\ 0 & \text{otherwise.} \end{cases}$$

Concerning the running time, first, a strong ordering of the vertices can be computed in linear time [28]. Second, the table in the dynamic program has $O(k^2)$ entries and initialization as well as updating works in $O(s^2)$ time. Hence, the total running time is $O(k^2 \cdot s^2) = O(n^2)$.

Concerning the correctness of the algorithm, we show that $T(k_U, k_V)$ is true if and only if there is a good s-star partition and, hence, if and only if there is an s-star partition. To this end, consider an s-star partition $\mathcal{P}'$ for $G' := G[\{u_1, u_2, \ldots, u_{x+s\cdot y},\ w_1, w_2, \ldots, w_{y+s\cdot x}\}]$ with minimum score. Now there are three simple cases:

Case (a). The rightmost vertex of G' in W is a center of a non-crossing star in $\mathcal{P}'$ and, hence, $G[\{u_1, u_2, \ldots, u_{x+s\cdot(y-1)}, w_1, w_2, \ldots, w_{y-1+s\cdot x}\}]$ has an s-star partition.

Case (b). The rightmost vertex of G' in U is a center of a non-crossing star in $\mathcal{P}'$ and, hence, $G[\{u_1, u_2, \dots, u_{x-1+s\cdot y}, w_1, w_2, \dots, w_{y+s\cdot(x-1)}\}]$ has an s-star partition.

Case (c). The rightmost vertex of G' in U and G's rightmost vertex in W are leaves of two interleaving stars. Due to Corollary 1, none of the other stars from $\mathcal{P}'$ is crossing these two stars. It follows that $G[\{u_1, u_2, \dots, u_{x-1+s\cdot(y-1)}, w_1, w_2, \dots, w_{y-1+s\cdot(x-1}\}]$ has an s-star partition.

Note that the rightmost vertex of G' in U can only be a leaf of a non-crossing star in $\mathcal{P}'$ if the rightmost vertex of G' in W is the center and vice versa. Otherwise, the corresponding star is clearly not non-crossing. Hence, these cases are already covered by (a) and (b). Furthermore, neither the rightmost vertex of G' in U nor in W can be a center of an interleaving star from $\mathcal{P}'$, because both are rightmost with respect to the strong ordering and, thus, interleaving is impossible. Thus we considered all cases and the update process is correct. $\square$

5 Split graphs

A *split graph* is a graph whose vertices can be partitioned into a clique (that is, a complete subgraph) and an independent set (that is, a subgraph with only isolated vertices). Remarkably, split graphs are the only graph class where we could show that P_3-Partition is solvable in polynomial time, but that Star Partition for $s \geq 3$ is NP-hard.

More precisely, we solve P_3-Partition on split graphs by reducing it to finding a restricted form of *factor* in an auxiliary graph; herein, a *factor* of a graph G is a spanning subgraph of G (that is, a subgraph containing all vertices). This graph factor problem then can be solved in polynomial time [11]. Alternatively, we can also solve the problem by reducing it to finding perfect matchings (Theorem 6).

Let $G = (C \cup I, E_C \cup E)$ be a split graph where (C, E_C) is a clique, I induces an independent set, and $B = (C \cup I, E)$ forms a bipartite graph over C and I. Note that if $|C| + |I|$ is not a multiple of 3, or if $|I| > 2|C|$, then G trivially has no P_3-partition. We thus assume that $|C| + |I|$ (and hence $2|C| - |I|$) is a multiple of 3, and that $2|C| - |I| \geq 0$.

First, we show how a P_3-partition of a split graph is related to a specific factor of the bipartite graph B. Assume that G admits a partition into P_3s and let P denote the set of edges in the partition. There are three types of P_3s:

(i) a P_3 consisting of three clique vertices,

(ii) a P_3 consisting of two clique vertices and one independent set vertex, and

(iii) a P_3 consisting of one clique vertex and two independent set vertices.

Note that, for each P_3, we can assume that the edges are selected so that each independent set vertex is incident with *at most* one edge in P. In particular, this implies that the two clique vertices in Type (ii) are adjacent in the corresponding P_3. This leads to the following definition.

Definition 8. A factor F of the bipartite graph B is *feasible* if, in F, every independent set vertex has degree one, every clique vertex has degree zero, one or two, and there are at least as many degree-zero clique vertices as degree-one clique vertices.

It turns out that Definition 8 is necessary and sufficient for obtaining a P_3-partition.

Lemma 6. *A split graph G admits a partition into P_3s if and only if there exists a feasible factor of its bipartite graph B.*

Proof. Assume that there is a partition of G into P_3s with edge set P such that each vertex in I is incident with exactly one edge in P. Let $P_E := P \cap E$ be the subset of edges of the partition which connect vertices from C with vertices from I, and let $F := (C \cup I, P_E)$ be the corresponding factor of $B = (C \cup I, E)$.

Each independent set vertex in I has degree one in F (since it is adjacent to exactly one edge in P, which is also in P_E). Each clique vertex v in C belongs to a P_3 from P. Depending on the type of this P_3, in F, vertex v can have degree zero (Type (i) or (ii), note that we assume each independent set vertex to be incident with at most one edge in P), degree one (Type (ii)), or degree two (Type (iii)). Let $n_{(\mathrm{i})}$ (respectively $n_{(\mathrm{ii})}$) denote the number of P_3s of Type (i) (respectively (ii)). It remains to show that the number of degree-zero clique vertices is equal to or greater than the number of degree-one clique vertices: The number of degree-zero vertices is $n_{(\mathrm{ii})} + 3n_{(i)}$, and the number of degree-one vertices is $n_{(\mathrm{ii})}$, hence the difference is positive. Thus, F is feasible.

