Cluster Deletion on Interval Graphs and Split Related Graphs*

Athanasios L. Konstantinidis† Charis Papadopoulos‡§

Abstract

In the Cluster Deletion problem the goal is to remove the minimum number of edges of a given graph, such that every connected component of the resulting graph constitutes a clique. It is known that the decision version of Cluster Deletion is NP-complete on (P_5-free) chordal graphs, whereas Cluster Deletion is solved in polynomial time on split graphs. However, the existence of a polynomial-time algorithm of Cluster Deletion on interval graphs, a proper subclass of chordal graphs, remained a well-known open problem. Our main contribution is that we settle this problem in the affirmative, by providing a polynomial-time algorithm for Cluster Deletion on interval graphs.

Moreover, despite the simple formulation of a polynomial-time algorithm on split graphs, we show that Cluster Deletion remains NP-complete on a natural and slight generalization of split graphs that constitutes a proper subclass of P_5-free chordal graphs. Although the later result arises from the already-known reduction for P_5-free chordal graphs, we give an alternative proof showing an interesting connection between edge-weighted and vertex-weighted variations of the problem. To complement our results, we provide faster and simpler polynomial-time algorithms for Cluster Deletion on subclasses of such a generalization of split graphs.

1 Introduction

In graph theoretic terms, *clustering* is the task of partitioning the vertices of the graph into subsets, called *clusters*, in such a way that there should be many edges within each cluster and relatively few edges between the clusters. In many applications, the clusters are restricted to induced cliques, as the represented data of each edge corresponds to a similarity value between two objects [18, 19]. Under the term cluster graph, which refers to a disjoint union of cliques, one may find a variety of applications that have been extensively studied [1, 7, 26]. Here we consider the Cluster Deletion problem which asks for a minimum number of edge deletions from an input graph, so that the resulting graph is a disjoint union of cliques. In the decision version of the problem, we are also given an integer k and we want to decide whether at most k edge deletions are enough to produce a cluster graph.

*A preliminary version of this paper appeared as an extended abstract in the proceedings of MFCS 2019 [24]. This long version contains full proofs of all statements, a new characterization of stable-like graphs, and an extended polynomial case of starlike graphs (laminar-like graphs) not appeared in [24]. The research work done by A. L. Konstantinidis is co-financed by Greece and the European Union (European Social Fund – ESF) through the Operational Programme "Human Resources Development, Education and Lifelong Learning" in the context of the project "Strengthening Human Resources Research Potential via Doctorate Research" (MIS–5000432), implemented by the State Scholarships Foundation (IKY). The research work done by C. Papadopoulos was supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the "First Call for H.F.R.I. Research Projects to support Faculty members and Researchers and the procurement of high-cost research equipment grant", Project FANTA (eFficient Algorithms for NeTwork Analysis), number HFRI-FM17-431.

†Department of Mathematics, University of Ioannina, Greece. E-mail: a.konstantinidis@uoi.gr.

‡Department of Mathematics, University of Ioannina, Greece. E-mail: charis@uoi.gr

§Corresponding author.

Although Cluster Deletion is NP-hard on general graphs [27], settling its complexity status restricted on graph classes has attracted several researchers. Regarding the maximum degree of a graph, Komusiewicz and Uhlmann [22] have shown an interesting dichotomy result: Cluster Deletion remains NP-hard on C_4-free graphs with maximum degree four, whereas it can be solved in polynomial time on graphs having maximum degree at most three. Quite recently, Golovach et al. [14] have shown that it remains NP-hard on planar graphs. For graph classes characterized by forbidden induced subgraphs, Gao et al. [12] showed that Cluster Deletion is NP-hard on $(C_5, P_5, \text{bull}, \text{fork}, \text{co-gem}, 4\text{-pan}, \text{co-4-pan})$-free graphs and on $(2K_2, 3K_1)$-free graphs. Regarding H-free graphs, Grüttemeier et al. [16], showed a complexity dichotomy result for any graph H consisting of at most four vertices. In particular, for any graph H on four vertices with $H \notin \{P_4, \text{paw}\}$, Cluster Deletion is NP-hard on H-free graphs, whereas it can be solved in polynomial time on P_4- or paw-free graphs [16]. Interestingly, Cluster Deletion remains NP-hard on P_5-free chordal graphs [3].

On the positive side, Cluster Deletion has been shown to be solved in polynomial time on cographs [12], proper interval graphs [3], split graphs [3], and P_4-reducible graphs [2]. More precisely, iteratively picking maximum cliques defines a clustering on the graph which actually gives an optimal solution on cographs (i.e., P_4-free graphs), as shown by Gao et al. in [12]. In fact, the greedy approach of selecting a maximum clique provides a 2-approximation algorithm, though not necessarily in polynomial-time [9]. As the problem is already NP-hard on chordal graphs [3], it is natural to consider subclasses of chordal graphs such as interval graphs and split graphs. Although for split graphs there is a simple polynomial-time algorithm, restricted to interval graphs only the complexity on proper interval graphs was determined by giving a solution that runs in polynomial-time [3]. Settling the complexity of Cluster Deletion on interval graphs, was left open [3, 2, 12].

For proper interval graphs, Bonomo et al. [3] characterized their optimal solution by consecutiveness of each cluster with respect to their natural ordering of the vertices. Based on this fact, a dynamic programming approach led to a polynomial-time algorithm. It is not difficult to see that such a consecutiveness does not hold on interval graphs, as potential clusters might require to break in the corresponding vertex ordering. Here we characterize an optimal solution of interval graphs whenever a cluster is required to break. In particular, we take advantage of their consecutive arrangement of maximal cliques and describe subproblems of maximal cliques containing the last vertex. One of our key observations is that the candidate clusters containing the last vertex can be enumerated in polynomial time given two vertex orderings of the graph. We further show that each such candidate cluster separates the graph in a recursive way with respect to optimal subsolutions, that enables to define our dynamic programming table to keep track about partial solutions. Thus, our algorithm for interval graphs suggests to consider a particular consecutiveness of a solution and apply a dynamic programming approach defined by two vertex orderings. The overall running time of our algorithm is $O(n^6)$ for an interval graph on n vertices and, thus, exploiting the first polynomial-time such algorithm.

