Star Partitions of Perfect Graphs

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

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 cases, for example, on interval graphs and bipartite permutation graphs, and also NP-hard cases, for example, on grid graphs and chordal graphs.

1 Introduction

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

Star Partition
Input: An undirected n-vertex graph $G = (V, E)$ and an integer $s \in \mathbb{N}$.
Question: Can the vertex set V be partitioned into $k := \lceil n/(s+1) \rceil$ 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 [13].

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 a few potentially challenging cases open; see Figure 1 for an overview of our results.

* 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.

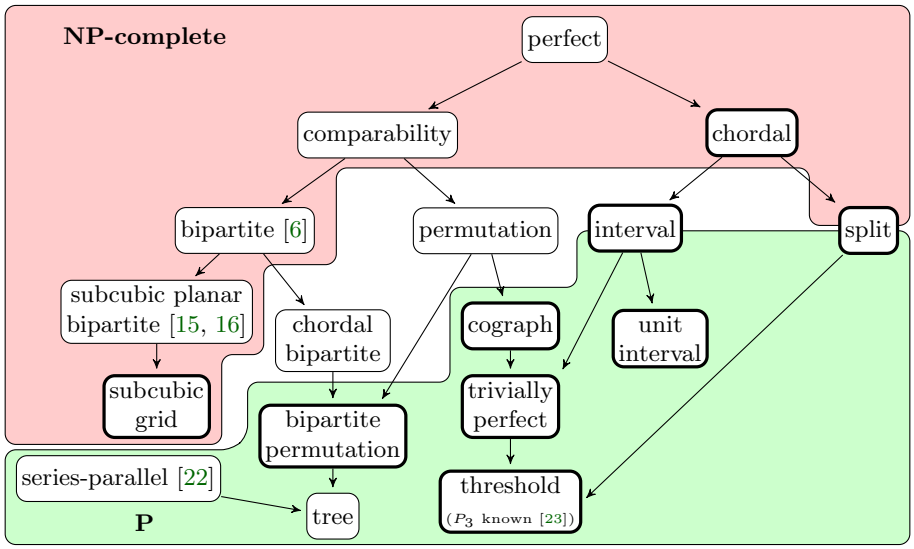


Fig. 1: Complexity classification of Star Partition. Bold borders indicate results of this paper. An arrow from a class A to a class B indicates that A contains B. In most classes, NP-completeness results hold for $s = 2$ (that is, for P_3-Partition). However, on split graphs, Star Partition is polynomial-time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

Motivation. The literature in algorithmic graph theory is full of 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 [14] as well as in the Test Cover problem arising in bioinformatics [10]. In particular, the application in distributed systems explicitly motivates the consideration of very restricted (perfect) graph classes such as grid-like structures. Star Partition on grid graphs naturally occurs in political redistricting problems [4]. We show that Star Partition remains NP-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 he or she can act as an information hub. Forming such teams is nothing else than solving Star Partition on interval graphs. We present efficient algorithms for Star Partition on unit interval graphs (that is, for the case when all agents are active for the same amount of time) and for P_3-Partition on general interval graphs.

Previous work. Packing and partitioning problems are central problems in algorithmic graph theory with many applications and with close connections to matching theory [25]. In the case of packing, one wants to maximize the number of graph vertices that are "covered" by vertex-disjoint copies of some fixed pattern graph H. In the case of partitioning, one wants to cover *all* vertices in the graph. We focus on the partitioning problem, which is also called H-Factor in the literature. In this work, we always refer to it as H-Partition. As Kirkpatrick and Hell [13] 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 [22] and on graphs of maximum degree two [16]. On the downside, P_k-Partition (for fixed $k \geq 3$) remains NP-complete on planar bipartite graphs [11]; this hardness result generalizes to H-Partition on planar graphs for any outerplanar pattern H with at least three vertices [2]. For every $s \geq 2$, Star Partition is NP-hard on bipartite graphs [6]. Partitioning into triangles, that is, K_3-Partition, is polynomial-time solvable on chordal graphs [9] and linear-time solvable on graphs of maximum degree three [17].

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 [20] and chordal bipartite graphs [21]. 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 [24], threshold graphs, cographs (for fixed k) [20] and bipartite permutation graphs [21].

Our contributions. So far, surprisingly little is 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, which is Star Partition with $s = 2$, 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 of NP-hardness, we strengthen a result of Małafiejski and Żyliński [15] and Monnot and Toulouse [16] by showing that P_3-Partition 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 [17]. Furthermore, we show P_3-Partition to be NP-complete on chordal graphs, while K_3-Partition is known to be polynomial-time solvable in

this case [9]. 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 are deferred to a full version [3].