Conversely, let $F = (C \cup I, P_E)$ be a feasible factor of B. Then we partition G into P_3s as follows. For each degree-two clique vertex v in C, add $\{v, x, y\}$ to P where x and y are the neighbors of v in F. For each degree-one clique vertex v in C, add $\{v, x, y\}$ to P, where x is v's neighbor in F and y is an arbitrary degree-zero clique vertex (there are enough such vertices). The number of remaining degree-zero vertices in C is thus a multiple of 3: these vertices are simply grouped up in arbitrary triples Add these triples to P. Overall, due to the degree constraints, P is a P_3-partition of G. □

Cornuéjols [11] shows that finding a feasible factor in B can be solved in polynomial time by reducing it to finding disjoint edges and triangles in a corresponding auxiliary graph. Nevertheless, we show in the following how to reduce the problem to finding perfect matchings. To this end, we formulate a nice property that a feasible factor in B must fulfill:

Property 5. Let $F = (C \cup I, P \cap E)$ be a feasible factor of the bipartite graph $B = (C \cup I, E)$. Let q and r be two non-negative integers such that $r \in \{0, 1\}$ and $(2|C| - |I|)/3 = 2q + r$. Then, in F, the number n_1 of degree-one vertices in C is $2i + r$ for some i, $0 \leq i \leq q$. In particular, $n_1 \leq (2|C| - |I|)/3$.

Proof. Let n_0, n_1, and n_2 be the number of degree-zero, degree-one, and degree-two clique vertices in F. Then, $n_0 + n_1 + n_2 = |C|$ (all clique vertices have degree 0, 1 or 2), $n_1 + 2n_2 = |I|$ (vertices in I have degree 1). Rearranging and resolving variable n_2 yields

$$3n_1 = 2|C| - |I| - 2(n_0 - n_1). \tag{2}$$

As mentioned, $2|C| - |I|$ is a multiple of three (as well as $|C| + |I|$). Note that $n_0 - n_1$, which is positive because F is feasible, is a multiple of three because it equals the number of clique vertices in P_3s of Type (i). Let j be an integer with $(n_0 - n_1)/3 = j$. Then, together with (2), we obtain that

$$\begin{aligned} n_1 &= \frac{2|C| - |I|}{3} - 2 \cdot \frac{n_0 - n_1}{3} \\ &= 2q + r - 2j. \end{aligned}$$

The last statement is satisfied since $2q + r = (2|C| - |I|)/3$ and $j \geq 0$. □

To be able use a perfect matching algorithm to solve our problem, we first reduce it to a restricted variant of the graph factor problem: We add an additional vertex z to the bipartite graph B, and connect it to all vertices in C. We call this graph B'. Now, the following lemma states that B' can be used to find a feasible factor for B.

Lemma 7. *The bipartite graph $B = (C \cup I, E)$ admits a feasible factor if and only if graph $B' = (C \cup I \cup \{z\}, E \cup \{\{z, c\} \mid c \in C\})$ has a factor satisfying the following degree constraints: (1) Every vertex in I has degree one, (2) every vertex in C has degree zero or two, and (3) the added vertex z has degree $2i + r$, where $r \in \{0, 1\}$ such that there is an integer q with $(2|C| - |I|)/3 = 2q + r$ and $i \in \{0, 1, \ldots, q\}$.*

Proof. Assume that B admits a feasible factor $F = (C \cup I, P_E)$. Then $F' = (C \cup I \cup \{z\}, P_E)$ is a factor of B'. For each degree-one vertex $v \in C$, we add edge $\{v, z\}$ to factor F'. By Property 5, we thus add $2i + r$ edges, with $0 \leq i \leq q$. It is easy to verify that the degree constraints stated in the lemma are satisfied.

Conversely, let F' be a factor for graph B' where every independent set vertex has degree one, every clique vertex has degree zero or two, and vertex z has degree $2i + r$ with $i \in \{0, 1, \ldots, q\}$. If we delete from F' all edges incident to vertex z, then we obtain a feasible factor F for B where the number n_1 of degree-one clique vertices is the original degree of z, that is,

$$n_1 = 2i + r \leq (2|C| - |I|)/3. \tag{3}$$

Since each independent set vertex still has degree one, the number of degree-two clique vertices is $(|I| - n_1)/2$, and the number of degree-zero clique vertices is $n_0 = |C| - n_1 - (|I| - n_1)/2 = (2|C| - |I| - n_1)/2$. The difference between the number of degree-zero and the number of degree-one vertices is $n_0 - n_1 = (2|C| - |I| - 3n_1)/2$, which is non-negative by using (3). □

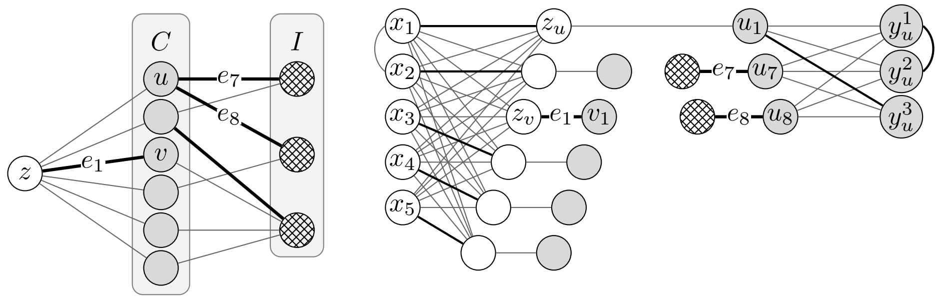


Figure 7: Left: An example of a factor for B' fulfilling the degree constraints as required in Lemma 7. The black thick bold edges reflects the constraints. Right: The gadget for vertex z and the gadget for the clique vertex u used to construct graph B^* (according to Construction 1). The black thick bold edges (labeled) correspond to the ones marked on the right. The black bold edges are additional matching edges.