Furthermore, we complement the previously-known NP-hardness of Cluster Deletion on P_5-free chordal graphs, by providing a proper subclass of such graphs for which we prove that the problem remains NP-hard. This result is inspired and motivated by the very simple characterization of an optimal solution on split graphs: either a maximal clique constitutes the only non-edgeless cluster, or there are exactly two non-edgeless clusters whenever there is a vertex of the independent set that is adjacent to all the vertices of the clique except one [3]. Due to the fact that true twins belong to the same cluster in an optimal solution, it is natural to extend split graphs by allowing two vertices that do not belong to the clique to be adjacent only if they are true twins, as they are expected not to influence the solution

characterization. Surprisingly, we show that Cluster Deletion remains NP-complete even on such a slight generalization of split graphs. This is achieved by observing that the constructed graphs given in the reduction for P_5-free graphs [3], constitute such split-related graphs. However, here we give a different reduction that highlights an interesting connection between edge-weighted and vertex-weighted split graphs. In fact, the resulting split-related graphs are known as *starlike graphs* which are exactly the intersection graphs of subtrees of a star [6]. We then study two different classes of starlike graphs that can be viewed as the parallel of split graphs that admit disjoint clique-neighborhood (that we call *stable-like graphs*) and nested clique-neighborhood (that we call *threshold-like graphs*). For Cluster Deletion we provide polynomial-time algorithms on both classes of graphs. In particular, for the former case, a polynomial-time algorithm is already known and is achieved through computing a minimizer of submodular functions [3]. Here we provide a simpler and faster (linear-time) algorithm for Cluster Deletion on such graphs that avoids the usage of submodular minimization. In order to unify both classes, we also consider the starlike graphs that are obtained from disjoint threshold-like graphs with a common clique (that we call *laminar-like graphs*). Our general approach that uses both subroutines on stable-like and threshold-like graphs, results in a quadratic-time algorithm for Cluster Deletion on laminar-like graphs.

2 Preliminaries

All graphs considered here are simple and undirected. A graph is denoted by $G = (V, E)$ with vertex set V and edge set E. We use the convention that $n = |V|$ and $m = |E|$. The *neighborhood* of a vertex v of G is $N(v) = \{x \mid vx \in E\}$ and the *closed neighborhood* of v is $N[v] = N(v) \cup \{v\}$. For $S \subseteq V$, $N(S) = \bigcup_{v \in S} N(v) \setminus S$ and $N[S] = N(S) \cup S$. A graph H is a *subgraph* of G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. For $X \subseteq V(G)$, the subgraph of G *induced* by X, $G[X]$, has vertex set X, and for each vertex pair u, v from X, uv is an edge of $G[X]$ if and only if $u \neq v$ and uv is an edge of G. For $R \subseteq E(G)$, $G \setminus R$ denotes the graph $(V(G), E(G) \setminus R)$, that is a subgraph of G and for $S \subseteq V(G)$, $G - S$ denotes the graph $G[V(G) - S]$, that is an induced subgraph of G. For two set of vertices A and B, we write $E(A, B)$ to denote the edges that have one endpoint in A and one endpoint in B. Two adjacent vertices u and v are called *true twins* if $N[u] = N[v]$, whereas two non-adjacent vertices x and y are called *false twins* if $N(u) = N(v)$.

A *clique* of G is a set of pairwise adjacent vertices of G, and a *maximal clique* of G is a clique of G that is not properly contained in any clique of G. An *independent set* of G is a set of pairwise non-adjacent vertices of G. For $k \geq 2$, the chordless path on k vertices is denoted by P_k and the chordless cycle on k vertices is denoted by C_k. For an induced path P_k, the vertices of degree one are called endvertices. A vertex v is *universal* in G if $N[v] = V(G)$ and v is *isolated* if $N(v) = \emptyset$. A graph is *connected* if there is a path between any pair of vertices. A *connected component* of G is a maximal connected subgraph of G. For a set of finite graphs $\mathcal{H}$, we say that a graph G is $\mathcal{H}$-free if G does not contain an induced subgraph isomorphic to any of the graphs of $\mathcal{H}$. *Contracting* a set of vertices S is the operation of substituting the vertices of S by a new vertex w with $N(w) = N(S)$.

The problem of Cluster Deletion is formally defined as follows: given a graph $G = (V, E)$, the goal is to compute the minimum set $F \subseteq E(G)$ of edges such that every connected component of $G \setminus F$ is a clique. A *cluster graph* is a P_3-free graph, or equivalently, any of its connected components is a clique. Thus, the task of Cluster Deletion is to turn the input graph G into a cluster graph by deleting the minimum number of edges. Let $S = C_1, \ldots, C_k$ be a solution of Cluster Deletion such that $G[C_i]$ is a clique. In such terms, the problem can be viewed as a vertex partition problem into $C_1, \ldots, C_k$. Each C_i is simple called *cluster*.

Edgeless clusters, i.e., clusters containing exactly one vertex, are called *trivial clusters*. The edges of G are partitioned into *internal* and *external* edges: an internal edge uv has both its endpoints $u, v \in C_i$ in the same cluster C_i, whereas an external edge uv has its endpoints in different clusters $u \in C_i$ and $v \in C_j$, for $i \neq j$. Then, the goal of Cluster Deletion is to minimize the number of external edges which is equivalent to maximize the number of internal edges. We write $S(G)$ to denote an optimal solution for Cluster Deletion of the graph G, that is, a cluster subgraph of G having the maximum number of edges. Given a solution $S(G)$, the number of edges incident only to the same cluster, that is the number of internal edges, is denoted by $|S(G)|$.

For a clique C, we say that a vertex x is *C-compatible* if $C \setminus \{x\} \subseteq N(x)$. We start with few preliminary observations regarding twin vertices. Notice that for true twins x and y, if x belongs to any cluster C then y is C-compatible.

Lemma 2.1 ([3])**.** *Let x and y be true twins in G. Then, in any optimal solution x and y belong to the same cluster.*

The above lemma shows that we can contract true twins and look for a solution on a vertex-weighted graph that does not contain true twins. Even though false twins cannot be grouped into the same cluster as they are non-adjacent, we can actually disregard one of the false twins whenever their neighborhood forms a clique.

Lemma 2.2. *Let x and y be false twins in G such that $N(x) = N(y)$ is a clique. Then, there is an optimal solution such that x constitutes a trivial cluster.*

Proof. Let C_x and C_y be the clusters of x and y, respectively, in an optimal solution such that $|C_x| \geq 2$ and $|C_y| \geq 2$. We construct another solution by replacing both clusters by $C_x \cup C_y \setminus \{y\}$ and $\{y\}$, respectively. To see that this indeed a solution, first observe that x is adjacent to all the vertices of $C_y \setminus \{y\}$ because $N(x) = N(y)$, and $C_x \cup C_y \setminus \{y\} \subseteq N[x]$ forms a clique by the assumption. Moreover, since $|C_x| \geq 2$ and $|C_y| \geq 2$, we know that $|C_x|+|C_y| \leq |C_x||C_y|$, implying that the number of internal edges in the constructed solution is at least as the number of internal edges of the optimal solution. □

Moreover, we prove the following generalization of Lemma 2.1.

Lemma 2.3. *Let C and C' be two clusters of an optimal solution and let $x \in C$ and $y \in C'$. If y is C-compatible then x is not C'-compatible.*

Proof. Let S be an optimal solution such that $C, C' \in S$. Assume for contradiction that x is C'-compatible. We show that S is not optimal. Since y is C-compatible, we can move y to C and obtain a solution S_y that contains the clusters $C \cup \{y\}$ and $C' \setminus \{y\}$. Similarly, we construct a solution S_x from S, by moving x to C' so that $C \setminus \{x\}, C' \cup \{x\} \in S_x$. Notice that the S_x forms a clustering, since x is C'-compatible. We distinguish between the following cases, according to the values $|C|$ and $|C'|$.

