Star partitions of perfect graphs

Star Partitions of Perfect Graphs

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

[1]Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Germany, {rene.vanbevern, robert.bredereck, jiehua.chen, vincent.froese}@tu-berlin.de, l.bulteau@campus.tu-berlin.de

[2]Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands, gwoegi@win.tue.nl

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 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 and NP-hard cases, for example, on interval graphs, grid graphs, and bipartite permutation 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 $k := \lceil n/(s+1) \rceil$ disjoint 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 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 [12]. The close connection of Star Partition to perfect matchings makes it a prime candidate for many applications.

Our goal in this paper is to achieve a better understanding of star partitions of certain classes 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 few potentially challenging cases open; see Figure 1 for an overview of our results.

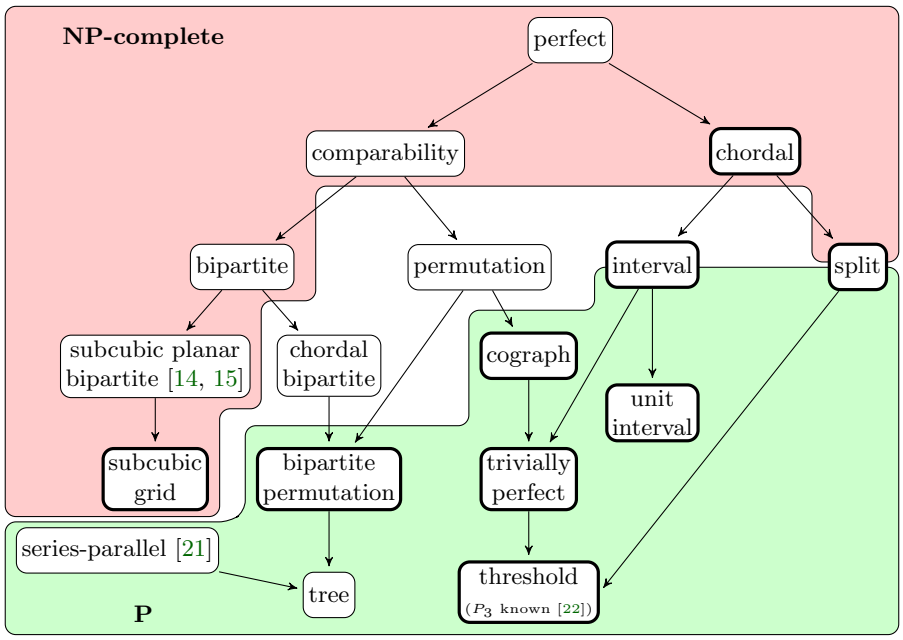


Figure 1: Complexity classification of Star Partition. Bold borders indicate results of this paper. An arrow from a class A to a class B indicates that A contains B. In most classes, NP-completeness results hold for $s = 2$ (that is, for P_3-Partition). However, on split graphs, Star Partition is polynomial-time 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 graph or interval graphs when $s \geq 3$.

Motivation. The literature in algorithmic graph theory is full with packing and partitioning problems. From a more applied point of view, P_3-Packing and P_3-Partition find applications in dividing distributed systems into subsystems [13] as well as in the Test Cover problem arising in bioinformatics [8]. In particular, the application in distributed systems explicitly motivates the consideration of very restricted (perfect) graph classes such as grid-like structures.

Let us discuss a few specific application scenarios for Star Partition with respect to important subclasses of *perfect graphs*. First, Star Partition on grid graphs naturally occurs in political redistricting problems [3]. We show that Star Partition remains NP-complete on subcubic grid graphs.

Interval graphs are another 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 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 it 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 interval graphs.

Finally, consider split graphs, another class of perfect graphs, whose vertices can be partitioned into a clique and an independent set. Assume that we have agents that are all employees of one institution and agents that are the institution's potential clients. Once again, the goal is to form teams of equal size, each containing a team leader as information hub. If we require the team leader to be a member of the institution, this immediately leads to Star Partition on split graphs: the clique vertices represent the institution agents and the independent set vertices represent the potential clients. We show that Star Partition restricted to split graphs is polynomial-time solvable for $s = 2$, but NP-complete otherwise.

Previous work. Packing and partitioning problems are central problems in algorithmic graph theory with many applications and with close connections to matching theory [24]. 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 a partitioning problem that has also been called H-Factor in the literature. In this work, we always refer to it as H-Partition. As Kirkpatrick and Hell [12] 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 [21] and on graphs of maximum degree two [15]. On the downside, P_k-Partition (for fixed $k \geq 3$) remains NP-complete on planar bipartite graphs [9]; this hardness result generalizes to H-Partition on planar graphs for any outerplanar pattern H with at least three vertices [2]. Partitioning into triangles, that is, K_3-Partition, is polynomial-time solvable on chordal graphs [7] and linear-time solvable on graphs of maximum degree three [16].

Optimization versions of P_k-Partition, called Min P_k-Partition, have also received considerable interest in the literature. This version asks for a partition of a given graph into a minimum number of paths of length *at most* k. Clearly, all hardness results for P_k-Partition carry over to the minimization version. If k is *part of* the input, then Min P_k-Partition is hard for cographs [19] and chordal bipartite graphs [20]. In fact, Min P_k-Partition is NP-hard 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 [23], threshold graphs, cographs (for fixed k) [19] and bipartite permutation graphs [20].

Our contributions. So far, surprisingly little has been known about the complexity of Star Partition for subclasses of perfect graphs. We provide a

detailed picture of the complexity landscape of perfect graphs; see Figure 1 for an overview. Let us briefly summarize some of our results.

As a central result, we provide a quasilinear-time algorithm for P_3-Partition on interval graphs; the complexity of Star Partition for $s \geq 3$ remains open. Furthermore, we develop a polynomial-time algorithm for Star Partition on cographs. 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 to NP-hardness, we strengthen a result of Małafiejski and Żyliński [14] and Monnot and Toulouse [15] by showing that P_3-Partition, which is Star Partition with $s = 2$, is NP-complete 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 [16]. Furthermore, P_3-Partition is NP-complete on chordal graphs, while K_3-Partition is polynomial-time solvable in this case [7]. We observe that P_3-Partition is typically not easier than Star Partition for $s \geq 3$. An exception to this rule is provided by the class of split graphs, where P_3-Partition is polynomial-time solvable but Star Partition is NP-complete for any constant value $s \geq 3$. Due to space constraints, most of our proofs have been moved to an appendix.

Preliminaries. We assume basic familiarity with standard graph classes [4, 10]. Definitions of the graph classes are provided when first studied in this paper. The graph $K_{1,s}$ is often called an s-star in the following. 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 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 *arbitrary $K_{1,s}$-subgraph of $G[V_i]$*. P_3-Partition is the special case of Star Partition ($s = 2$), where the star is a P_3, that is, a path on three vertices. 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). Throughout the paper, we denote by $n := |V|$ the number of vertices and by $m := |E|$ the number of edges in a graph $G = (V, E)$.

2 Interval graphs

In this section, we present algorithms that solve Star Partition on unit interval graphs in linear time and 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. In a *unit interval graph*, all representing intervals have the same length.

Star Partition on unit interval graphs

The restricted structure of unit interval graphs allows us to solve Star Partition using a simple greedy approach: repeatedly select the $s+1$ leftmost intervals to form an s-star and then deleting them. If, at some point, the $s+1$ leftmost intervals do not contain an s-star, it can be shown that the graph cannot be partitioned into s-stars. This algorithm yields the following result.

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

P_3-Partition on interval graphs

While it might come unsurprisingly that Star Partition can be solved efficiently on unit interval graphs using a greedy strategy, this is far from obvious for general interval graphs. 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 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 Figure 1), 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 edges called *links*. We associate with each interval two so-called "handles". We require that the link between a leaf and a center "uses" both of the leaf's handles (such that a leaf can have only one link) and one handle 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 say a link $\{x, y\}$ "consumes" the handles 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 free handles ends, but there are not enough free handles in other intervals to create a link. It is a yes-instance if the number of handles is always sufficient.

The algorithm works according to the following two rules: when an interval starts, its two handles are added to a list of free handles; when an interval with free handles ends, then three handles are deleted from this list (ideally: the handles of the earliest-ending intervals). The algorithm is sketched in Algorithm 1. Figure 2 shows an example instance and the lists of handles created by the algorithm. Note that a handle of an interval x is simply represented by a copy of interval x itself. We now introduce the necessary formal definitions.

Definitions. We consider a fixed interval graph $G = (V, E)$. We assume that any vertex $u \in V$ represents a right-open interval $u = [\text{start}(u), \text{end}(u)[$ with integer endpoints $\text{start}(u) < \text{end}(u)$. Moreover, without loss of generality, each position in $(1, \ldots, 2n)$ corresponds to exactly one event.