Figure 7 (Left) depicts an example factor for the graph B' fulfilling the degree constraints of Lemma 7.

Using a gadget introduced by Cornuéjols [11], we can even reduce P_3-Partition to the perfect matching problem. We construct a graph B^* from B' in which we are searching for a perfect matching. The idea is to replace every clique vertex v by a gadget which can simulate the constraint that v has degree zero or two, and to replace vertex z by a gadget to simulate its degree constraint.

Construction 1. Let $m = |E|$ (number of edges between C and I in the original split graph) and d_u be the degree of a clique vertex u in C. Note that due to the edges going to vertex z, we have $\sum_{u \in C} d_u = m + |C|$. It holds that $|I| = O(m)$ and $|C| = O(m)$.

To construct the vertex set of B^*, first add a copy of I to B^*. For each clique vertex $u \in C$, add d_u vertices $y_u^1, y_u^2, \ldots, y_u^{d_u}$ to B^*; denote this set as $Y(u)$. For each edge $e_j \in E$ that is incident with u, add a vertex u_j to B^*; denote this set as $V(u)$. Note that $|Y(u)| = |V(u)|$ ($Y(u)$ and $V(u)$ are used to form a complete bipartite subgraph). For each edge $\{z, u\}$ in B', add a vertex z_u to B^*; denote this set as $V(z)$. Finally, add a set X of $|C| - r$ copies of vertex z, named as $x_1, x_2, \ldots, x_{|C|-r}$ to B^* ($V(z)$ and X are used to form a complete bipartite subgraph).

This completes the construction for the vertex set, which consists of $|I| + \sum_{u \in C} 2d_u + 2|C| - r = O(m)$ vertices.

Now we are ready to add edges to B^*. For each edge $e_j = \{u, v\}$ in B' which connects a clique vertex u and an independent set vertex v, add an edge $\{u_j, v\}$ to B^*. Analogously, for each edge $e_j = \{u, z\}$ in B' which connects z and a clique vertex u, add an edge $\{u_j, z_u\}$ to B^*. This is used to model the original edges of B'. Now, to model the degree constraints, for each clique vertex $u \in C$, add to B^* an edge between every vertex from $V(u)$ and every vertex from $Y(u)$, and an edge between y_u^1 and y_u^2. Add to B^* an edge between every vertex from

$V(z)$ and every vertex from X. Finally, for each integer i with $1 \leq i \leq q$, add to B^* edge $\{x_{2i-1}, x_{2i}\}$.

The overall number of edges in B^* is $(m+|C|)+\sum_{u\in C}(d_u)^2+|C|(|C|-r)+|C|+q \leq (m+|C|)^2+|C|^2+O(m) = O(m^2)$. This finishes the construction.

Figure 7 (Right), shows the gadget constructed for a clique vertex and the gadget for vertex z. Now, we show how the constructed graph B^* can be used to find a factor for B' satisfying the specific degree constraints as specified in Lemma 7.

Lemma 8. *Graph B^* constructed according to Construction 1 admits a perfect matching if and only if graph B' admits a factor F' satisfying the condition that (1) Every vertex in I has degree one, (2) every vertex in C has degree zero or two, and (3) the added vertex z has degree $2i+r$, where $r \in \{0,1\}$ such that there is an integer q with $(2|C|-|I|)/3 = 2q+r$ and $i \in \{0,1,\ldots,q\}$.*

Proof. Let M be a perfect matching for B^*. We construct a factor $F' = (C \cup I \cup \{z\}, P'_E)$ for B'. For each edge $\{u_j, v\} \in M$ which connects an independent set vertex v, add to P'_E edge $\{u,v\}$. For each edge $\{z_u, u_j\} \in M$ which connects vertices in $V(z)$ and $V(u)$, $u \in C$, add to P'_E edge $\{z,u\}$.

We show that F' is a factor for B^* satisfying the properties stated in the lemma. Obviously, every independent set vertex $u \in I$ has degree one. Consider a clique vertex $u \in C$. By the construction of graph B^*, in order to match all vertices in $Y(u)$, either (i) every vertex in $Y(u)$ has to be matched to a vertex in $V(u)$ or (ii) y^1_u and y^2_u are matched together while every vertex in $Y(u)\setminus\{y^1_u, y^2_u\}$ is matched to exactly one vertex in $V(u)$. This implies that either no vertex or exactly two vertices in $V(u)$ are matched to some vertices which are not from $Y(u)$. Thus, v has either degree zero or degree two in F'.

Analogously, by the construction of graph B^*, in order to match all vertices in X which has size $|C|-r$, exactly i pairs of vertices in X can be left without being matched to any vertex in $V(z)$ where $0 \leq i \leq q$ (note that only the first $2q$ vertices are connected by a path). This implies that exactly $|C|-(|C|-r-2i) = 2i+r$ vertices from $V(z)$ are matched to vertices that are not from X. Thus, z has degree $2i+r$.

Conversely, assume that B' admits a factor F' satisfying the above properties. We show that the following construction yields a perfect matching M for B^*.

For each edge $e_j = \{u,v\}$ in F' that connects a clique vertex u and an independent set vertex v, add to M edge $\{u_j, v\}$. For each edge $e_j = \{z,u\}$ in F' that connects vertex z with a clique vertex u, add to M edge $\{z_u, u_j\}$.