- If $|C| \geq |C'|$ then $|S_y| > |S|$, because $\binom{|C|+1}{2} + \binom{|C'|-1}{2} > \binom{|C|}{2} + \binom{|C'|}{2}$.
- If $|C| < |C'|$ then $|S_x| > |S|$, because $\binom{|C|-1}{2} + \binom{|C'|+1}{2} > \binom{|C|}{2} + \binom{|C'|}{2}$.

In both cases we reach a contradiction to the optimality of S. Therefore, x is not C'-compatible. □

Corollary 2.4. *Let C be a cluster of an optimal solution and let $x \in C$. If there is a vertex y that is C-compatible and $N[y] \subseteq N[x]$, then y belongs to C.*

Proof. Assume for contradiction that y belongs to a cluster C' different than C. Then, observe that x is C'-compatible. Indeed, for any vertex u of C', we know $xu \in E(G)$, since u is adjacent to y and $N[y] \subseteq N[x]$. Thus, by Lemma 2.3 we reach a contradiction, so that $y \in C$. □

3 Polynomial-time algorithm on interval graphs

Here we present a polynomial-time algorithm for the Cluster Deletion problem on interval graphs. A graph is an *interval graph* if there is a bijection between its vertices and a family of closed intervals of the real line such that two vertices are adjacent if and only if the two corresponding intervals intersect. Such a bijection is called an *interval representation* of the graph, denoted by $\mathcal{I}$. We identify the intervals of the given representation with the vertices of the graph, interchanging these notions appropriately. Whether a given graph is an interval graph can be decided in linear time and if so, an interval representation can be generated in linear time [11]. Notice that every induced subgraph of an interval graph is an interval graph.

Let G be an interval graph. Instead of working with the interval representation of G, we consider its sequence of maximal cliques. It is known that a graph G with p maximal cliques is an interval graph if and only if there is an ordering $K_1, \ldots, K_p$ of the maximal cliques of G, such that for each vertex v of G, the maximal cliques containing v appear consecutively in the ordering (see e.g., [4]). A path $\mathcal{P} = K_1 \cdots K_p$ following such an ordering is called a *clique path* of G. Notice that a clique path is not necessarily unique for an interval graph. Also note that an interval graph with n vertices contains at most n maximal cliques. By definition, for every vertex v of G, the maximal cliques containing v form a connected subpath in $\mathcal{P}$.

Given a vertex v, we denote by $K_{a(v)}, \ldots, K_{b(v)}$ the maximal cliques containing v with respect to $\mathcal{P}$, where $K_{a(v)}$ and $K_{b(v)}$ are the *first* (*leftmost*) and *last* (*rightmost*) maximal cliques containing v. Notice that $a(v) \leq b(v)$ holds. Moreover, for every edge of G there is a maximal clique K_i of $\mathcal{P}$ that contains both endpoints of the edge. Thus, two vertices u and v are adjacent if and only if $a(v) \leq a(u) \leq b(v)$ or $a(v) \leq b(u) \leq b(v)$.

For a set of vertices $U \subseteq V$, we write $a\text{-}\min U$ and $a\text{-}\max U$ to denote the minimum and maximum value, respectively, among all $a(u)$ with $u \in U$. Similarly, $b\text{-}\min U$ and $b\text{-}\max U$ correspond to the minimum and maximum value, respectively, with respect to $b(u)$.

With respect to the Cluster Deletion problem, observe that for any cluster C of a solution, we know that $C \subseteq K_i$ where $K_i \in \mathcal{P}$, as C forms a clique. A vertex y is called *guarded* by two vertices x and z if

$$\min\{a(x), a(z)\} \leq a(y) \text{ and } b(y) \leq \max\{b(x), b(z)\}.$$

For a clique C, observe that y is C-compatible if and only if there exists a maximal clique K_i such that $C \subseteq K_i$ with $a(y) \leq i \leq b(y)$.

Lemma 3.1. *Let x, y, z be three vertices of G such that y is guarded by x and z. If x and z belong to the same cluster C of an optimal solution and y is C-compatible then $y \in C$.*

Proof. To ease the presentation, for three non-negative numbers i, j, k we write $i \in [j, k]$ if $j \leq i \leq k$ holds. Without loss of generality, assume that $a(y) \in [a(x), a(z)]$. Assume for contradiction that y belongs to another cluster C'. We apply Lemma 2.3 to either x and y or z and y. To do so, we need to show that x is C'-compatible or z is C'-compatible, as y is already C-compatible. Since C' is a cluster that contains y, there is a maximal clique K_i such that $C' \subseteq K_i$ with $i \in [a(y), b(y)]$.

We show that $i \in [a(x), b(x)]$ or $i \in [a(z), b(z)]$. If $i \notin [a(x), b(x)]$ then $b(x) < i \leq b(y)$, because $a(x) \leq a(y) \leq i$. As y is guarded by x and z, we know that $i \leq b(y) \leq b(z)$. Now observe that if $i < a(z)$ then $b(x) < a(z)$, implying that x and z are non-adjacent, reaching a contradiction to the fact that $x, z \in C$. Thus, $a(z) \leq i \leq b(z)$ which shows that $i \in [a(z), b(z)]$. This means that $i \in [a(x), b(x)]$ or $i \in [a(z), b(z)]$.

Hence, x or z belong to the maximal clique K_i for which $C' \subseteq K_i$. Therefore, at least one of x or z is C'-compatible and by Lemma 2.3 we conclude that $y \in C$. □

Let $v_1, \ldots, v_n$ be an ordering of the vertices such that $b(v_1) \leq \cdots \leq b(v_n)$. For every v_i, v_j with $b(v_i) \leq b(v_j)$, we define the following set of vertices:

$$V_{i,j} = \{v \in V(G) : \min\{a(v_i), a(v_j)\} \leq a(v) \text{ and } b(v) \leq b(v_j)\}.$$

That is, $V_{i,j}$ contains all vertices that are guarded by v_i and v_j. We write $a(i,j)$ to denote the value of $\min\{a(v_i), a(v_j)\}$ and we simple write $K_{a(j)}$ and $K_{b(j)}$ instead of $K_{a(v_j)}$ and $K_{b(v_j)}$. Notice that for a neighbor u of v_j with $u \in V_{i,j}$, we have either $a(v_j) \leq a(u)$ or $a(v_i) \leq a(u) \leq a(v_j)$. This means that all neighbors of v_j that are totally included (i.e., all vertices u such that $a(v_j) \leq a(u) \leq b(u) \leq b(v_j)$) belong to $V_{i,j}$ for any v_i with $b(v_i) \leq b(v_j)$. To distinguish such neighbors of v_j, we define the following sets:

- $U(j)$ contains the neighbors $u \in V_{i,j}$ of v_j such that $a(u) < a(v_j) \leq b(u) \leq b(v_j)$ (neighbors of v_j in $V_{i,j}$ that partially overlap v_j).
- $M(j)$ contains the neighbors $w \in V_{i,j}$ of v_j such that $a(v_j) \leq a(w) \leq b(w) \leq b(v_j)$ (neighbors of v_j that are totally included within v_j).

In the forthcoming arguments, we restrict ourselves to the graph induced by $V_{i,j}$. It is clear that the first maximal clique that contains a vertex of $V_{i,j}$ is $K_{a(i,j)}$, whereas the last maximal clique is $K_{b(j)}$.