Preliminaries. We assume basic familiarity with standard graph classes [5, 12]. Definitions of the graph classes are provided when first studied in this paper. We call the graph $K_{1,s}$ an *s-star*. For a graph $G = (V, E)$, an *s-star partition* is a set of $k := |V|/(s+1)$ pairwise disjoint vertex subsets $V_1, V_2, \ldots, V_k \subseteq V$ with $\bigcup_{1 \leq i \leq k} V_i = V$ such that each 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 with $s = 2$). Without loss of generality, we assume throughout the paper that the input graph G is connected (otherwise, we can solve the partition problem separately for each connected component of G). 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 are open and 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 delete 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 not come as a surprise that Star Partition can be solved efficiently on unit interval graphs using a greedy strategy, this is far from obvious for general interval graphs. 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

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`.

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

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 *tokens*. We require that the link between a leaf and a center *consumes* both of the leaf's tokens (such that a leaf can have only one link) and one token of the center (which can thus be linked to two leaves).

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

The algorithm works according to the following two rules: when an interval starts, its two tokens are added to a list; when an interval with remaining tokens ends, then three tokens are deleted from this list. Tokens are only picked from the earliest-ending intervals (this choice may not directly translate into a "sane" solution, with each link using tokens from only two intervals, but it turns out not to be problem). The algorithm is sketched in Algorithm 1. Figure 2 shows an example instance and the list of tokens maintained by the algorithm. Note that a token of an interval x is simply represented by a copy of interval x itself. We now introduce the necessary formal definitions.

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.

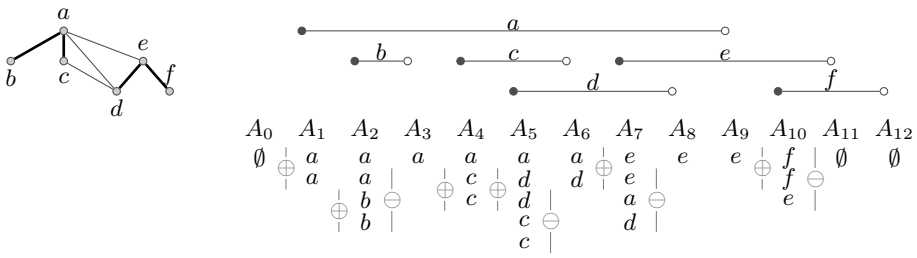


Fig. 2: Left: Interval graph with six vertices and a P_3-partition $\mathcal{P}$ (bold). Right: Interval representation of this graph and successive token lists $A_0, \ldots, A_{12}$ computed by Algorithm 1 (additions and deletions are marked with $\oplus$ and $\ominus$).

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

A *token list* Q is a list of intervals $(q_1, \ldots, q_k)$ sorted in decreasing order of their end points ($\text{end}(q_i) \geq \text{end}(q_j)$ for $1 \leq i \leq j \leq k$). We consider the list to be represented vertically, with the latest-ending interval on top (to distinguish from the left-to-right sequence of event points). We write $\|Q\|$ for the length of Q, $\emptyset$ for the empty token list, and $x \in Q$ if interval x appears in Q. We now define insertion, deletion and comparison of token lists: $Q \oplus (x_1, \ldots, x_l)$ is the token 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 tokens($\mathcal{P}$) as a tuple of $2n + 1$ token lists $(T_0, T_1, \ldots, T_{2n})$ such that $T_0 := \emptyset$ and for $t > 0$,

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

Note that in Figure 2, each token list T_t for $\mathcal{P}$ is equal to the respective A_t, except for $T_6 = (d, d)$ and $T_7 = (e, e, d, d)$.

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

Lemma 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 $T_t \preccurlyeq A_t$ and $\|T_t\| - \|A_t\| \equiv 0 \pmod 3$, where* $\text{tokens}(\mathcal{P}) = (T_0, T_1, \ldots, T_{2n})$.

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 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 [15] and Monnot and Toulouse [16], who showed that P_3-Partition is NP-complete on planar bipartite graphs of maximum degree three.

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

To show NP-hardness of P_3-Partition on grid graphs, we exploit the above mentioned result of Małafiejski and Żyliński [15] and Monnot and Toulouse [16] 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 3. *P_3-Partition is NP-hard on grid graphs of maximum degree three.*

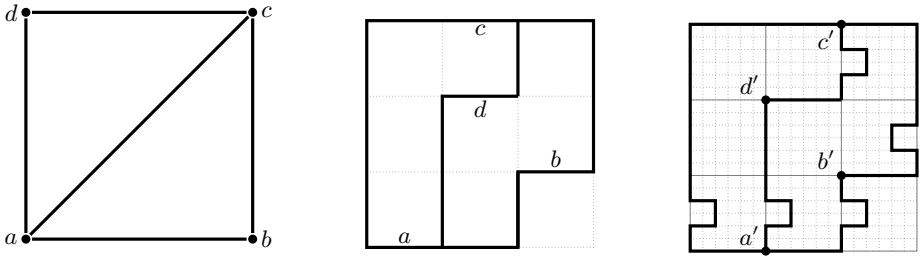
The following observation helps us embed planar graphs into grid graphs, as it allows us to replace edges by paths on $3i$ new vertices for any $i \in \mathbb{N}$.