For each clique vertex $u \in C$, let $R(u) \subseteq V(u)$ be the set of vertices which are not yet matched by M. We need to match all vertices in $R(u)$. Depending on whether u has degree zero or two in F', $|V(u)|-|R(u)|$ is either zero or two, and $|Y(u)|-|R(u)| = |V(u)|-|R(u)|$. Moreover, $B^*[V(u)\cup Y(u)|$ contains a complete bipartite graph for $V(u)$ and $Y(u)$. If $|R(u)| = |V(u)|$ (which means that u has degree zero in F'), then add to M edges connecting exactly one vertex of $R(u)$ and one vertex of $Y(u)$; otherwise, $|R(u)| = |V(u)|-2$: add edge $\{y^1_u, y^2_u\}$ to M, and edges connecting exactly one vertex of $R(u)$ and one vertex of $Y(u) \setminus \{y_u^1, y_u^2\}$. Analogously, let $R(z) \subseteq V(z)$ be the set of vertices in $V(z)$ which are not yet matched by M. By assumption, there is an integer i, $0 \leq i \leq q$ such that $|V(z)| - |R(z)| = 2i + r$. Since $|X| = |V(z)| - r$, we have $|X| - |R(z)| = 2i$. Moreover, $B^*[V(z) \cup X|$ contains a complete bipartite graph for $V(z)$ and X. Thus, for each $1 \leq k \leq i$, add edge $\{x_{2k-1}, x_{2k}\}$ to M, and add edges connecting exactly one vertex of $R(z)$ and one vertex of X to match vertices of R_z and of $V_z' \setminus \{x_1, x_2, \ldots, x_{2i}\}$. It is easy to verify that M is indeed a perfect matching. $\square$

We now have gathered all ingredients to show Theorem 6.

Theorem 6. Star Partition *on split graphs is solvable in* $O(m^{2.5})$ *time for* $s = 2$.

Proof. Let $G = (C \cup V, C_E \cup E)$ be a split graph with m being the number of edges in E. Let $B' = (C \cup V \cup \{z\}, E \cup \{\{z, v\} \mid v \in C\})$ be a bipartite graph over C and $I \cup \{z\}$. Let B^* be computed from B' using Construction 1. By Lemmas 6 to 8, G admits a P_3-partition if and only if B^* admits a perfect matching. Since deciding whether a graph with s vertices and t edges has a perfect matching can be done in $O(t\sqrt{s})$ time [27, Theorem 16.4] and since B^* has $O(m)$ vertices and $O(m^2)$ edges, deciding whether G has a P_3-partition can be done in $O(m^{2.5})$ time. $\square$

In contrast, we can show that Star Partition is NP-hard for each $s \geq 3$ by a reduction from Exact Cover by s-Sets.

Theorem 7. Star Partition *on split graphs is NP-hard for* $s \geq 3$.

Proof. We show that it is NP-hard to find an s-star partition of a split graph via reduction from Exact Cover by s-Sets [16] (illustrated in Figure 8).

> Exact Cover by s-Sets
> *Input:* A finite set U and a collection $\mathcal{S}$ of size-s subsets of U.
> *Question:* Is there a subcollection $\mathcal{S}' \subseteq \mathcal{S}$ that partitions U (each element of U is contained in exactly one subset in $\mathcal{S}'$)?

Given $(U, \mathcal{S})$ with $|U| = sn$ and $|\mathcal{S}| = m \geq n$ where $n, m \in \mathbb{N}$, we construct a split graph $G = (C \cup I, E)$ as follows: The vertex set consists of a clique C and an independent set I. The clique C contains a vertex for each subset in $\mathcal{S}$, the independent set I contains a vertex for each element of U. For each $S \in \mathcal{S}$ the corresponding vertex in C is adjacent to the s vertices in I that correspond to the elements of S. Moreover, let $q, r \in \mathbb{N}$ such that $m - n = (s-1)q + r$. We add q dummy vertices to both C and I and connect every dummy vertex in C with all other vertices in C and uniquely with one of the dummy vertices in I. Finally, we add one more dummy vertex to C and another $s - r$ dummy vertices to I. This last dummy in C is connected to all other vertices in C and to each of the $s - r$ dummies in I. Note that each dummy vertex in I has degree one. The above construction can be carried out in polynomial time.

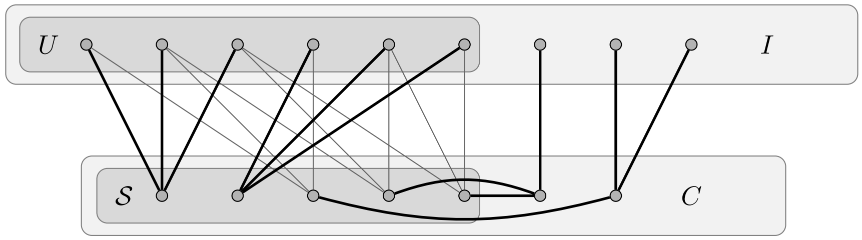


Figure 8: Reduction from Exact Cover by s-Sets to Star Partition with $s = 3$. Dummy vertices are in the light gray area and non-dummy vertices are in the dark gray area. The 3-star partition for the constructed graph G is indicated by thick edges.

Now, let $\mathcal{S}' \subseteq \mathcal{S}$ be a partition of U. Then we can partition G into stars of size s in the following way: For each $S \in \mathcal{S}'$, we choose the star containing the vertex from C corresponding to S and the vertices in I corresponding to the elements of S. Moreover, each of the dummy vertices in I is put together with its neighboring dummy in C and filled up to a star of size s with the remaining non-dummy vertices in C corresponding to the subsets in $\mathcal{S} \setminus \mathcal{S}'$. Indeed, the values of q and r are chosen in a way that guarantees that this is possible. Since $\mathcal{S}'$ partitions U, we get a valid s-star partition of G.