We now explain the necessary sets that our dynamic programming algorithm uses in order to compute an optimal solution of G.

Definition 3.2 (Optimal solutions $A_{i,j}$)**.** *For two vertices v_i, v_j with $b(v_i) \leq b(v_j)$,*

- *$A_{i,j}$ is the value of an optimal solution for* Cluster Deletion *of the graph $G[V_{i,j}]$.*

To ease the notation, when we say a cluster of $A_{i,j}$ we mean a cluster of an optimal solution of $G[V_{i,j}]$. Notice that $A_{1,n}$ is the desired value for the whole graph G, since $V_{1,n} = V(G)$.

Our task is to construct the values for $A_{i,j}$ by taking into account all possible clusters that contain v_j. To do so, we show that (i) the number of clusters containing v_j in $A_{i,j}$ is polynomial and (ii) each such candidate cluster containing v_j separates the graph in a recursive way with respect to optimal subsolutions.

Observe that if $v_iv_j \in E(G)$ then $v_i \in U(j)$ if and only if $a(v_i) < a(v_j)$, whereas $v_i \in M(j)$ if and only if $a(v_j) \leq a(v_i)$; in the latter case, it is not difficult to see that $V_{i,j} = M(j) \cup \{v_j\}$, according to the definition of $V_{i,j}$. Thus, whenever $v_i \in M(j)$ holds, we have $V_{i,j} = V_{j,j}$. The candidates of a cluster of $A_{i,j}$ containing v_j lie among $U(j)$ and $M(j)$. Let us show with the next two lemmas that we can restrict ourselves into a polynomial number of such candidates. To avoid repeating ourselves, in the forthcoming statements we let v_i, v_j be two vertices with $b(v_i) \leq b(v_j)$.

Lemma 3.3. *Let C be a cluster of $A_{i,j}$ containing v_j. If there is a vertex $w \in M(j)$ such that $w \in C$ then there is a maximal clique K_t with $a(v_j) \leq t \leq b(v_j)$ such that $K_t \cap M(j) \subseteq C$ and $C \cap M(j) \subseteq K_t$.*

Proof. Since $v_j, w \in C$, we know that there is a maximal clique K_t for which $C \subseteq K_t$ with $a(v_j) \leq a(w) \leq t \leq \min\{b(v_j), b(w)\}$. We show that all other vertices of $K_t \cap M(j)$ are guarded by v_j and w. Notice that for every vertex $y \in M(j)$ we already know that $a(v_j) \leq a(y)$ and $b(y) \leq b(v_j)$. Thus, for every vertex $y \in M(j)$ we have $a(v_j) = \min\{a(v_j), a(w)\} \leq a(y)$ and $b(y) \leq \max\{b(v_j), b(w)\}$. This means that all vertices of $K_t \cap M(j) \setminus \{w\}$ are guarded by v_j and w. Moreover, since $C \subseteq K_t$, we know that all vertices of $K_t \cap M(j)$ are C-compatible. Therefore, we apply Lemma 3.1 to every vertex of $K_t \cap M(j)$, showing that $K_t \cap M(j) \subseteq C$. Furthermore, there is no vertex of $M(j) \setminus K_t$ that belongs to C, because $C \subseteq K_t$. □

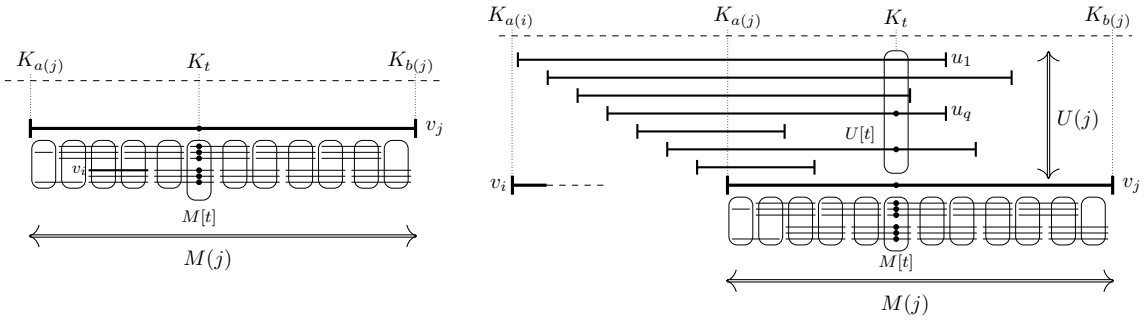

Figure 1: Illustrating the sets $M(j)$ and $U(j)$ for v_j. The left part shows the case in which $v_i \in M(j)$ (or, equivalently, $V_{i,j} = V_{j,j}$), whereas the right part corresponds to the case in which $a(v_i) < a(v_j)$.

By Lemma 3.3, we know that we have to pick the entire set $K_t \cap M(j)$ for constructing candidates to form a cluster that contains v_j and some vertices of $M(j)$. As there are at most n choices for K_t, we get a polynomial number of such candidate sets. We next show that we can construct polynomial number of candidate sets that contain v_j and vertices of $U(j)$. For doing so, we consider the vertices of $U(j)$ increasingly ordered with respect to their first maximal clique. More precisely, let $U(j)_{\leq a} = (u_1, \ldots, u_{|U(j)|})$ be an increasingly order of the vertices of $U(j)$ such that $a(u_1) \leq \cdots \leq a(u_{|U(j)|})$. The right part of Figure 1 illustrates the corresponding case.

Lemma 3.4. *Let C be a cluster of $A_{i,j}$ containing v_j and let $u_q \in U(j)_{\leq a}$.If $u_q \in C$ then every vertex of $\{u_{q+1}, \ldots, u_{|U(j)|}\}$ that is C-compatible belongs to C.*

Proof. Let u be a vertex of $\{u_{q+1}, \ldots, u_{|U(j)|}\}$. We show that u is guarded by u_q and v_j. By the definition of $U(j)_{\leq a}$, we know that $a(u_q) < a(u) < a(v_j)$. Moreover, observe that $b(u) \leq b(v_j)$ holds by the fact that $u \in V_{i,j}$ and $b(u_q) \leq b(v_j)$. Thus, we apply Lemma 3.1 to u, because $u_q, v_j \in C$ and u is C-compatible, showing that $u \in C$ as desired. □

For $a(v_j) \leq t \leq b(v_j)$, let $M[t] = K_t \cap M(j)$. Observe that each $M[t]$ may be an empty set. On the part $M(j)$, all vertices are grouped into the sets $M[a(v_j)], \ldots, M[b(v_j)]$. Similar to $M[t]$, let $U[t] = U(j) \cap K_t$. Then, all vertices of $U[t]$ are $\{v_j, M[t]\}$-compatible and all vertices of $M[t]$ are $\{v_j, U[t]\}$-compatible. Figure 1 depicts the corresponding sets.