Algorithm 1: P_3-partition of an interval graph

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

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

```
A_0 ← empty handle list ∅;
for t ← 1 to 2n do
    if t = start(x) then A_t ← A_{t−1} ⊕ (x, x) if t = end(x) then
        if x ∉ A_{t−1} then A_t ← A_{t−1} else if ‖A_{t−1}‖ < 3 then return false
        else
            (x, y, z) ← lowest three elements of A_{t−1} (intervals ending first);
            A_t ← A_{t−1} ⊖ (x, y, z);
return true;
```

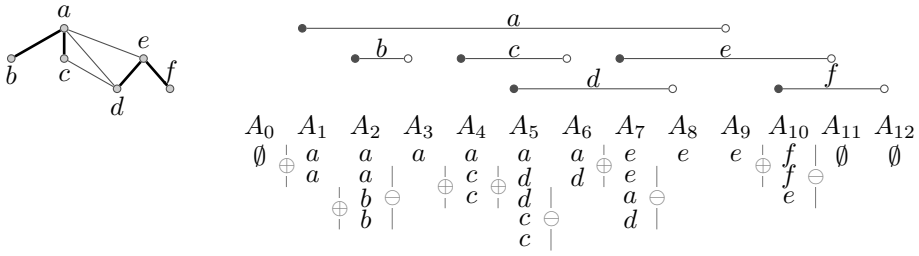


Figure 2: Left: Interval graph with six vertices and a P_3-partition $\mathcal{P}$ (bold). Right: Interval representation of this graph and successive handle lists $A_0, \ldots, A_{12}$ computed by Algorithm 1 (additions and deletions are marked with $\oplus$ and $\ominus$).

Given $P = \{x, y, z\}$ containing a P_3 with $\text{end}(x) < \text{end}(y) < \text{end}(z)$, we write $\text{rank}(x) = 1$, $\text{rank}(y) = 2$, and $\text{rank}(z) = 3$. 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 *handle list* Q is a list of intervals $(q_1, \ldots, q_k)$ sorted by decreasing end points ($\text{end}(q_i) \geq \text{end}(q_j)$ for $1 \leq i \leq j \leq k$). We consider the list to be represented vertically, with the latest-ending interval on top. We write $\|Q\|$ for the length of Q, $\emptyset$ for the empty handle list, and $x \in Q$ if interval x appears in Q. We now define insertion, deletion and comparison of handle lists: $Q \oplus (x_1, \ldots, x_l)$ is the handle list obtained from Q by inserting intervals $x_1 \ldots, x_l$ 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 $Q \ominus (x_1, \ldots, x_l) = Q \ominus x_1 \ominus \ldots \ominus x_l$. We write $(q_1, \ldots, q_k) \preccurlyeq (q'_1, \ldots, q'_{k'})$ if $k \leq k'$ and $\forall i \in \{1, \ldots k\}, \text{end}(q_i) \leq \text{end}(q'_i)$.

Let $\mathcal{P}$ be a P_3-partition. We define handles($\mathcal{P}$) as a tuple of $2n + 1$ handle lists $(H_0, H_1, \ldots, H_{2n})$ such that $H_0 := \emptyset$ and for $t > 0$,

- if $t = \text{start}(x)$, then $H_t := H_{t-1} \oplus (x, x)$,
- if $t = \text{end}(x)$, then let $P := \{x, y, z\}$ be the P_3 in $\mathcal{P}$ containing x and

– if $\operatorname{rank}(x) = 1$, then $H_t := H_{t-1} \ominus (x, x, c)$ where c is the center of P,
– if $\operatorname{rank}(x) = 2$, then $H_t := H_{t-1} \ominus (x, x, y, y, z, z)$,
– if $\operatorname{rank}(x) = 3$, then $H_t := H_{t-1}$.

Note that in Figure 2, each handle list H_t for $\mathcal{P}$ is equal to the respective A_t, except for $H_6 = (d, d)$ and $H_7 = (e, e, d, d)$.

The following lemmas state that, on the one hand, if there is a P_3-partition, then Algorithm 1 will find a sequence of handle lists that always contains enough handles to 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.

Lemma 1. *If an interval graph G has a P_3-partition $\mathcal{P}$, then for all $0 \leq t \leq 2n$, Algorithm 1 defines set A_t with $H_t \preccurlyeq A_t$ and $\|H_t\| - \|A_t\| \equiv 0 \pmod 3$, where* $\operatorname{handles}(\mathcal{P}) = (H_0, H_1, \ldots, H_{2n})$.

Lemma 2. *Let G be an interval graph such that Algorithm 1 returns `true` on G. Then there exists a P_3-partition of G.*

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

Theorem 2. *P_3-Partition on interval graphs is solvable in $O(n \log n + m)$ time.*

3 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 can be characterized using *strong orderings* of the vertices of a bipartite graph:

Definition 1 (Spinrad et al. [18])**.** 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 $\{u, w\}$, $\{u', w'\}$ in E, where $u, u' \in U$ and $w, w' \in W$, $u \prec u'$ and $w' \prec w$ implies that $\{u, w'\}$ and $\{u', w\}$ are 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. In this case a strong ordering can be computed in linear time [18].

Our key to obtain star partitions in 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 solution exploiting these normal forms. We sketch these structural properties of an s-star partition of bipartite (permutation) graphs in the following.

Definition 2. 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 $\prec$. Assume that G admits an s-star partition $\mathcal{P}$.

Let $X \in \mathcal{P}$ form a star. We denote $\operatorname{lm}(X)$ (resp. $\operatorname{rm}(X)$) as the leftmost (resp. rightmost) leaf of X with respect to $\prec$. The *scope* of star X is the

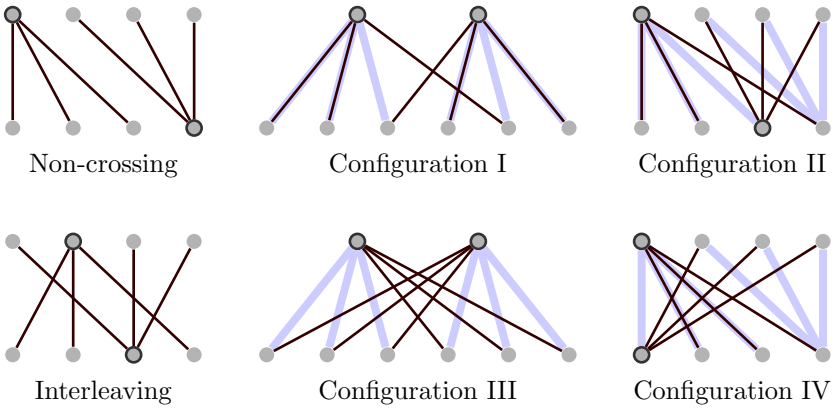


Figure 3: Possible interactions between two stars of a partition. Centers are marked with circled nodes. By Lemma 3, any partition containing one of the configurations I to IV can be edited to reduce the score (see the thick light-color edges).

set $\text{scope}(X) := \{v \mid x_l \preccurlyeq v \preccurlyeq x_r\}$ containing all vertices from $x_l = \text{lm}(X)$ to $x_r = \text{rm}(X)$. The *width* of star X is the cardinality of its scope, that is, $\text{width}(X) := |\text{scope}(X)| = r - l + 1$. The *width* of $\mathcal{P}$, $\text{width}(\mathcal{P})$, is the sum of $\text{width}(X)$ 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 either ($u \prec u'$ and $w' \prec w$) or ($u' \prec u$ 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' 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 3. 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;
- *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.

This definition allows us to show a normal form for star partitions in bipartite permutation graphs.

Lemma 3. *Any s-star partition of a bipartite permutation graph G with minimum score is a good s-star partition.*

Corollary 1. *Let $\mathcal{P}$ be an s-star partition of a bipartite permutation graph G with minimum score. Then for every 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.*

We now informally describe a dynamic programming algorithm focused on deciding whether there is 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 a star is added with the center in U, or a star is added with the center in W, or 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 above. This algorithm leads to the following result.

Theorem 3. Star Partition *can be solved in $O(n^2)$ time on bipartite permutation graphs.*

4 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 [14] and Monnot and Toulouse [15], 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 $\{\{u, v\} \mid u = (i, j) \in V, v = (k, \ell) \in V, |i - j| + |k - \ell| \leq 1\}$. 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 at most 1 in at most one dimension.