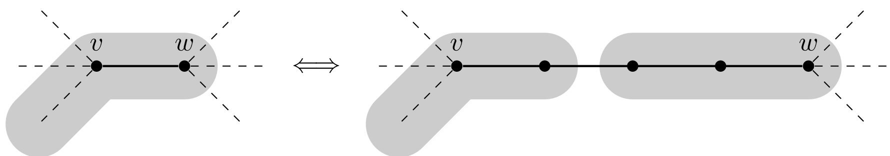
Conversely, in any s-star partition of G, all dummy vertices in I are grouped together into an s-star with their one dummy neighbor in C and the respective number of other non-dummy vertices from C. The values of q and r are such that there are exactly n non-dummy vertices left in C together with the $s \cdot n$ vertices in I corresponding to U. It follows that each remaining non-dummy vertex in C forms a star with its s neighbors in I, which yields a partition of U. □

6 Grid graphs

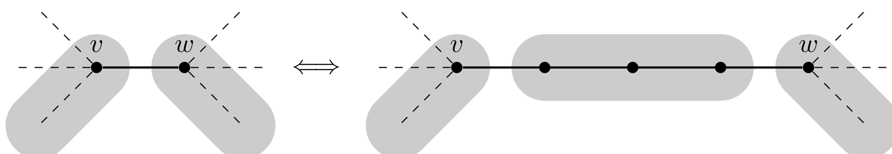
In this section, we show that P_3-Partition is NP-hard even on grid graphs with maximum degree three, thus strengthening a result of Małafiejski and Żyliński [22] and Monnot and Toulouse [23], who showed that P_3-Partition is NP-complete on planar bipartite graphs of maximum degree three.

A *grid graph* is a graph with a vertex set $V \subseteq \mathbb{N} \times \mathbb{N}$ and edge set $\{\{u, v\} \mid u = (i, j) \in V, v = (k, \ell) \in V, |i - k| + |j - \ell| = 1\}$. That is, its vertices can be given integer coordinates such that every pair of vertices is joined by an edge if and only if their coordinates differ by 1 in exactly one dimension.

To show NP-hardness of P_3-Partition on grid graphs, we exploit the above mentioned result of Małafiejski and Żyliński [22] and Monnot and Toulouse [23] and find a suitable embedding of planar graphs into grid graphs while maintaining the property of a graph having a P_3-partition. This allows us to prove the following.



(a) Case 1: Vertex v and vertex w are covered by the same P_3 (which implies that one vertex is an endpoint and the other vertex is an internal point of the P_3).



(b) Case 2: Vertex v and vertex w are covered by different P_3s (vertex v and/or vertex w can also be internal vertices).

Figure 9: All possibilities of two vertices v and w participating in a P_3-partition if they are joined by an edge or a path on three other degree-two vertices. Edges participating in the same P_3 are grouped together in a gray background.

Theorem 8. *P_3-PARTITION is NP-hard on grid graphs of maximum degree three.*

Towards proving Theorem 8, the following observation helps us embed planar graphs into grid graphs: it allows us to replace edges by paths on $3i$ new vertices for any $i \in \mathbb{N}$.

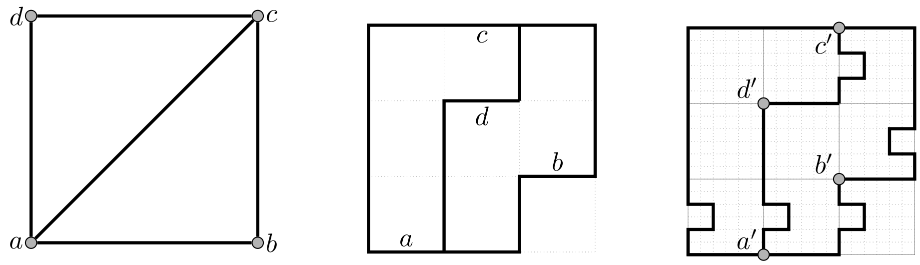
Observation 5. Let G be a graph, $e = \{v, w\}$ be an edge of G, and G' be the graph obtained by removing the edge e from G and by connecting v and w using a path on three new vertices. Then, G has a P_3-partition if and only if G' has one.

Note that the correctness of Observation 5 is proven by Figure 9, which enumerates all possible cases.

We can now prove Theorem 8 by showing that G has a P_3-partition if and only G' has, where G' is the graph obtained from a planar graph G of maximum degree three using the following construction.

Construction 2. Let G be a planar n-vertex graph of maximum degree three. Using a polynomial-time algorithm of Rosenstiehl and Tarjan [26] we obtain a crossing-free *rectilinear embedding* of G into the plane such that:

1. Each vertex is represented by a horizontal line.
2. Each edge is represented by a vertical line.
3. All lines end at integer coordinates with integers in $O(n)$.



(a) A planar graph G.

(b) A rectilinear embedding of G.

(c) The grid graph G' obtained from G.

Figure 10: Various embeddings of a planar graph. In the rectilinear embedding in Figure 10b, horizontal lines represent vertices of G, while vertical lines represent its edges. In Figure 10c, every intersection of a line with a grid point is a vertex, but only the vertices corresponding to vertices in Figure 10a are shown. The horizontal lines of the rectilinear embedding are now replaced by paths highlighted in gray.

4. If two vertices are joined by an edge, then the vertical line representing this edge ends on the horizontal lines representing the vertices.