Lemma 3.5. *Let C be a cluster of $A_{i,j}$ containing v_j. Then, there is $a(v_j) \leq t \leq b(v_j)$ such that $M[t] \subseteq C$.*

Proof. Assume for contradiction that no set $M[t]$ is contained in C. Let $U_C = U(j) \cap C$ and let $i' = b\text{-}\min(U_C)$. Notice that $C = \{v_j\} \cup U_C$ because of the assumption as there are no other neighbors of v_j in $V_{i,j}$. Then, $a(v_j) \leq i' \leq b(v_j)$ holds, because $v_j \in C$. We show that $M[i'] \subseteq C$. Observe that $C \subseteq K_{i'}$. If $M[i'] = \emptyset$ then clearly $M[i'] \subset C$. Assume that $M[i'] \neq \emptyset$ and let C' be a non-empty subset of $M[i']$ that forms a cluster in $A_{i,j}$. Then, all vertices of C are C'-compatible and all vertices of C' are C-compatible, because $C, C' \in K_t$. Thus, we reach a contradiction by Lemma 2.3 to the optimality of $A_{i,j}$. This means that there is a vertex $w \in M(j)$ that is contained in C together with v_j. Therefore, by Lemma 3.3, there is a set $M[t] = K_t \cap M(j)$ that is included in C. □

All vertices of a cluster C containing v_j belong to $U(j) \cup M(j)$. Thus, $C \setminus \{v_j\}$ can be partitioned into $C \cap U(j)$ and $C \cap M(j)$. Also notice that $C \subseteq K_t$ for some $a(v_j) \leq t \leq b(v_j)$. Combined with the previous lemmas, we can enumerate all such subsets C of $U(j) \cup M(j)$ in

polynomial-time. In particular, we first build all candidates for $C \cap M(j)$, which are exactly the sets $M[t]$ by Lemma 3.3 and Lemma 3.5. Then, for each of such candidate $M[t]$, we apply Lemma 3.4 to construct all subsets containing the last q vertices of $U[t]_{\leq a}$. Thus, there are at most n^2 number of candidate sets from the vertices of $U(j) \cup M(j)$ that belong to the same cluster with v_j.

3.1 Splitting into partial solutions

We further partition the vertices of $M(j)$. Given a pivot group $M[t]$, we consider the vertices that lie on the right part of $M[t]$. More formally, for $a(v_j) \leq t < b(v_j)$, we define the set

$$B_j(t) = \big(\big(K_{t+1} \cup \cdots \cup K_{b(j)}\big) \setminus K_t\big) \cap M(j).$$

The reason of breaking the vertices of the part $M(j)$ into sets $B_j(t)$ is the following.

Lemma 3.6. *Let C be a cluster of $A_{i,j}$ such that $\{v_j\} \cup M[t] \subseteq C$, for $a(v_j) \leq t \leq b(v_j)$. Then, for any two vertices $x \in V_{i,j} \setminus B_j(t)$ and $y \in B_j(t)$, there is no cluster of $A_{i,j}$ that contains both of them.*

Proof. First observe that $y \in (M[t+1] \cup \cdots \cup M[b(j)]) \setminus M[t]$. We consider two cases for x, depending on whether $x \in M(j)$ or not. Assume that $x \in M(j)$. Then, we show that $xy \notin E(G)$. To see this, observe that by the definition of each group $M[t] = K_t \cap M(j)$, there is no maximal clique that contains both x and y. Thus, there is no cluster that contains both of them.

Now assume that $x \in U(j)$. If $x \in C$, then y does not belong to K_t, so that $y \notin C$. If $x \notin C$, then we show that x does not belong to a cluster with any vertex of $B_j(t)$. Assume for contradiction that x belongs to a cluster C' such that $C' \cap B_j(t) \neq \emptyset$. This means that $x \in K_{i'}$ with $t < i' \leq b(v_j)$ and $C' \subseteq K_{i'}$. Then v_j is C'-compatible and x is C-compatible, as both x and v_j belong to $K_t \cap K_{i'}$. Therefore, by Lemma 2.3 we reach a contradiction to x and v_j belonging to different clusters. □

Definition 3.7 (Optimal solution $A(S)$)**.** *For a non-empty set $S \subseteq V(G)$, we write $A(S)$ to denote the following solutions:*

- *$A(S) = A_{i',j'}$, where $v_{i'}$ is the vertex of S having the smallest $a(v_{i'})$ and $v_{j'}$ is the vertex of S having the largest $b(v_{j'})$.*

Having this notation, observe that $A_{i,j} = A(V_{i,j})$, for any v_i, v_j with $b(v_i) \leq b(v_j)$. However, it is important to notice that $A(S)$ does not necessarily represent the optimal solution of $G[S]$, since the vertices of S may not be consecutive with respect to $V_{i',j'}$, so that S is only a subset of $V_{i',j'}$ in the corresponding solution $A_{i',j'}$ for $A(S)$. Under the following assumptions, with the next result we show that for the chosen sets we have $S = V_{i',j'}$.

Observation 3.8. *Let v_i, v_j be two vertices with $b(v_i) \leq b(v_j)$ and let $V_t = K_t \cap V_{i,j}$, for any maximal clique K_t of $\mathcal{P}$ with $a(v_j) \leq t \leq b(v_j)$.*

(i) If $S_L = \big(V_{a(i,j)} \cup \cdots \cup V_{t-1}\big) \setminus V_t$ then $S_L = V_{i',j'}$, where $i' = a\text{-}\min(S_L)$ and $j' = b\text{-}\max(S_L)$.

(ii) If $S_R = \Big(V_{t+1} \cup \cdots \cup V_{b(v_j)}\Big) \setminus V_t$ then $S_R = V_{i',j'}$, where $i' = a\text{-}\min(S_R)$ and $j' = b\text{-}\max(S_R)$.

Proof. We prove the case for $S_L = \big(V_{a(i,j)} \cup \cdots \cup V_{t-1}\big) \setminus V_t$. As each V_t contains vertices of $V_{i,j}$, we have $V_{i',j'} \subseteq V_{i,j}$. Observe that either $a(v_{i'}) < a(v_{j'})$ or $a(v_{j'}) \leq a(v_{i'})$. In both cases we show that $b(v_{j'}) = t-1$. Assume that there is a vertex $w \in S_L$ with $t-1 < b(w)$. Then $a(w) \leq t-1$ as $w \in S_L$, and $w \in K_t$ by the consecutiveness of the clique path. This shows that $w \notin S_L$ because $w \in V_t$. Thus, $b(v_{j'}) = t-1$. We show that $a(v_{i'}) = \min\{a(v_i), a(v_j)\}$. If there is a vertex w in S_L with $a(w) < \min\{a(v_i), a(v_j)\}$ then $w \notin V_{i,j}$ leading to a contradiction that $V_{i',j'} \subseteq V_{i,j}$. Hence we have $a(v_{i'}) = \min\{a(v_i), a(v_j)\}$ and $b(v_{j'}) = t-1$. Moreover, observe that by the definition of S_L, we already know that $S_L \subseteq V_{i',j'}$. Now it remains to notice that for every vertex w with $\min\{a(v_i), a(v_j)\} \leq a(w)$ and $b(w) \leq t-1$ we have $w \in S_L$. This follows from the fact that $w \in V_{a(w)} \cup \cdots \cup V_{b(w)}$ and $w \notin V_t$. Therefore we get $S_L = V_{i',j'}$. Completely symmetric arguments along the previous lines, shows the case for S_R. □

Given the clique path $\mathcal{P} = K_1 \cdots K_p$, a *clique-index* t is an integer $1 \leq t \leq p$. Let $\ell(j), r(j)$ be two clique-indices such that $a(i,j) \leq \ell(j) \leq a(v_j)$ and $a(v_j) \leq r(j) \leq b(v_j)$. We denote by $\ell_r(j)$ the minimum value of $a(v)$ among all vertices of $v \in K_{r(j)} \cap V_{i,j}$ having $\ell(j) \leq a(v)$. Clearly, $\ell(j) \leq \ell_r(j) \leq r(j)$ holds.