To show NP-hardness of P_3-Partition on grid graphs, we exploit the above mentioned result of Małafiejski and Żyliński [14] and Monnot and Toulouse [15] 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

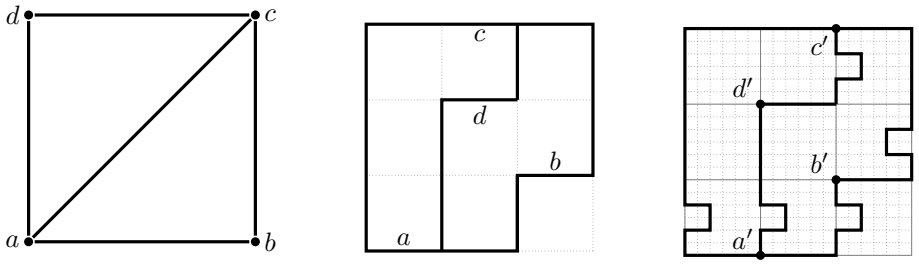
Theorem 4. P_3-Partition *is NP-hard on grid graphs of maximum degree three.*

The following observation helps us embedding planar graphs into grid graphs, as it allows us to replace edges by paths on $3i$ new vertices for any $i \in \mathbb{N}$.

Observation 1. 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.

We can now prove Theorem 4 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 1. Let G be a planar graph of maximum degree three. Using a polynomial-time algorithm of Rosenstiehl and Tarjan [17] we obtain a crossing-free *rectilinear embedding* of G into the plane such that:



(a) A planar graph G.

(b) A rectilinear embedding of G.

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

Figure 4: Various embeddings of a planar graph. In the rectilinear embedding in Figure 4b, horizontal lines represent vertices of G, while vertical lines represent its edges. In Figure 4c, every intersection of a line with a grid point is a vertex, but only the vertices corresponding to vertices in Figure 4a are shown.

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 $O(n)$.
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 4b 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 4c).
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 $P(v)$.
3. For each vertical line, all its grid points become a vertex in G', except for one point that we bypass by adding a bend of five vertices to the vertical line (see Figure 4c).
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 $P(v)$.

5 Further results

This section briefly summarizes our hardness and tractability results for cographs, split graphs, and chordal graphs.

Cographs. A cograph is a graph that does not contain a P_4 (path of length four) as an induced subgraph. Cographs allow for a so-called linear-time computable *cotree* [5]. Using a dynamic programming approach on the cotree representation of the cograph, we can solve Star Partition in polynomial time.

Theorem 5. Star Partition *can be solved in* $O(kn^2)$ *time on cographs.*

Split graphs. A *split graph* is a graph whose vertices can be partitioned into a clique and an independent set. Remarkably, split graphs are the only graph class where we could show that P_3-Partition is solvable in polynomial time, but Star Partition for $k \geq 3$ 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 [6].

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

Theorem 6. Star Partition *on split graphs is solvable in* $O(m^{2.5})$ *time for* $s = 2$*, but is NP-hard for each* $s \geq 3$*.*

Chordal graphs. A graph is *chordal* if any 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 by reduction from the well-known 3-Dimensional Matching problem [11].

Theorem 7. P_3-Partition *restricted to chordal graphs is NP-hard.*

For the reduction, we use the construction that Dyer and Frieze [9] used 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.

6 Conclusion

With this paper, we worked towards closing a wide gap concerning the assessment of the computational complexity of a class of fundamental graph partitioning problems—partitioning perfect graphs into stars. 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 DFG, project DAPA (NI 369/12), Robert Bredereck by the DFG, project PAWS (NI 369/10), Vincent Froese by the DFG, project DAMM (NI 369/13), Laurent Bulteau and Gerhard J. Woeginger by the Alexander von Humboldt Foundation, Bonn, Germany, and Jiehua Chen by the Studienstiftung des Deutschen Volkes.

Bibliography

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

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

[3] R. van Bevern, R. Bredereck, J. Chen, V. Froese, R. Niedermeier, and G. J. Woeginger. Network-based dissolution. Manuscript, TU Berlin, Feb. 2014. `arXiv:1402.2664 [cs.DM]`.

[4] 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.

[5] D. Corneil, Y. Perl, and L. Stewart. A linear recognition algorithm for cographs. *SIAM J. Comput.*, 14(4):926–934, 1985.

[6] G. Cornuéjols. General factors of graphs. *J. Combin. Theory Ser. B*, 45(2): 185–198, 1988.

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

[8] 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. *Math. Program.*, 98(1-3):477–491, 2003.

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

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

[11] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum, New York, 1972.

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

[13] A. Kosowski, M. Małafiejski, and P. Żyĺinski. Parallel processing subsystems with redundancy in a distributed environment. In *Parallel Processing and Applied Mathematics, 6th International Conference, PPAM 2005*, volume 3911 of *LNCS*, pages 1002–1009. Springer, 2006.

[14] M. Małafiejski and P. Żyliński. Weakly cooperative guards in grids. In *Proc. 5th ICCSA*, volume 3480 of *LNCS*, pages 647–656, 2005.

[15] J. Monnot and S. Toulouse. The path partition problem and related problems in bipartite graphs. *Oper. Res. Lett.*, 35(5):677–684, 2007.

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

[17] P. Rosenstiehl and R. E. Tarjan. Rectilinear planar layouts and bipolar orientations of planar graphs. *Discrete Comput. Geom.*, 1(1):343–353, 1986.

[18] J. Spinrad, A. Brandstädt, and L. Stewart. Bipartite permutation graphs.

Discrete Appl. Math., 18(3):279–292, 1987.

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

[20] G. Steiner. On the k-path partition of graphs. *Theor. Comput. Sci.*, 290(3): 2147–2155, 2003.

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

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

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

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

Appendix

A Proofs for Section 2 (Interval Graphs)

A.1 Proof of Theorem 1

Proof. Given a unit interval graph $G = (V, E)$, we can in linear time compute a *bicompatible elimination order* of its vertices [32], that is, an ordering $\sigma\colon V \to \{1, \ldots, |V|\}$ such that for any vertex $v \in V$ it holds that the set $N_l[v] := \{u \in N[v] \mid \sigma(u) \leq \sigma(v)\}$ of its left neighbors and the set $N_r[v] := \{u \in N[v] \mid \sigma(u) \geq \sigma(v)\}$ of its right neighbors each form a clique in G. Moreover, since we assume G to be connected, it can be shown that 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 [25]. For a subset $V' \subseteq V$ let $r(V') := \arg\max_{v \in V'} \sigma(v)$ denote the rightmost vertex in V' with respect to σ.

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

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

We now show that we can greedily cover S' by stars, because S' consists of two cliques where one vertex of the first clique is connected to all vertices of the second clique. To this end, let $u := r(N[v_{s+1}])$ be the rightmost neighbor of v_{s+1} It holds that $S' \subseteq N[u]$, because for a vertex $v' \in S'$ either $\sigma(v_{s+1}) \leq \sigma(v') \leq \sigma(u)$ or $\sigma(u) \leq \sigma(v')$ and there is some star center $v'' \in S'$ with $\sigma(v'') \leq \sigma(u)$ and $\{v'', v'\} \in E$ since no star S_i could have a center c_i with $\sigma(c_i) > \sigma(u)$. In particular, all vertices in $S' \cap N_r[u]$ are connected to the rightmost vertex $x := r(S' \cap N_l[u])$ of $S' \cap N_l[u]$. The vertices in $S' \cap N_l[u]$ are also connected to x as they induce a clique including x. Moreover, $N_r[u]$ also induces a clique, because x is, in particular, connected to the rightmost vertex in $N_r[u]$.

Now, since $S' \cap N_r[u]$ induces a clique in G, we arbitrarily pack as many s-stars as possible into $S' \cap N_r[u]$. Any remaining vertices are grouped together

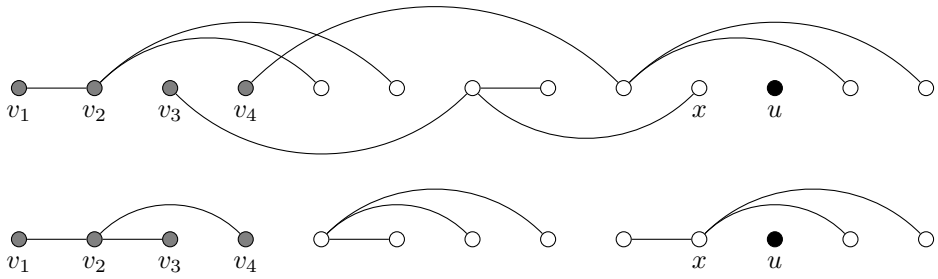


Figure 5: Example of a 3-star partition of a unit interval graph with vertices ordered according to a bicompatible elimination order from left to right. Only the edges and vertices of the first three stars as well as the vertex u (black) are shown. Top: $v_1, \ldots, v_4$ are not grouped together into 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.

with x and the necessary number of arbitrary vertices from $S' \cap N_l[u]$. Finally, the remaining vertices in $S' \cap N_l[u]$ are arbitrarily partitioned into s-stars. Again, since $S' \cap N_l[u]$ induces a clique in G, this is possible. Figure 5 depicts an example of the rearranged partition. □

A.2 Additional properties

To more closely compare the handle 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 handles$(\mathcal{P}) = (H_0, H_1, \ldots, H_{2n})$ and let x be an interval with $t := \text{end}(x)$. Then, one of the following is true:
i) $x \in H_{t-1}$, $\|H_{t-1}\| \geq 3$, and $\|H_t\| = \|H_{t-1}\| - 3$ or
ii) $x \notin H_{t-1}$ and $H_t = H_{t-1}$.
Moreover, in both cases, $x \notin H_t$.