Figure 10b illustrates such an embedding. Without loss of generality, every end point of a line lies on another line. Now, in polynomial time, we obtain a grid graph G' from the rectilinear embedding, as follows:

1. We multiply all coordinates by six (see Figure 10c).

2. Every point in the grid touched by a horizontal line that represents a vertex v of G becomes a vertex in G'. The horizontal path resulting from this horizontal line we denote by $P(v)$ (indicated by a gray background in Figure 10c).

3. For each vertical line, all its grid points become vertices in G', except for the third point from the bottom horizontal line that we bypass by adding a bend of five vertices to the vertical line (see Figure 10c).

4. With each vertex v in G, we associate the vertex v' of G' that lies on $P(v)$ and has degree three. There is at most one such vertex. If no such vertex exists, then we arbitrarily associate with v one of the end points of $P(v)$.

Proof of Theorem 8. Since Construction 2 runs in polynomial time, it remains to prove that a graph G has a P_3-partition if and only if the graph G' obtained by Construction 2 has. By Observation 5, it is sufficient to verify that every edge $e := \{u, v\}$ in G is replaced by a path p between u' and v' in G' whose number of inner vertices is divisible by three. To this end, we partition the path p into two parts: one part consists of the subpaths p_u, p_v of p that lie on $P(u)$ and $P(v)$, respectively. Note that each of p_u and p_v might consist only of one vertex, as seen for the path from d' to a' in Figure 10c. The other part is a path p_e that connects p_u to p_v. We consider p_e not to contain the vertices of p_u or p_v. Hence, p_e contains no vertices of any horizontal paths.

The number of *inner* vertices of p shared with the horizontal paths p_u and p_v is divisible by three (it is possibly zero) since all coordinates that start or end paths are divisible by three (in fact, by six). Herein, note that we do not count the vertices u' and v' lying on p_u or p_v, respectively.

Moreover, the number of vertices on p_e is also divisible by three: the number of vertices on a strictly vertical path p_s connecting p_u with p_v would leave a remainder of two when divided by three (as the two vertices on p_u and p_v are not considered to be part of p_s). However, our added bend of five new vertices makes p_e by four vertices longer compared to p_s. Hence, the number of vertices on p_e is also divisible by three. It follows that the total number of inner vertices of p is divisible by three. □

7 Chordal graphs

A graph is *chordal* if every induced subgraph containing a cycle of length at least four also contains a *triangle*, that is, a cycle of length three. We show that P_3-Partition restricted to chordal graphs is NP-hard (in contrast to the polynomial-time solvability on split graphs which form a subclass of chordal graphs) by reduction from 3-Dimensional Matching. More precisely, we use the construction that Dyer and Frieze [14] provided to show that P_3-Partition is NP-complete and observe that we can triangulate the resulting graph while maintaining the correctness of the reduction.

3-Dimensional Matching (3DM)
Input: Pairwise disjoint sets R, B, Y with $|R| = |B| = |Y| = q$ and a set of triples $T \subseteq R \times B \times Y$.
Question: Does there exist a *perfect 3-dimensional matching* $M \subseteq T$, that is, $|M| = q$ and each element of $R \cup B \cup Y$ occurs in exactly one triple of M?

Dyer and Frieze [14] introduced Construction 3 described below and illustrated in Figure 11). Using it as a reduction from the NP-complete restriction of 3DM to planar graphs [15], they proved that P_3-Partition restricted to bipartite planar graphs is NP-complete.

Construction 3. Let $(R, B, Y, T \subseteq R \times B \times Y)$ with $|R| = |B| = |Y| = q$ be an instance of 3DM. Construct a graph $G = (V, E)$ as follows: For each element $a \in R \cup B \cup Y$, create two vertices u_a, u'_a and connect them by an edge $\{u_a, u'_a\}$. We call u_a an *element-vertex* and u'_a a *pendant-vertex*. For each triple $t = (r, b, y) \in T$, create three vertices v^t_r, v^t_b, v^t_y. We call these three vertices *triple-vertices*. Make triple-vertex v^t_b adjacent to both v^t_r and v^t_y. Also

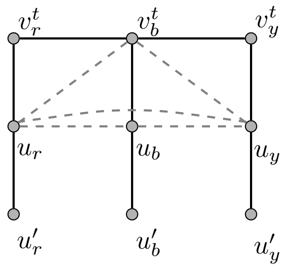


Figure 11: Gadget for a triple $t = (r, b, y) \in T$ based on Construction 3 (solid edges). The reduction in the proof of Theorem 9 introduces the dashed edges.

make triple-vertex v_r^t (respectively v_b^t and v_y^t) adjacent to element-vertex u_r (respectively u_b and u_y). Formally,

$$\begin{aligned} V &= \{u_a, u'_a \mid a \in R \cup B \cup Y\} \cup \{v_r^t, v_b^t, v_y^t \mid t = (r, b, y) \in T\}, \text{ and} \\ E &= \{\{u_a, u'_a\} \mid a \in R \cup B \cup Y\} \cup \\ &\quad \{\{u_r, v_r^t\}, \{u_b, v_b^t\}, \{u_y, v_y^t\}, \{v_r^t, v_b^t\}, \{v_b^t, v_y^t\} \mid t = (r, b, y) \in T\}. \end{aligned}$$

Theorem 9. P_3-PARTITION *restricted to chordal graphs is NP-hard.*

Proof. We extend Construction 3 to show the NP-hardness of P_3-PARTITION restricted to chordal graphs. Make any two element-vertices adjacent to each other such that the graph induced by all element-vertices is complete. Furthermore, for each triple $t = (r, b, y) \in T$, add two edges $\{v_b^t, u_r\}$ and $\{v_b^t, u_y\}$ to the graph, as illustrated in Figure 11. Let G be the resulting graph.