Definition 3.9 (Admissible pair and crossing)**.** *A pair of clique-indices $(\ell(j), r(j))$ is called* admissible pair *for a vertex v_j, if both*

- $a(i,j) \leq \ell(j) \leq a(v_j)$ *and*
- $a(v_j) \leq r(j) \leq b(v_j)$ *hold.*

Given an admissible pair $(\ell(j), r(j))$, we define the following set of vertices:

- $C(\ell(j), r(j)) = \{z \in V_{i,j} : \ell_r(j) \leq a(z) \leq r(j) \leq b(z)\}$.

We say that a vertex u crosses the pair $(\ell(j), r(j))$ if we have $a(u) < \ell_r(j)$ and $r(j) \leq b(u)$.

Observe that all vertices of $C(\ell(j), r(j))$ induce a clique in G, because $C(\ell(j), r(j)) \subseteq K_{r(j)}$. It is not difficult to see that for a vertex u that crosses $(\ell(j), r(j))$, we have $u \notin C(\ell(j), r(j))$. We prove the following properties of $C(\ell(j), r(j))$.

Lemma 3.10. *Let $v_{i'}, v_{j'}$ be two vertices with $b(v_{i'}) \leq b(v_{j'})$ and let (ℓ, r) be an admissible pair for $v_{j'}$. Moreover, let v_i, v_j be the vertices of $V_{i',j'} \setminus C(\ell, r)$ having the smallest $a(v_i)$ and largest $b(v_j)$, respectively. If the vertices of $C(\ell, r)$ form a cluster in $A_{i',j'}$ then the following statements hold:*

1. *$V_{i,j} = V_{i',j'} \setminus C(\ell, r)$.*
2. *If $a(x) \leq r \leq b(x)$ holds for a vertex $x \in V_{i,j}$, then x crosses (ℓ, r).*
3. *Every vertex of $B_j(r)$ does not belong to the same cluster with any vertex of $V_{i,j} \setminus B_j(r)$.*
4. *Every vertex that crosses (ℓ, r) does not belong to the same cluster with any vertex $y \in V_{i,j}$ having $\ell_r \leq a(y)$.*

Proof. First we show that $V_{i,j} = V_{i',j'} \setminus C(\ell, r)$. Assume that there is a vertex $v \in V_{i,j} \setminus V_{i',j'}$. Then $v \notin C(\ell, r)$ and v is distinct from v_i, v_j because, by definition, $v_i, v_j \in V_{i',j'}$. Also notice that $v \in V_{i,j}$ implies $a(i,j) \leq a(v)$ and $b(v) \leq b(v_j)$. By the second inequality, we get $b(v) \leq b(v_j) \leq b(v_{j'})$. Suppose that $a(v) < a(i',j')$. As we already know that $a(i,j) \leq a(v)$, we conclude that $a(i,j) < a(i',j')$ leading to a contradiction that $v_i, v_j \in V_{i',j'}$. Thus we have $a(i',j') \leq a(v)$ and $b(v) \leq b(v_{j'})$, showing that $v \in V_{i',j'}$. This means that $V_{i,j} \subset V_{i',j'}$, so that $V_{i,j} = V_{i',j'} \setminus C(\ell, r)$.

For the second statement, observe that if $\ell_r \leq a(x)$ then $x \in C(\ell, r)$. Since $x \in V_{i,j}$, we conclude that $x \notin C(\ell, r)$ by the first statement. Thus $a(x) < \ell_r$ holds, implying that x crosses (ℓ, r).

With respect to the third statement, observe that no vertex of $B_j(r)$ belongs to the clique K_r. This means that all vertices of $B_j(r)$ belong to both sets $V_{i,j}$ and $V_{i',j'}$. Thus Lemma 3.6 and the first statement show that no two vertices $x \in V_{i,j} \setminus B_j(r)$ and $y \in B_j(r)$ belong to the same cluster.

For the fourth statement, let x be a vertex that crosses (ℓ, r). By the first statement we know that $x \in V_{i,j}$. If $r < a(y)$ then $y \in B_j(r)$ and the third statement show that x and y do not belong to the same cluster. Suppose that $\ell_r \leq a(y) \leq r$. If $r \leq b(y)$ then $y \in C(\ell, r)$ contradicting the fact that $y \in V_{i,j}$. Putting together, we have $\ell_r \leq a(y) \leq b(y) < r$. Now assume for contradiction that x and y belong to the same cluster C_{xy}. By the fact that $a(x) < a(y)$, observe that $a(y) \leq a\text{-}\min(C_{xy}) \leq b\text{-}\min(C_{xy}) \leq \min\{b(v_j), b(y)\}$. We consider the graph induced by $V_{i',j'}$. We show that there is a vertex of C_{xy} that is $C(\ell, r)$-compatible and there is a vertex of $C(\ell, r)$ that is C_{xy}-compatible. Notice that x is $C(\ell, r)$-compatible, because x crosses (ℓ, r) so that $x \in K_r$. To see that there is a vertex of $C(\ell, r)$ that is C_{xy}-compatible, choose z to be the vertex of $C(\ell, r)$ having the smallest $a(z)$. This means that $a(z) = \ell_r$. Then z is adjacent to every vertex of C_{xy} because $a(z) \leq a(y)$ and $b(y) < r \leq b(z)$. Thus, $z \in C(\ell, r)$ is C_{xy}-compatible. Therefore, Lemma 2.3 shows the desired contradiction, implying that x and y do not belong to the same cluster. □

Notice that the number of admissible pairs $(\ell(j), r(j))$ for v_j is polynomial because there are at most n choices for each clique-index. Moreover, if $v_i \in M(j)$ then we have $\ell(j) = a(v_j)$.

Definition 3.11 (Bounding pair)**.** *A pair of clique-indices (ℓ, r) with $\ell \leq r$ is called* bounding pair for v_j *if either $b(v_j) < r$ holds, or v_j crosses (ℓ, r). Given an bounding pair (ℓ, r) for v_j, we write $(\ell(j), r(j)) \prec (\ell, r)$ to denote the set of admissible pairs $(\ell(j), r(j))$ for v_j such that*

- *$r(j) \leq b(v_j)$, whenever $b(v_j) < r$ holds, and*
- *$r(j) < \ell$, otherwise.*

Observe that if $b(v_j) < r$ holds, then $(\ell(j), r(j)) \prec (\ell, r)$ describes all admissible pairs for v_j with no restriction, regardless of ℓ. On the other hand, if $\ell < a(v_j)$ and $r \leq b(v_j)$ hold, then (ℓ, r) is not a bounding pair for v_j. In fact, we will show that the latter case will not be considered in our partial subsolutions. For any admissible pair $(\ell(j), r(j))$ and any bounding pair (ℓ, r) for v_j, observe that $v_j \in C(\ell(j), r(j))$ and $v_j \notin C(\ell, r)$. Intuitively, an admissible pair $(\ell(j), r(j))$ corresponds to the cluster containing v_j, whereas a bounding pair (ℓ, r) forbids v_j to select certain vertices as they have already formed a cluster that does not contain v_j.