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 connected to x, it follows that $\text{start}(c) < t$. Since x ranks first, H_{t-1} contains twice both elements x and c. Hence, $\|H_{t-1}\| \geq 4$, and from the definition of $H_t = H_{t-1} \ominus (x, x, c)$ it follows that $\|H_t\| = \|H_{t-1}\| - 3$, we are thus in case (i). Moreover, only one copy of c remains in H_t.

If $\text{rank}(x) = 2$ and x is not the center. Let c be the center of P and y be the interval of rank 1 in P. Using the reasoning above, it follows that H_{t-1} contains once c and twice x but not y, implying that $\|H_{t-1}\| \geq 3$. As $H_t = H_{t-1} \ominus (x, x, c, c, y, y)$, this implies that $\|H_t\| = \|H_{t-1}\| - 3$: 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 H_{t-1} contains x once and z twice, but not y, implying that $\|H_{t-1}\| \geq 3$. As $H_t = H_{t-1} \ominus (x, x, y, y, z, z)$, this implies that $\|H_t\| = \|H_{t-1}\| - 3$: we are in case (i).

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

The fact that $x \notin H_t$ is clear in each case (all copies are removed when $x \in H_{t-1}$, 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 $\text{handles}(\mathcal{P}) = (H_0, H_1, \ldots, H_{2n})$ and $x \in H_t$, it holds that $\text{start}(x) \leq t < \text{end}(x)$.

Proof. An element x is only added to a handle list A_t or H_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 $t := \text{end}(x)$. We show that neither A_t nor H_t contain x, which suffices to complete the proof.

The fact that $x \notin H_t$ 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 lowest interval) and all of its copies (at most two) are removed from A_{t-1} to obtain A_t in line 6 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 handle 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 $\text{end}(q_j) \leq \text{end}(q'_j)$ since $Q \preccurlyeq Q'$ and $\text{end}(q'_j) \leq \text{end}(q'_{j-1})$ since Q' is sorted. Hence, q_j is aligned with an interval 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 lower than the insertion point of x (assuming that already-present copies of x in Q or Q' are not shifted), this is equivalent to $\text{end}(q) < \text{end}(x)$. Note that if some q'_i is shifted but q_i is not, then $\text{end}(q'_i) < \text{end}(x) \leq \text{end}(q_i)$, a contradiction to $\text{end}(q'_i) \geq \text{end}(q_i)$. This implies that the insertion point of x in Q' is not higher than the insertion point 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 the insertion point is not higher in Q' than in 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 higher in Q' than in Q), hence q_i is shifted and q'_i is not. In Q we have $\mathrm{end}(q_i) < \mathrm{end}(x)$, and in Q' intervals q'_i and q' must be consecutive and cannot occur lower than x (note that $q' = x$ is possible), thus we have $\mathrm{end}(q'_i) \geq \mathrm{end}(q') \geq \mathrm{end}(x)$. Overall, we indeed have $\mathrm{end}(q) \leq \mathrm{end}(q')$. □

A.3 Proof of Lemma 1

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

For $t = 0$, Algorithm 1 defines set $A_0 = \emptyset$, and $H_0 = \emptyset \preccurlyeq A_0$. 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 $x \notin H_{t-1}$, $x \notin A_{t-1}$, $H_t = H_{t-1} \oplus (x, x)$, and Algorithm 1 defines $A_t := A_{t-1} \oplus (x, x)$. Then property $\|H_t\| - \|A_t\| \equiv 0 \pmod 3$ is trivially preserved, and Property 3 implies $H_t \preccurlyeq A_t$.

If an interval x ends at t, then we first show 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, $\|H_{t-1}\| \leq 2$ since $H_{t-1} \preccurlyeq A_{t-1}$, which implies $\|H_{t-1}\| = \|A_{t-1}\|$ (since $\|H_{t-1}\| - \|A_{t-1}\| \equiv 0 \pmod 3$). By Property 1, we must have $H_t = H_{t-1}$ and $x \notin H_{t-1}$. On the other hand, with $H_{t-1} \preccurlyeq A_{t-1}$, the lowest element x' in H_{t-1} must have $\mathrm{end}(x') \leq \mathrm{end}(x) = t$. By Property 2, $\mathrm{end}(x') > t - 1$, i.e. $\mathrm{end}(x') = \mathrm{end}(x)$, and $x' = x$: a contradiction since $x \notin H_{t-1}$.

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

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

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

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

A.4 Proof of Lemma 2

Proof. We first introduce some definitions. Given an interval x and a handle 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 $\{u \in V \mid \text{start}(u) \leq t\}$. Then $\mathcal{P}$ is called a *partial partition at* t if each V_j is either (i) a singleton $\{x\}$, in which case $\text{end}(x) > t$, (ii) an edge $\{x, y\}$, in which case $\max\{\text{end}(x), \text{end}(y)\} > t$, or (iii) a triple $\{x, y, z\}$ containing a P_3. Trivially, P_3-partitions of an interval graph one-to-one correspond to partial partitions 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 $\text{end}(x) < \text{end}(y)$ 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 $\text{start}(x) \leq t < \text{end}(x)$ (Property 2), x must be in a singleton or 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).

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 $t = \text{start}(x)$ 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 $t = \text{end}(x)$ with $x \notin A_{t-1}$. Then, in $\mathcal{P}$, either x is part of an edge $\{x, y\}$ with $\text{end}(y) > t$, 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 $t = \text{end}(x)$ with $x \in A_{t-1}$. Then, the lowest element of A_{t-1} must be x (no other interval $u \in A_{t-1}$ can have $t - 1 < \text{end}(u) \leq \text{end}(x)$). Let y and z be the two elements above x in A_{t-1}. Then, by construction, $A_t = A_{t-1} \ominus (x, y, z)$ and $\text{end}(x) \leq \text{end}(y) \leq \text{end}(z) \leq \text{end}(u)$ 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 $\text{end}(z) > t$) 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 indeed 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 $\text{start}(y) < \text{start}(x)$, 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 $\mathcal{P}' := (\mathcal{P} \setminus \{\{x, u\}, \{y, v\}, \{z\}\}) \cup \{\{y, u, v\}, \{x, z\}\}$. In the case where $\text{start}(x) < \text{start}(y)$, $\{x, u, v\}$ contains a P_3 and $\mathcal{P}' := (\mathcal{P} \setminus \{\{x, u\}, \{y, v\}, \{z\}\}) \cup \{\{x, u, v\}, \{y, z\}\}$ 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 respectively $\mathcal{P}' := (\mathcal{P} \setminus \{\{x, u\}, \{y, v\}, \{z, w\}\}) \cup \{\{y, u, v\}, \{x, z, w\}\}$ and $\mathcal{P}' := (\mathcal{P} \setminus \{\{x, u\}, \{y, v\}, \{z, w\}\}) \cup \{\{x, u, v\}, \{y, z, w\}\}$. 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. □

A.5 Proof of Theorem 2

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 2. For the "if" part, suppose that G has a P_3-partition $\mathcal{P}$. Then Lemma 1 implies that Algorithm 1 defines set 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 $\{1, \ldots, n\}$ [27, Section 8]. Hence, in $O(n)$ time, we can get an increasingly sorted list L of $2n$ event points (for example, using "pigeon sort"). Then, in $O(n)$ time, iterate over L and reassign the event points to points in $\{1, \ldots, 2n\}$ in the order of their 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 handle list can be executed in $O(\log n)$ time if it is implemented as a balanced binary tree (not 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 handle list). □

B Proofs for Section 3 (Bipartite permutation graphs)

B.1 Additional definitions and first observations

Given a vertex x and a set of vertices X', we write $x \ll X'$ if for every vertex $x' \in X'$ it holds $x \prec x'$. Given another set of vertices Y', we write $X' \ll Y'$ if for every vertex $x' \in X'$ and every vertex $y' \in Y'$ it holds that $x' \prec y'$.