We first show that G is chordal. Consider any size-ℓ set C of vertices with $\ell \geq 4$ such that the subgraph G_C induced by C contains a simple cycle of length ℓ. Since the pendant-vertices all have degree one, C cannot contain any pendant-vertex. If C contains a degree-two triple-vertex v_r^t (respectively v_y^t) for some $t = (r, b, y) \in T$, then C contains both its neighbors: v_b^t and u_r (respectively v_b^t and u_y) which are connected, that is, forming a triangle. Otherwise, if C contains a degree-five triple-vertex v_b^t but does not contain v_r^t nor v_y^t, then, in the cycle, v_b^t lies between two element-vertices. Since two element-vertices are always connected, G_C contains a triangle. Finally, if C does not contain any triple-vertex, then it is included in the set of element-vertices, which form a complete graph. Hence, G_C contains a triangle.

Second, we show that $(R, B, Y, T \subseteq R \times B \times Y)$ has a perfect 3-dimensional matching if and only if G can be partitioned into P_3s.

For the "only if" part, suppose that $M \subseteq T$ is a perfect 3-dimensional matching for $(R, B, Y, T \subseteq R \times B \times Y)$. Then, the P_3s in

$$\Big(\bigcup_{t=(r,b,y)\notin M} \{\{v_r^t, v_b^t, v_y^t\}\}\Big) \cup \Big(\bigcup_{t=(r,b,y)\in M} \{\{u'_r, u_r, v_r^t\}, \{u'_b, u_b, v_b^t\}, \{u'_y, u_y, v_y^t\}\}\Big)$$

indeed partition the graph G.

For the "if" part, suppose that G has a partition P into P_3s. We first enumerate the possible centers of the P_3s. Since each pendant-vertex is only adjacent to its element-vertex, every element-vertex is the center of a P_3 that contains a pendant-vertex. We call such a P_3 an *element-P_3*. For each triple $t = (r,b,y) \in T$, neither v_r^t nor v_y^t can be the center of a P_3 since they are adjacent to only one vertex which is not already a center (namely v_b^t). Thus, any P_3 which is not an element-P_3 must have vertex v_b^t as a center for some $t = (r,b,y) \in T$. We call such a P_3 a *triple-P_3 corresponding to t*.

Now, consider a triple $t = (r,b,y) \in T$. If there exists a triple-P_3 corresponding to t (that is, with center v_b^t), then its two leaves can only be v_r^t and v_y^t (since u_r^t, u_b^t, and u_y^t are centers). Otherwise, each of the three triple-vertices v_r^t, v_b^t, and v_y^t must be a leaf of an element-P_3 centered on u_r, u_b, and u_y. Indeed, there is only one way to match the three triple-vertices to these three element-vertices, that is, by using the edges $\{v_r^t, u_r\}$, $\{v_b^t, u_b\}$ and $\{v_y^t, u_y\}$. As a consequence, the leaves of the element-P_3 centered on u_a are u'_a and v_a^t for some triple t containing a.

It remains to show that the triples with *no* corresponding triple-P_3 in P form a perfect 3-dimensional matching M for $(R,B,Y,T \subseteq R \times B \times Y)$. Note that for each element $a \in R \cup B \cup Y$, the element-P_3 centered in u_a uses a triple-vertex v_a^t for some triple t containing a, which means that no triple-P_3 in P corresponds to t. Hence, $t \in M$ and element a is matched by t. Now, it remains to show that every element is matched at most once. Suppose for the sake of contradiction that there is an element $a \in R \cup B \cup Y$ which is matched at least twice. To this end, let v_a^t be a triple-vertex that together with u_a and u'_a forms an element-P_3. Thus, $t \in M$. Furthermore, let t' be another triple in M that matches a. Since t' has no corresponding triple-P_3 in P, there is an element-P_3 containing $v_a^{t'}$. But then, $v_a^{t'}$ must form an element-P_3 together with u_a and u'_a, which is a contradiction. □

8 Conclusion

We close with three open questions for future research. What is the complexity of Star Partition for $s \geq 2$ on permutation graphs? What is the complexity of Star Partition for $s \geq 3$ on interval graphs? Are there other important graph classes (not necessarily perfect ones) where Star Partition is polynomial-time solvable?

Acknowledgments. René van Bevern was supported by the Russian Foundation for Basic Research (RFBR), project 16-31-60007 mol_a_dk, at Novosibirsk State University, and by the German Research Foundation (DFG), project DAPA (NI 369/12), at TU Berlin. Robert Bredereck was supported by the DFG, project PAWS (NI 369/10). Laurent Bulteau and Gerhard J. Woeginger were supported by the Alexander von Humboldt Foundation, Bonn, Germany, while visiting TU Berlin. Jiehua Chen was supported by the Studienstiftung des Deutschen Volkes. Vincent Froese was supported by the DFG, project DAMM (NI 369/13).

This work started at the yearly research retreat of the group "Algorithms and Computational Complexity TU Berlin" in March 2013, held in Bad Schandau, Germany.

References

[1] K. Asdre and S. D. Nikolopoulos. NP-completeness results for some problems on subclasses of bipartite and chordal graphs. *Theoretical Computer Science*, 381(1-3):248–259, 2007.

[2] F. Berman, D. Johnson, T. Leighton, P. W. Shor, and L. Snyder. Generalized planar matching. *Journal of Algorithms*, 11(2):153–184, 1990.

[3] R. van Bevern, C. Komusiewicz, H. Moser, and R. Niedermeier. Measuring indifference: Unit Interval Vertex Deletion. In *Proceedings of the 36th International Workshop on Graph-Theoretic Concepts in Computer Science (WG '10)*, volume 6410 of *LNCS*, pages 232–243. Springer, 2010.