Our task is to construct subsolutions over all admissible pairs for v_j with the property that the vertices of $C(\ell(j), r(j))$ form a cluster. To do so, we consider a vertex $v_{j'}$ with $b(v_j) \leq b(v_{j'})$ and a cluster containing $v_{j'}$. Let (ℓ, r) be an admissible pair for $v_{j'}$ such that $a(v_j) \leq r \leq b(v_j)$. The previous results suggest to consider solutions in which the vertices of $C(\ell, r)$ form a cluster in an optimal solution. It is clear that if $\ell \leq a(v_j)$ then $v_j \in C(\ell, r)$. Moreover, if $b(v_j) < r$, then no vertex of $V_{i,j}$ belongs to $C(\ell, r)$. Thus, we need to construct solutions for $A_{i,j}$, whenever (ℓ, r) is a bounding pair for v_j and the vertices of $C(\ell, r)$ form a cluster. Such an idea is formally described in the following restricted solutions.

Definition 3.12 ((ℓ, r)-restricted solution)**.** *Let (ℓ, r) be a bounding pair for v_j. We call the following solution,* (ℓ, r)-restricted solution*:*

- *$A_{i,j}[\ell, r]$ is the value of an optimal solution for* Cluster Deletion *of the graph $G[V_{i,j}] - (C(\ell, r) \cup B_j(r))$ such that the vertices of $C(\ell, r)$ form a cluster.*

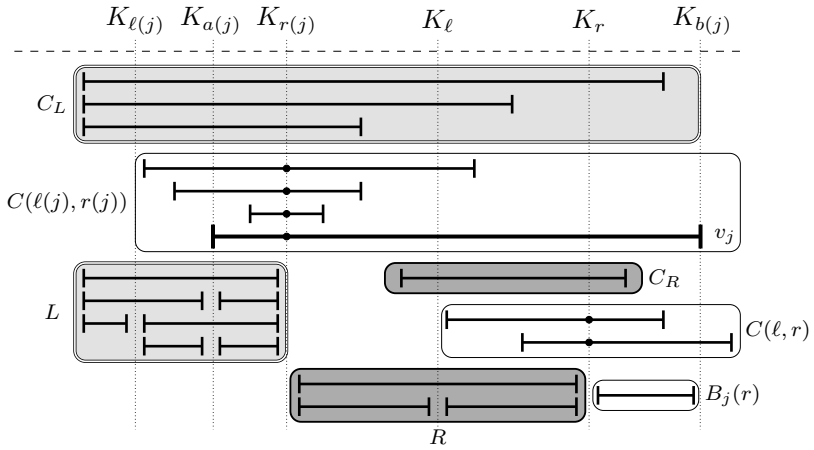

Figure 2: A partition of the set of vertices given in $A_{i,j}[\ell, r]$, where $V_L = C_L \cup L$ and $V_R = C_R \cup R$. Observe that $B_j(r(j)) = R \cup C_R \cup (C(\ell, r) \cap V_{i,j}) \cup B_j(r)$.

Hereafter, we assume that $B_j(t)$ with $t \geq b(v_j)$ corresponds to an empty set. Figure 2 illustrates a partition of the vertices with respect to $A_{i,j}[\ell, r]$. Notice that an optimal solution $A_{i,j}$ without any restriction is described in terms of $A_{i,j}[\ell, r]$ by $A_{i,j}[1, b(v_j) + 1]$, since no vertex of $V_{i,j}$ belongs to $C(1, b(v_j) + 1)$. Therefore, $A_{1,n}[1, n + 1]$ corresponds to the optimal solution of the whole graph G. As base cases, observe that if $V_{i,j}$ contains at most one vertex then $A_{i,j}[\ell, r] = 0$ for all bounding pairs (ℓ, r), since there are no internal edges. For a set C, we write $|C|_2$ to denote the number $\binom{|C|}{2}$. With the following result, we describe a recursive formulation for the optimal solution $A_{i,j}[\ell, r]$, which is our central tool for our dynamic programming algorithm.

Lemma 3.13. *Let (ℓ, r) be a bounding pair for v_j. Then,*

$$A_{i,j}[\ell, r] = \max_{(\ell(j), r(j)) \prec (\ell, r)} \left(A(V_L)[\ell(j), r(j)] + |C(\ell(j), r(j))|_2 + A(V_R)[\ell, r] \right),$$

where $V_L = V_{i,j} \setminus (C(\ell(j), r(j)) \cup B_j(r(j)))$ and $V_R = B_j(r(j)) \setminus (C(\ell, r) \cup B_j(r))$.

Proof. We first argue that $C(\ell(j), r(j))$ corresponds to the correct cluster C containing v_j. Observe that $v_j \notin C(\ell, r)$, because (ℓ, r) is a bounding pair for v_j, so that $a(v_j) < \ell$ whenever $a(v_j) \leq r \leq b(v_j)$ holds. By Lemmas 3.4 and 3.3, there are $r(j) = t$ and $\ell(j) = k$, where $a(v_j) \leq t \leq b(v_j)$ and $k = a\text{-}\min(K_t \cap C)$, such that $C = C(\ell(j), r(j))$. We show that such a set $C(\ell(j), r(j))$ is obtained from a correct choice among the described $(\ell(j), r(j))$. Assume first that $b(v_j) < r$. Then $A_{i,j}[\ell, r] = A_{i,j}$, because for every vertex u of $C(\ell, r)$ we know the $b(v_j) < b(u)$, so that $V_{i,j} \cap C(\ell, r) = \emptyset$. This means that $a(v_j) \leq r(j) \leq b(v_j)$ for every admissible pair $(\ell(j), r(j))$, as described in the given formula. Now assume that $r \leq b(v_j)$. Since v_j crosses (ℓ, r), Lemma 3.10 (4) shows that v_j is not contained in a cluster with a vertex y having $\ell < a(y)$. Thus, for any vertex $y \in C$ we know that $y \in K_t$ where $a(v_j) \leq t < \ell$. This means that there is a set $C(\ell(j), r(j))$ that contains exactly the vertices of C such that $a(v_j) \leq r(j) < \ell$. Therefore, $(\ell(j), r(j)) \prec (\ell, r)$ holds, as desired.