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 3 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 [18] we obtain a crossing-free *rectilinear embedding* of G into the plane such that:

1. Each vertex is represented by a horizontal line.
2. Each edge is represented by a vertical line.
3. All lines end at integer coordinates with integers in $O(n)$.
4. If two vertices are joined by an edge, then the vertical line representing this edge ends on the horizontal lines representing the vertices.



(a) A planar graph G.

(b) A rectilinear embedding of G.

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

Fig. 3: Various embeddings of a planar graph. In the rectilinear embedding in Figure 3b, horizontal lines represent vertices of G, while vertical lines represent its edges. In Figure 3c, every intersection of a line with a grid point is a vertex, but only the vertices corresponding to vertices in Figure 3a are shown.

Figure 3b 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 3c).
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 vertices in G', except for one point that we bypass by adding a bend of five vertices to the vertical line (see Figure 3c).
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 $P(v)$.

4 Bipartite permutation graphs

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

Definition 1 (Spinrad et al. [19]). 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, which can be computed in linear time [19].

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. By $\mathrm{lm}(X)$ (resp. $\mathrm{rm}(X)$), we denote the leftmost (that is, the smallest), resp. the rightmost (that is, the largest) leaf of X with respect to $\prec$. The *scope* of star X is the set $\mathrm{scope}(X) := \{v \mid x_l \preccurlyeq v \preccurlyeq x_r\}$ containing all vertices from $x_l = \mathrm{lm}(X)$ to $x_r = \mathrm{rm}(X)$. The *width* of star X is the cardinality of its scope, that is, $\mathrm{width}(X) := |\mathrm{scope}(X)| = r - l + 1$. The *width* of $\mathcal{P}$, $\mathrm{width}(\mathcal{P})$, is the sum of $\mathrm{width}(X)$ 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 4. 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 $\mathrm{center}(X) \in \mathrm{scope}(Y)$ and $\mathrm{center}(Y) \in \mathrm{scope}(X)$;

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 $(\mathrm{width}(\mathcal{P}), \#\text{edge-crossings}(\mathcal{P}))$. 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 for 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.

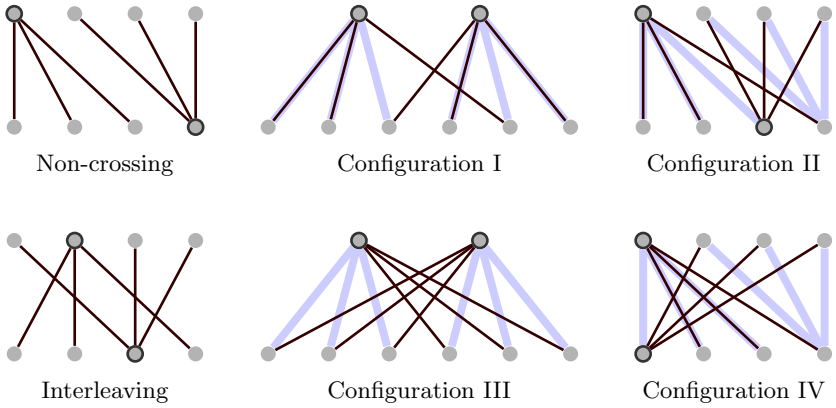


Fig. 4: Possible interactions between two stars of a partition. Centers are marked with circled nodes. The four possible configurations of star centers and scopes that are neither non-crossing nor interleaving are labeled I to IV. By Lemma 3, any partition containing one of the configurations I to IV can be edited to reduce the score (see the thick light-color edges).

Theorem 4. Star Partition *can be solved in* $O(n^2)$ *time on bipartite permutation graphs.*

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 on four vertices) as an induced subgraph. Cographs allow for a so-called *cotree* to be computed in linear time [7]. 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 [8].

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

Theorem 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 every induced subgraph containing a cycle of length at least four also contains a *triangle*, that is, a cycle of length three. We show that P_3-Partition restricted to chordal graphs is NP-hard by reduction from 3-Dimensional Matching.

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

For the reduction, we use the construction that Dyer and Frieze [11] 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

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. We are indebted to three anonymous *ICALP* reviewers whose constructive feedback helped to significantly improve our presentation.

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. Star partitions of perfect graphs. Manuscript, TU Berlin, Feb. 2014. `arXiv:1402.2589 [cs.DM]`.

[4] 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]`.

[5] 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.

[6] J. Chalopin and D. Paulusma. Packing bipartite graphs with covers of complete bipartite graphs. *Discrete Applied Mathematics*, 168(0):40–50, 2014.

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

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

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

[10] 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.

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

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

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

[14] 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.

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

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

[17] 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.

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

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

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

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

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

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

[24] 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.

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