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 exactly a star and there is only one way to assign the star center. Thus, we also refer to $G[X]$ as a star. We denote center(X) as the center of 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 size s of the stars:

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

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.

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 (resp. both) edge(s) in $\{\{u_0, w_0\}, \{u_1, w_1\}\}$, e crosses one (resp. 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 1. 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 with $u \prec u_0$).

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. □

B.2 Proof of Lemma 3

Proof. Let $\mathcal{P}$ be an s-star partition for G. First, we show that for any two stars $X \neq Y \in \mathcal{P}$ where X and Y are neither non-crossing nor interleaving, then X and Y are in one of the following four configurations (possibly after exchanging the roles of X and Y, see Figure 3 for an illustration):

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

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

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

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

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

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

We now prove that $\mathcal{P}$ does not contain any pair of stars $X \neq Y \in \mathcal{P}$ in Configurations I, II, III or IV. For each configuration we proceed by contradiction: assuming a pair of stars is in this configuration, we construct an s-star partition $\mathcal{P}'$ with a strictly smaller score than $\mathcal{P}$.

Configuration I. Let X, Y be two stars of $\mathcal{P}$ in Configuration I, that is, $\text{scope}(X) \cap \text{scope}(Y) \neq \emptyset$. Write $x_c = \text{center}(X)$ and $y_c = \text{center}(Y)$. Then x_c and y_c must be in the same subset of U or W. Without loss of generality, we can 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 $z_k = \text{lm}(Y)$. 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 $z_k \ll Z_r$. 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 which 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 1. With a symmetrical argument, G has an edge $\{z, x_c\}$ for all $z \in Z_l$. Thus the vertex sets $X' = Z_l \cup \{x_c\}$ and $Y' = Z_r \cup \{y_c\}$ both form a star in G.

We now compare the widths of $G[X']$ and $G[Y']$ with 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}, i.e. the cardinality of 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 $\mathcal{P}' = (\mathcal{P} \setminus \{X, Y\}) \cup \{X', Y'\}$ with $\text{width}(\mathcal{P}') < \text{width}(\mathcal{P})$, i.e. with strictly smaller score. Thus, no pair of stars in the s-star partition $\mathcal{P}$ may be in Configuration I.

Configuration II. Let X, Y be two stars of $\mathcal{P}$ in Configuration II, that is, $\text{center}(Y) \in \text{scope}(X)$ 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 $x_c \ll \text{scope}(Y)$ or $\text{scope}(Y) \ll x_c$. We only consider the case of $x_c \ll \text{scope}(Y)$. The case of $\text{scope}(Y) \ll x_c$ works analogously.

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

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

Since $y_c \prec v$, $\text{width}(X') \leq \text{width}(X) - 1$. 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 $\text{width}(\mathcal{P}') < \text{width}(\mathcal{P})$, i.e. with strictly smaller score. Thus, 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 $x_c := \text{center}(X)$ and $y_c := \text{center}(Y)$ and assume without loss of generality that $x_c \prec y_c$. Then, $\text{scope}(Y) \ll \text{scope}(X)$. Thus, all edges of $G[X]$ cross with 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 (resp. B_Y, $B_{X'}$, $B_{Y'}$) for the *branches* of the corresponding star, that is, for the set of edges of $G[X]$ (resp. $G[Y]$, $G[X']$, $G[Y']$), and $R_{X,Y}$ (resp. $R_{X',Y'}$) for the set of edges in $G[Z]$ for any $Z \in \mathcal{P} \setminus \{X, Y\}$ (resp. $Z \in \mathcal{P}' \setminus \{X', Y'\}$). Note that by definition of $\mathcal{P}'$, $R_{X',Y'} = R_{X,Y}$, we thus simply denote this set by R. We write $\times_{b,b}$ (resp. $\times_{b',b'}$) for the number of crossings between branches of B_X and B_Y (resp. $B_{X'}$ and $B_{Y'}$), $\times_{b,r}$ (resp. $\times_{b',r}$) for the number of crossings between a branch in $B_X \cup B_Y$ (resp. $B_{X'} \cup B_{Y'}$) and an edge in R, and $\times_{r,r}$ for the number of crossings between two edges in R. Note that $\#\text{edge-crossings}(\mathcal{P}) = \times_{b,b} + \times_{b,r} + \times_{r,r}$ and that $\#\text{edge-crossings}(\mathcal{P}') = \times_{b',b'} + \times_{b',r} + \times_{r,r}$.

It is easy to see that $\times_{b',b'} = 0$ (X' and Y' form non-crossing stars), and $\times_{b,b} > 0$ (actually, $\times_{b,b} = s^2$). Let x_i (resp. y_i) be the i-th leaf of X (resp. Y) by 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

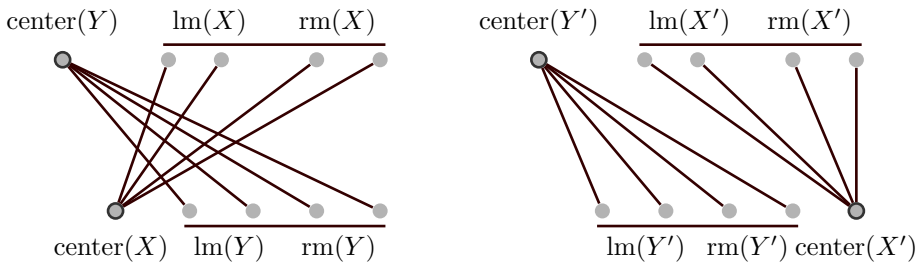


Figure 6: Left: Two stars X and Y in Configuration IV, where $\mathrm{d}(X, Y)$ is minimal. Right: Two stars X' and Y' obtained from X and Y, with equal width and fewer crossings.

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 $\mathrm{center}(X) \ll \mathrm{scope}(Y)$ and $\mathrm{center}(Y) \ll \mathrm{scope}(X)$. We moreover assume that X and Y are chosen so that the number of elements between $\mathrm{center}(X)$ and $\mathrm{rm}(Y)$, 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 $\mathrm{center}(X)$ and $\mathrm{rm}(Y)$, except for the $s - 1$ other leaves of Y. Suppose towards a contradiction that there is a vertex $z \notin \{\mathrm{center}(X)\} \cup \mathrm{scope}(Y)$ such that $\mathrm{center}(X) \prec z \prec \mathrm{rm}(Y)$.

Assume first that z is the center of a star $G[X']$ with $X' \in \mathcal{P}$. Thus, $\mathrm{scope}(X) \ll \mathrm{scope}(X')$ since X and X' cannot be in Configuration I or III. Also, we have $\mathrm{center}(X) \prec z \ll \mathrm{scope}(Y)$ 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 $\mathrm{center}(X') \ll \mathrm{scope}(Y)$, $\mathrm{center}(Y) \ll \mathrm{scope}(X')$), 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': $\mathrm{scope}(Y') \cap \mathrm{scope}(Y) = \emptyset$ as otherwise Y and Y' would be in Configuration I. Using $z \prec \mathrm{rm}(Y)$, it follows that $\mathrm{scope}(Y') \ll \mathrm{scope}(Y)$. This implies that $\mathrm{center}(Y') \prec \mathrm{center}(Y) \ll \mathrm{scope}(X)$ as otherwise Y' and Y would be in Configuration III. We now compare X and Y'. By the above, we have $\mathrm{center}(Y') \ll \mathrm{scope}(X)$. Also, $\mathrm{center}(X) \notin \mathrm{scope}(Y')$ as otherwise Y' and X would be in Configuration II. Using $\mathrm{center}(X) \prec z$, we thus have $\mathrm{center}(X) \ll$

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.

Overall, no other vertex than the leaves of Y may exist between center(X) and rm(Y).

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 edge $\{\mathrm{center}(Y), \mathrm{rm}(Y)\}$ and therefore implies by definition that G contains the edge $\{\mathrm{rm}(Y), u\}$. Thus, $X' = X_0 \cup \{\mathrm{rm}(Y)\}$ forms a star, see Figure 6 (right). Also, X' and Y' are non-crossing (X' is completely to the right of Y').