[4] R. van Bevern, R. Bredereck, J. Chen, V. Froese, R. Niedermeier, and G. J. Woeginger. Network-based vertex dissolution. *SIAM Journal on Discrete Mathematics*, 29(2):888–914, 2015.

[5] R. van Bevern, M. Mnich, R. Niedermeier, and M. Weller. Interval scheduling and colorful independent sets. *Journal of Scheduling*, 18(5):449–469, 2015.

[6] A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph Classes: A Survey*, volume 3 of *SIAM Monographs on Discrete Mathematics and Applications*. SIAM, 1999.

[7] J. Chalopin and D. Paulusma. Packing bipartite graphs with covers of complete bipartite graphs. *Discrete Applied Mathematics*, 168:40–50, 2014.

[8] E. Cohen and M. Tarsi. NP-completeness of graph decomposition problems. *Journal of Complexity*, 7(2):200–212, 1991.

[9] D. G. Corneil, Y. Perl, and L. Stewart. A linear recognition algorithm for cographs. *SIAM Journal on Computing*, 14(4):926–934, 1985.

[10] D. G. Corneil, S. Olariu, and L. Stewart. The LBFS structure and recognition of interval graphs. *SIAM Journal on Discrete Mathematics*, 23(4):1905–1953, 2009.

[11] G. Cornuéjols. General factors of graphs. *Journal of Combinatorial Theory. Series B*, 45(2):185–198, 1988.

[12] E. Dahlhaus and M. Karpinski. Matching and multidimensional matching in chordal and strongly chordal graphs. *Discrete Applied Mathematics*, 84 (1-3):79–91, 1998.

[13] K. M. J. De Bontridder, B. V. Halldórsson, M. M. Halldórsson, C. A. J. Hurkens, J. K. Lenstra, R. Ravi, and L. Stougie. Approximation algorithms for the test cover problem. *Mathematical Programming*, 98(1-3):477–491, 2003.

[14] M. E. Dyer and A. M. Frieze. On the complexity of partitioning graphs into connected subgraphs. *Discrete Applied Mathematics*, 10(2):139–153, 1985.

[15] M. E. Dyer and A. M. Frieze. Planar 3DM is NP-complete. *Journal of Algorithms*, 7(2):174–184, 1986.

[16] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.

[17] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Annals of Discrete Mathematics. Elsevier, Amsterdam, Boston, Paris, 2004.

[18] D. G. Kirkpatrick and P. Hell. On the complexity of general graph factor problems. *SIAM Journal on Computing*, 12(3):601–608, 1983.

[19] A. W. J. Kolen, J. K. Lenstra, C. H. Papadimitriou, and F. C. R. Spieksma. Interval scheduling: A survey. *Naval Research Logistics*, 54(5):530–543, 2007.

[20] A. Kosowski, M. Małafiejski, and P. Żyĺinski. Parallel processing subsystems with redundancy in a distributed environment. In *Proceedings of the 6th International Conference on Parallel Processing and Applied Mathematics (PPAM '05)*, volume 3911 of *LNCS*, pages 1002–1009. Springer, 2006.

[21] L. M. Lovász, C. Thomassen, Y. Wu, and C. Zhang. Nowhere-zero 3-flows and modulo k-orientations. *Journal of Combinatorial Theory. Series B*, 103 (5):587–598, 2013.

[22] M. Małafiejski and P. Żyliński. Weakly cooperative guards in grids. In *Proceedings of the International Conference on Computational Science and Its Applications (ICCSA '05)*, volume 3480 of *LNCS*, pages 647–656. Springer, 2005.

[23] J. Monnot and S. Toulouse. The path partition problem and related problems in bipartite graphs. *Operations Research Letters*, 35(5):677–684, 2007.

[24] B. S. Panda and S. K. Das. A linear time recognition algorithm for proper interval graphs. *Information Processing Letters*, 87(3):153–161, 2003.

[25] J. M. M. van Rooij, M. E. van Kooten Niekerk, and H. L. Bodlaender. Partition into triangles on bounded degree graphs. *Theory of Computing Systems*, 52(4):687–718, 2013.

[26] P. Rosenstiehl and R. E. Tarjan. Rectilinear planar layouts and bipolar orientations of planar graphs. *Discrete & Computational Geometry*, 1(1): 343–353, 1986.

[27] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume A. Springer, 2003.

[28] J. Spinrad, A. Brandstädt, and L. Stewart. Bipartite permutation graphs. *Discrete Applied Mathematics*, 18(3):279–292, 1987.

[29] G. Steiner. On the k-path partition problem in cographs. *Congressus Numerantium*, 147:89–96, 2000.

[30] G. Steiner. On the k-path partition of graphs. *Theoretical Computer Science*, 290(3):2147–2155, 2003.

[31] K. Takamizawa, T. Nishizeki, and N. Saito. Linear-time computability of combinatorial problems on series-parallel graphs. *Journal of the ACM*, 29 (3):623–641, 1982.

[32] C. Thomassen. The weak 3-flow conjecture and the weak circular flow conjecture. *Journal of Combinatorial Theory. Series B*, 102(2):521–529, 2012.

[33] J.-H. Yan, J.-J. Chen, and G. J. Chang. Quasi-threshold graphs. *Discrete Appl. Math.*, 69(3):247–255, 1996.

[34] J.-H. Yan, G. J. Chang, S. M. Hedetniemi, and S. T. Hedetniemi. k-path partitions in trees. *Discrete Applied Mathematics*, 78(1-3):227–233, 1997.

[35] R. Yuster. Combinatorial and computational aspects of graph packing and graph decomposition. *Computer Science Review*, 1(1):12–26, 2007.