We now compare their widths with the original stars $G[X]$ and $G[Y]$. Obviously, $\mathrm{width}(X') = \mathrm{width}(X)$. 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 width did not change, we need to show that #edge-crossings($\mathcal{P}'$) < #edge-crossings($\mathcal{P}$). We introduce the same notations as in Configuration III: Let B_X (resp. B_Y, $B_{X'}$, $B_{Y'}$) be the set of branches of the corresponding star, that is, the set of edges in $G[X]$ (resp. $G[Y]$, $G[X']$, $G[Y']$), and let $R_{X,Y}$ (resp. $R_{X',Y'}$) be the set of edges in $G[Z]$ for any $Z \in \mathcal{P} \setminus \{X, Y\}$ (resp. $Z \in \mathcal{P}' \setminus \{X', Y'\}$). Note that by definition of $\mathcal{P}'$, $R_{X',Y'} = R_{X,Y}$, and we thus simply denote this set by R. Further, let $\times_{b,b}$ (resp. $\times_{b',b'}$) be the number of crossings between branches of B_X and B_Y (resp. $B_{X'}$ and $B_{Y'}$), let $\times_{b,r}$ (resp. $\times_{b',r}$) be the number of crossings between a branch in $B_X \cup B_Y$ (resp. $B_{X'} \cup B_{Y'}$) and an edge in R, and let $\times_{r,r}$ be the number of crossings between two edges in R. Note that #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 $\times_{b,b} > 0$ (actually, $\times_{b,b} = s^2$).

We now show that $\times_{b',r} \leq \times_{b,r}$. First, remember that no edge in R has an endpoint between center(X) and rm(Y). We consider the branches in $B_{X'} \cup B_{Y'}$ and give for each 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 $\{\mathrm{center}(X), \mathrm{center}(Y)\}$ must also cross $\{\mathrm{rm}(Y), \mathrm{center}(Y)\}$. For any other branch $b = \{\mathrm{center}(Y), y\}$ 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 #edge-crossings($\mathcal{P}'$) < #edge-crossings($\mathcal{P}$).

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

B.3 Proof of Corollary 1

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

Any star interleaving with $G[X]$ must contain 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 as proven before. □

B.4 Proof of Theorem 3

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 $u_i \prec u_j$ (resp. $w_i \prec w_j$) 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 with any other star from $\mathcal{P}$ (see Lemma 3 and Corollary 1). In both cases, the part of the graph that lies consecutively left to the star (resp. the two interleaving stars) with respect to the strong ordering must have an s-star partition by its own. This is clearly also true for the part of the graph that lies consecutively 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 follows:

$$T(x, y) = (G[\{u_1, u_2, \ldots, u_{x+s\cdot y}, w_1, w_2, \ldots, w_{y+s\cdot x}\}], s) \in \text{Star Partition}).$$

Initialize the table T by:

$$\begin{aligned}
&T(0, 1) = (G[\{u_1, u_2, \ldots, u_s, w_1\}] \text{ contains an } s\text{-star}),\\
&T(1, 0) = (G[\{u_1, w_1, w_2, \ldots, w_s\}] \text{ contains an } s\text{-star}), \text{ and}\\
&T(1, 1) = (G[\{u_1, u_2, \ldots, u_{s+1}, w_1, w_2, \ldots, w_{s+1}\}] \text{ contains disjoint } s\text{-stars}).
\end{aligned}$$

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

$$\begin{aligned}
T(x, y) = &\big((G[\{u_{x+s\cdot(y-1)+1}, u_{x+s\cdot(y-1)+2}, \ldots, u_{x+s\cdot(y-1)+s}, w_{(y-1)+s\cdot x+1}\}]\\
&\text{contains an } s\text{-star}) \wedge T(x, y-1)\big) \qquad \text{(a)}\\
&\vee \big((G[\{u_{(x-1)+s\cdot y+1}, w_{y+s\cdot(x-1)+1}, w_{y+s\cdot(x-1)+2}, \ldots, w_{y+s\cdot(x-1)+s}\}])\\
&\text{contains an } s\text{-star}) \wedge T(x-1, y)\big) \qquad \text{(b)}\\
&\vee \big((G[\{u_{(x-1)+s\cdot(y-1)+1}, u_{(x-1)+s\cdot(y-1)+2}, \ldots, u_{(x-1)+s\cdot(y-1)+s+1},\\
&w_{(y-1)+s\cdot(x-1)+1}, w_{(y-1)+s\cdot(x-1)+2}, \ldots, w_{(y-1)+s\cdot(x-1)+s+1}, \}]\\
&\text{contains an } s\text{-star}) \wedge T(x-1, y-1)\big). \qquad \text{(c)}
\end{aligned}$$

Finally, $T(k_U, k_V)$ is true if and only if there is an s-star partition with minimum score, and hence, if and only if there is an s-star partition.

Concerning the running time, first, a strong ordering of the vertices can be computed in linear time [34]. Second, the table in the dynamic program have

$O(k^2)$ table 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, 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, $G[\{u_1, u_2, \ldots, u_{x-1+s\cdot y}, w_1, w_2, \ldots, w_{y+s\cdot(x-1)}\}]$ 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 $G[\{u_1, u_2, \ldots, u_{x-1+s\cdot(y-1)}, w_1, w_2, \ldots, w_{y-1+s\cdot(x-1}\}]$ has an s-star partition.

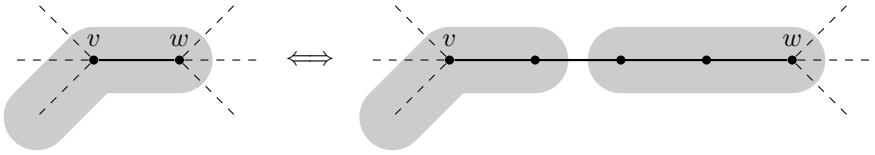
Note that the rightmost vertex of G' in U can only be a leaf of a non-crossing star in $\mathcal{P}'$ if the rightmost vertex of G' in W is the center and vice versa. Otherwise, the corresponding star is clearly not non-crossing. Hence, these cases are already covered by (a) and (b). Furthermore, neither 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. □

C Proofs for Section 4 (Grid graphs)

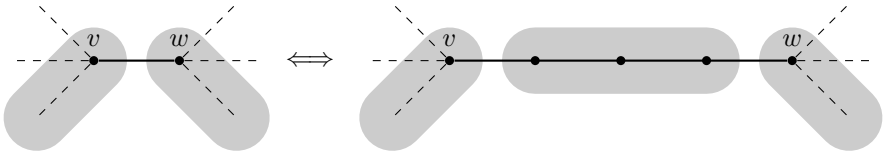
First, note that the correctness of Observation 1 is proven by Figure 7, which enumerates all possible cases.

Now, for the correctness of Theorem 4, it remains to show that, for a graph G' obtained using Construction 1 from a planar graph G of maximum degree three, G' has a P_3-partition if and only if G has. Then, Theorem 4 follows from the NP-hardness of P_3-Partition on planar graphs of maximum degree three.

Proof of correctness of Construction 1. It remains to show that G has a P_3-partition if and only if G' has. By Observation 1, 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 4c. 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.



(a) Case 1: Exactly one of v or w is an end point of a P_3 (the cases are symmetric).



(b) Case 2: Both of v or w are end points of a P_3.

Figure 7: All possibilities of two vertices v and w participating in a P_3-partitioning 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.

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. □

D Proofs for Section 5 (Further results)

D.1 Proof of Theorem 5 (Cographs)

Definition 3. A cotree $\mathrm{cot}(G)$ of a cograph $G = (V, E)$ is a rooted binary tree $T = (V_T, E_T, r), r \in V_T$, together with a label function $l : (V_T \setminus V_L) \to \{\oplus, \otimes\}$ that assigns a binary label to each non-leaf vertex of T and the set of leaves V_L 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 $\mathrm{cot}(G)$ corresponds to G.

Proof of Theorem 5. Let $(G = (V, E), s)$ be a Star Partition instance with G being a cograph. Let $T = (V_T, E_T, r) = \mathrm{cot}(G)$ 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 by x.

We define the 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 $\ell_i, i \in \{1, 2\}$, 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 $T[x]$.) 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: $(c_1 \cdot s > \ell_2) \wedge (c_2 \cdot s > \ell_1)$**.** 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: $(c_1 \cdot s \leq \ell_2) \wedge (c_2 \cdot s \leq \ell_1)$**.** In this case, we can cover $c \cdot s$ leaves in $T[x]$ by covering $c_1 \cdot s$ leaves from $T[x_2]$ by centers from $T[x_1]$ and $c_2 \cdot s$ leaves from $T[x_1]$ by centers from $T[x_2]$. This is obviously the best one can do. Thus, $A[c_1, c_2] = c \cdot s$.

Case 3: $(c_1 \cdot s > \ell_2) \wedge (c_2 \cdot s \leq \ell_1)$ **or** $(c_1 \cdot s \leq \ell_2) \wedge (c_2 \cdot s > \ell_1)$**.** To see that also in this case greedily maximizing the number of leaves from $T[x_2]$ that are covered by centers from $T[x_1]$ and vice versa is optimal, 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.

Without loss of generality, let $y_1 > 0$ and $y_2 = 0$ (if both become zero, we would be in Case 2). Note that this corresponds to the case $(c_1 \cdot s > \ell_2) \wedge (c_2 \cdot s \leq \ell_1)$—the other case works analogously. As $y_2 = 0$ and $c_2 \cdot s \leq \ell_1$, we can assume that c_2 centers from $T[x_2]$ cover altogether $c_2 \cdot s$ leaves from $T[x_1]$. Furthermore, all ℓ_2 leaves from $T[x_2]$ are covered by centers from $T[x_1]$. Since $(c_1 \cdot s > \ell_2)$, the centers from $T[x_1]$ might additionally cover some number ℓ' of leaves from $T[x_1]$, but clearly:

- ℓ' is at most $c_1 \cdot s - \ell_2$, the maximum number of leaves that can be covered by c_1 centers,
- ℓ' is at most $\ell_1 - c_2 \cdot s$, 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 $T[x_1]$. 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.)

Thus, $A[c_1, c_2] =$

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

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 analyzed as follows: Computing the cotree representation runs in linear time [26]. 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 $O(n)$. □

D.2 Proof of Theorem 6 (Split graphs)

D.2.1 First part: proof of the NP-completeness for $s \geq 3$

Proof of Theorem 6, NP-hardness part. 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 [31] (illustrated in Figure 8).

Exact Cover by s-Sets
Input: A finite set U and a collection $\mathcal{S}$ of subsets of U of size s.
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.

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. □

D.2.2 Second part: polynomial-time solvability for $s = 2$.

Let $G = (C \cup I, E_C \cup E)$ be a split graph where (C, E_C) is an induced clique, I is 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 has trivially no P_3-partition. We thus assume that $|C| + |I|$ (and hence $2|C| - |I|$) is a multiple of 3, and that $2|C| - |I| \geq 0$.

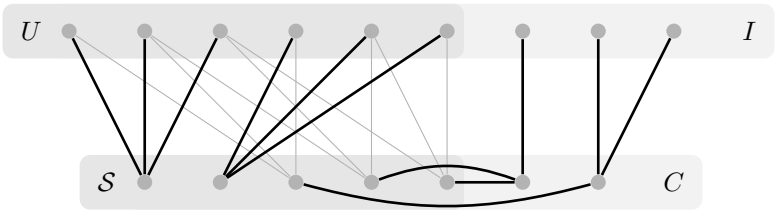


Figure 8: Reduction from Exact Cover by s-Sets to Star Partition. Dummy vertices are in the light area and non-dummy vertices are in the dark area. The solution for the constructed graph G is indicated by thick edges.

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.

Thus, for each P_3, we can assume that the edges are selected so that exactly one edge is incident to the independent set vertex. Due to this observation, any P_3-partition can be edited until each independent set vertex is incident with exactly one edge in P.

Definition 4. A factor F of the bipartite graph B is *feasible* if, in F, all independent set vertices have degree 1, every clique vertex has degree 0, 1 or 2, and there are at least as many degree-zero clique vertices as degree-one clique vertices.

Property 5. In any feasible factor F of B, the difference between the number of degree-zero clique vertices and degree-one clique vertices is a multiple of 3.

Proof. Write n_0 (resp. n_1, n_2) for the number of degree-zero (resp. degree-1, degree-2) clique vertices in F. We have $|C| = n_0+n_1+n_2$ and $|I| = 2n_2+n_1$ (each vertex in I has degree one). Then $n_1 - n_0 = n_1 + 2n_0 - 3n_0 = 2|C| - |I| - 3n_0 = -(|C| + |I|) + 3(|C| - n_0)$, which is a multiple of 3 since the number of vertices in G is a multiple of 3 itself. □

Lemma 4. *G admits a partition into P_3s if and only if there exists a feasible factor of 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 $F := (C \cup I, P_E)$ be the corresponding factor of B.

Each 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 vertex v in C belongs to a P_3 from P. Depending on the type of this P_3, v can have degree zero (Type (i) or (ii)), one (Type (ii)) or two (Type (iii)). Let $n_{(i)}$ (resp. $n_{(ii)}$) denote the number of P_3s of Type (i) (resp. (ii)). Then, in C, the number of degree-zero vertices is $n_{(ii)} + 3n_{(i)}$, and the number of degree-one vertices is $n_{(ii)}$, 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-one vertex v in C, add $\{v, u, w\}$ to P where u is its neighbor in I and w 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. Overall, due to the degree constraints, P is a P_3-partition of G. □

With the above lemma, we can reduce our P_3-partition problem to a special graph factor problem: We add an additional vertex z to B, connect it to all vertices in C. We call this graph B'. Now, the following lemma states without proof that B' can be used to find a feasible factor for B.

Lemma 5. *The bipartite graph B admits a feasible factor if and only if graph B' has a factor satisfying the following degree constraints: Every vertex in I has degree one, every vertex in C has degree zero or two, and vertex z has degree at most $(2|C| - |I|)/3$.*

Figure 9 depicts an example factor for the graph B' fulfilling the degree constraints of Lemma 5. This graph factor problem can be solved in polynomial time [28].

Using a gadget introduced by Cornuéjols [28], we can even reduce our problem to a perfect matching problem. To this end, we will prove an even stronger result than Lemma 5. We introduce integers q and r such that $(2|C| - |I|)/3 = 2q + r$ (with $q \geq 0$ and $r \in \{0, 1\}$). We start with the following property about the number of degree-one vertices that are actually possible.

Property 6. Let $F = (C \cup I, P \cap E)$ be a feasible factor of B. Then, the number of degree-one vertices in C is $2i + r$ for some $0 \leq i \leq q$.

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), and $n_0 - n_1$ is a positive multiple of 3 (by Property 5). Let $n_0 - n_1 = 3j$ with $j \in \mathbb{N}$. This can be rewritten as $2n_1 = 2n_0 - 6j$ and $3n_1 = 2n_0 + n_1 - 6j$, which implies, using the above equations, that $n_1 = (2|C| - |I|)/3 - 2j$. Thus, $n_1 = 2(q - j) + r$, which completes the proof of the property. □

In accordance with Lemma 5, we get the following.

Lemma 6. *The bipartite graph B admits a feasible factor if and only if graph B' has a factor satisfying the following degree constraints: Every vertex in I has*

degree one, every vertex in C has degree zero or two, and vertex z has degree at most $(2|C| - |I|)/3$. Moreover, in such a case, the degree of z is $2i + r$ with $0 \leq i \leq q$.

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 6, 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 have degree one, every clique vertex have degree zero or two, and vertex z has degree at most $(2|C| - |I|)/3$. 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 at most the original degree of z, that is, $n_1 \leq (2|C| - |I|)/3$.

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-1 and degree-0 vertices is $n_1 - n_0 = (3n_0 - 2|C| - |I|)/2$, which is positive. □

Now we construct a graph B^* from B' for 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. Also, we have to replace vertex z by a gadget to simulate its degree constraint.

Construction 2. Let $m = |E|$ (number of edges between C and I in the original split graph) and d_v be the degree of a clique vertex v in B'. Note that due to the edges going to vertex z, we have $\sum_{v \in C} d_v = m + |C|$. Moreover, we can safely assume that $|I| = O(m)$ and $|C| = O(m)$.

For each independent set vertex $u \in I$, add to B^* a vertex u^I. We now consider each clique vertex v. For each edge e in B' which contains v add a vertex v_e to B^*, and denote the set of these vertices as V_v. Add d_v vertices $v'_1, v'_2, \ldots, v'_{d_v}$ to B^*, and denote the set of these vertices as V'_v. Then $|V_v| = |V'_v| = d_v$.

Consider now vertex z. For each edge $\{z, v\}$ in B', add a vertex z_v to B^*, and denote the set of these vertices as V_z. Finally, add vertices $z'_1, z'_2, \ldots, z'_{|C|-q}$ to B^* and denote the set of these vertices as V'_z. Thus, $|V_z| = |C|$ and $|V'_z| = |C| - q$. In total, B^* has $|I| + 2\sum_{v \in C} d_v + |C| + |C| - a = 2m + 4|C| + |I| - a = O(m)$ vertices.

Now we are ready to add edges to B^*. Consider first each edge $e = \{u, v\}$ (resp. $e = \{z, v\}$) in B' which connects an independent set vertex u (resp. vertex z) and a clique vertex v. Add edge $\{u^I, v_e\}$ (resp. $\{z_e, v_e\}$) to B^*, and edges $\{v_e, v'_i\}, i \in \{1, \ldots, d_v\}$. Now, for each edge $e = \{z, v\}$ in B' and each $i \in \{1, \ldots, |C| - r\}$, add edge $\{z_e, z'_i\}$ to B^*. Finally, for each $v \in C$ add edge $\{v'_1, v'_2\}$, and for each $i \in \{1, \ldots, q\}$, add edge $\{z'_{2i-1}, z'_{2i}\}$ to B^*.

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

Now, we show how the constructed graph B^* can be used to find a factor for B' satisfying the specific degree constraint as stated in Lemma 6.

Lemma 7. *Graph B' admits a factor F' satisfying the condition that every vertex in I has degree one, every vertex in C has degree zero or two, and vertex z has degree $2i+r$, for some $i \in \{0, 1, \ldots, q\}$, if and only if graph B^* has a perfect matching.*

Proof. Assume that B' admits a factor F' satisfying the above condition. We show that the following construction yields a perfect matching M for B^*.

For each edge $e = \{u, v\}$ in F' that connects an independent set vertex u and a clique vertex v, add to M edge $\{u^I, v_e\}$. For each edge $e = \{z, v\}$ in F' that connects vertex z with a clique vertex v, add to M edge $\{z_e, v_e\}$.

For each vertex $v \in C \cup \{z\}$, let $R_v \subseteq V_v$ be the set of vertices which are not yet matched by M. We now need to cover all vertices in R_v or V'_v for any $v \in C \cup \{z\}$. Consider first a clique vertex v. Depending on whether v has degree zero or two in F', $|V_v| - |R_v|$ is either zero or two, and $|V'_v| - |R_v| = |V_v| - |R_v|$. Moreover, all edges between vertices V_v and V'_v are B^*. If $|R_v| = |V'_v|$, then add edges to match vertices of R_v and of V'_v; otherwise, $|R_v| = |V'_v| - 2$: add edge $\{v'_1, v'_2\}$ to M, and add edges to match vertices of R_v and of $V'_v \setminus \{v'_1, v'_2\}$. Analogously, let $R_z \subseteq V_z$ be the set of vertices in V_z which are not yet matched by M. Note that depending on the degree of v in F, $|V_z| - |R_z|$ is $2i+r$ where $i \in \{0, 1, \ldots, q\}$. Since $|V'_z| = |V_z| - r$, we have $|V'_z| - |R_z| = 2i$. Moreover, all edges between vertices V_z and V'_z are in B^*. Add edges $\{v_1, v_2\}, \{v_3, v_4\}, \ldots, \{v_{2i-1}, v_{2i}\}$ to M, and add edges to match vertices of R_z and of $V'_z \setminus \{v_1, v_2, \ldots, v_{2i}\}$. It is easy to verify that M is a perfect matching.

Conversely, 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 $\{u^I, v_e\} \in M$ which connects u^I (where u is an independent set vertex) with a vertex v_e corresponding to the clique vertex $v \in C$, we add to P'_E edge $\{u, v\}$. For each edge $\{z_e, v_e\} \in M$ which connects vertices in V_z and $V_v, v \in C$, add to P'_E edge $\{z, v\}$.

We show that F' is a factor for B^* with the property that each independent set vertex $u \in I$ has degree one, each clique vertex $v \in C$ has degree zero or two, and vertex z has degree $2i + r$, where $i \in \{1, \ldots, b\}$. Obviously, every independent set vertex $u \in I$ has degree one. Consider a clique vertex $v \in C$. By the construction of graph B^*, in order to match all vertices in V'_v, either (i) every vertex in V'_v has to be matched to a vertex in V_v or (ii) v_1 and v_2 are matched together while every vertex in $V'_v \setminus \{v_1, v_2\}$ is matched to exactly one vertex in V_v. This implies that either no vertex or exactly two vertices in V_v are matched to some vertices which are not from V'_v. 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 V'_z which has size $|C| - r$, exactly i pairs of vertices in V'_z can be left without being matched to any vertex in V_z where $0 \leq i \leq b$. This implies that exactly $|C| - (|C| - r - 2i) = 2i + r$ vertices from V_z are matched to vertices that are not from V'_z. Thus, z has degree $2i + r$. □

We now have gathered all ingredients to show Theorem 6.

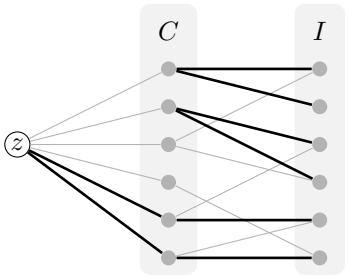


Figure 9: An example of a factor for B' fulfilling the degree constraints as required in Lemma 5.

Proof of Theorem 6, polynomial-time part. 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 created from B' by Construction 2. By Lemmas 4 and 7 and Property 6, G admits a P_3-partition if and only if B^* has a perfect matching. Since deciding whether a graph with p vertices and q edges has a perfect matching can be done in $O(q\sqrt{p})$ time using the algorithm of Hopcroft and Karp [33, 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.

□

D.3 Proof of Theorem 7 (Intractability on chordal graphs)

We reduce the following problem to P_3-Partition on chordal graphs.

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 [30] introduced Construction 3 described below (see also Figure 10). Using it as a reduction from the NP-hard restriction of 3DM to planar graphs, they proved that P_3-Partition restricted to bipartite planar graphs is NP-complete [29].

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 $\{u_a, u'_a\}$. Denote u_a as an *element-vertex* and u'_a as a *pendant-vertex*. For each triple $t = (r, b, y) \in T$ create three vertices v^t_r, v^t_b, v^t_y. Denote these three vertices as *triple-vertices*. Make triple-vertex v^t_b adjacent to both v^t_r and v^t_y. Also make triple-vertex v^t_r (resp. v^t_b and v^t_y) adjacent to element-vertex u_r (resp. u_b and u_y). Formally, $V = \{u_a, u'_a \mid a \in R \cup B \cup Y\} \cup \{v^t_r, v^t_b, v^t_y \mid t = (r, b, y) \in T\}$ and

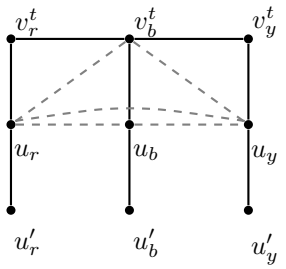


Figure 10: Gadget for a triple $t = (r, b, y) \in T$ based on Construction 3 (solid edges). The reduction in Theorem 7 introduces dashed edges.

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

Proof of Theorem 7. 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 10. 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 (resp. v_y^t) for some $t = (r, b, y) \in T$, then C contains both its neighbors: v_b^t and u_r (resp. v_b^t and u_y) which are connected, i.e. 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 v_b^t lies, in the cycle, 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: again, 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 is a perfect 3-dimensional matching for $(R, B, Y, T \subseteq R \times B \times Y)$. Then, the P_3s in $\bigcup_{t=(r,b,y) \notin M} \{\{v_r^t, v_b^t, v_y^t\}\}$ and those in $\bigcup_{t=(r,b,y) \in M} \{\{u'_r, u_r, v_r^t\}, \{u'_b, u_b, v_b^t\}, \{u'_y, u_y, v_y^t\}\}$ 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 $t \in T$: we

call such a P_3 a *triple-P_3 corresponding to t*.

Consider now any triple $t \in T$. If there exists a triple-P_3 corresponding to t (i.e., centered on v_b^t), then the two leaves can only be v_r^t and v_y^t (since u_b^t is a center). 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, or u_y. There is actually only one way to match the three triple-vertices to these three element-vertices, that is by using 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 any u_a are u_a' and v_a^t for some triple t containing a.

We 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 the triple-vertex that together with u_a and u_a' forms an element-P_3. Thus, $t \in M$. Further, let t' be another triple in M that matches a. Since t' does not correspond to any P_3 in P, there is an element-P_3 containing a triple-vertex $v_a^{t'}$ such that t also contains element a. But then $v_a^{t'}$ together with u_a and u_a' must also form an element-P_3–a contradiction. □

Bibliography of the appendix

[25] R. van Bevern, C. Komusiewicz, H. Moser, and R. Niedermeier. Measuring indifference: Unit Interval Vertex Deletion. In *Proc. 36th WG*, volume 6410 of *LNCS*, pages 232–243, 2010.

[26] D. Corneil, Y. Perl, and L. Stewart. A linear recognition algorithm for cographs. *SIAM J. Comput.*, 14(4):926–934, 1985.

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

[28] G. Cornuéjols. General factors of graphs. *J. Combin. Theory Ser. B*, 45(2): 185–198, 1988.

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

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

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

[32] B. S. Panda and S. K. Das. A linear time recognition algorithm for proper interval graphs. *Inf. Process. Lett.*, 87(3):153–161, 2003.

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

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