Next, we consider the sets V_L and V_R. We show that $A(V_L)[\ell(j), r(j)]$ and $A(V_R)[\ell, r]$ correctly store the optimal values of each part. To do so, we show first that the vertex sets of each part correspond to the correct sets and, then, each pair $(\ell(j), r(j))$ and (ℓ, r) is indeed a bounding pair for the last vertex of V_L and V_R, respectively. We start with some preliminary observations. Notice that $B_j(r) \subseteq B_j(r(j))$, because $r(j) < r$, which means that every vertex $B_j(r)$ does not belong to $V_L \cup V_R$. Since $C(\ell(j), r(j))$ contains only vertices of

$K_{r(j)}$ and $r(j) < \ell$, no vertex of $B_j(r)$ is considered in the described formula, as required in $A_{i,j}[\ell, r]$. By the properties of $C(\ell(j), r(j))$ and $C(\ell, r)$, we have the following:

- Let $x \in K_{r(j)} \cap V_{i,j}$. Then, either $x \in C(\ell(j), r(j))$ or x crosses the pair $(\ell(j), r(j))$. Moreover, if a vertex v crosses $(\ell(j), r(j))$ then $v \in V_L$.
- Let $y \in K_r \cap V_{i,j}$. Then, either $y \in C(\ell, r)$ or y crosses the pair (ℓ, r). Moreover, if a vertex v crosses (ℓ, r) but does not cross $(\ell(j), r(j))$ then $v \in V_R$.

Let C_L be the set of vertices of $V_{i,j}$ that cross $(\ell(j), r(j))$ and let C_R be the set of vertices of $V_{i,j} \setminus C_L$ that cross (ℓ, r). The previous properties imply that we can partition V_L to the vertices of C_L and the vertices of $V_{i,j}$ that belong to $L = (K_{a(i,j)} \cup \cdots \cup K_{r(j)-1}) \setminus K_{r(j)}$. Similarly, V_R is partitioned to the vertices of C_R and the vertices of $V_{i,j}$ that belong to $R = (K_{r(j)+1} \cup \cdots \cup K_{r-1}) \setminus (K_{r(j)} \cup K_r)$. See Figure 2 for an exposition of the corresponding sets. Thus, we have the following partitions for V_L and V_R:

- $V_L = C_L \cup L$, where $L = \big((K_{a(i,j)} \cup \cdots \cup K_{r(j)-1}) \setminus K_{r(j)}\big) \cap V_{i,j}$.
- $V_R = C_R \cup R$, where $R = \big((K_{r(j)+1} \cup \cdots \cup K_{r-1}) \setminus (K_{r(j)} \cup K_r)\big) \cap V_{i,j}$.

Let $v_{i'}, v_{j'}$ be the vertices of V_L with $i' = a\text{-}\min(V_L)$ and $j' = b\text{-}\max(V_L)$. We now show that $A(V_L)[\ell(j), r(j)]$ corresponds to the optimal solution of the graph $G[V_{i',j'}] - \big(B_{j'}(r(j)) \cup C(\ell(j), r(j))\big)$ such that the vertices of $C(\ell(j), r(j))$ form a cluster. Assume for contradiction that there is a vertex x of $V_{i',j'} \setminus \big(C(\ell(j), r(j)) \cup B_{j'}(r(j))\big)$ that does not belong to $V_L = V_{i,j} \setminus (B_j(r) \cup C(\ell, r))$. First notice that $K_{r(j)} \cap V_{i,j} = C(\ell(j), r(j))$ if and only if C_L is an empty set. In such a case, by Observation 3.8, we have $V_{i',j'} = V_{i,j} \setminus \big(K_{r(j)} \cup \cdots \cup K_{b(j)}\big)$, contradicting the existence of such a vertex x. Suppose that $v_{i'} \neq v_i$. Then $v_i \in M(j)$ or $v_i \in C(\ell(j), r(j))$, because $\min\{a(v_i), a(v_j)\}$ is the first maximal clique of all vertices of $V_{i,j}$. If $v_i \in M(j)$ then $U(j) = \emptyset$ and $\ell(j) = a(j)$. This means that for every $a(v_j) \leq r(j) \leq b(v_j)$, we have $K_{r(j)} \cap V_{i,j} = C(\ell(j), r(j))$, reaching a contradiction. If $v_i \in C(\ell(j), r(j))$ then $\ell(j) = a(v_i)$ and C_L is empty, reaching again a contradiction. Suppose now that $i' = i$. It is clear that $x \neq v_{j'}$. If $v_{j'} \in L$ then $C_L = \emptyset$, so that $K_{r(j)} \cap V_{i,j} = C(\ell(j), r(j))$. Assume that $v_{j'} \in C_L$. Now observe that if $x \in L \cup C_L$, then x is a vertex of $V_{i,j} \setminus (B_j(r) \cup C(\ell, r))$. Thus, $x \notin L \cup C_L$. If $b(x) < r(j)$ then $x \in L$ because $a(v_i) \leq a(x)$. This means that $r(j) \leq b(x)$. If $\ell(j) \leq a(x) \leq r(j)$ then $x \in C(\ell(j), r(j))$, leading to a contradiction that $x \in V_L$, and if $a(x) < \ell(j)$ then $x \in C_L$, leading to a contradiction that $x \notin L \cup C_L$. Thus, we know that $r(j) < a(x)$ and $b(x) \leq b(v_{j'})$. This, however, implies that $x \in B_{j'}(r(j))$, reaching a contradiction to the fact that $x \in V_{i',j'} \setminus B_{j'}(r(j))$. Therefore, we have shown that an optimal solution of the vertices of $V_{i',j'} \setminus \big(B_{j'}(r(j)) \cup C(\ell(j), r(j))\big)$ corresponds to an optimal solution of the vertices of V_L.

Furthermore, we argue that $(\ell(j), r(j))$ is a bounding pair for $v_{j'}$ in $A(V_L)[\ell(j), r(j)]$. Assume that $r(j) \leq b(v_{j'})$. If $r(j) \leq a(v_{j'})$ then $v_{j'} \in B_j(r(j))$, because $a(v_j) \leq r(j)$. As $v_{j'} \in V_L$, we have $a(v_{j'}) < r(j) \leq b(v_{j'})$. Then, if $\ell(j) \leq a(v_{j'})$, we get $v_{j'} \in C(\ell(j), r(j))$, which implies that $a(v_{j'}) < \ell(j)$, showing that $(\ell(j), r(j))$ is a bounding pair for $v_{j'}$. Assume next that $b(v_{j'}) < r(j)$. Then, $v_{j'} \notin C_L$, implying that $C_L = \emptyset$. Thus, for any value of $\ell(j)$ we know that $(\ell(j), r(j))$ is a bounding pair for $v_{j'}$. Therefore, $A(V_L)[\ell(j), r(j)]$ corresponds to the optimal solution of the graph $G[V_{i',j'}] - \big(B_{j'}(r(j)) \cup C(\ell(j), r(j))\big)$.

Next we consider the vertices of V_R, in order to show that $A(V_R)[\ell, r]$ corresponds to an optimal solution of the graph $G[V_R]$. Let $v_{i''}, v_{j''}$ be the vertices of V_R with $i'' = a\text{-}\min(V_R)$ and $j'' = b\text{-}\max(V_R)$. Assume for contradiction that there is a vertex x of $V_{i'',j''} \setminus \big(C(\ell, r) \cup B_{j''}(r)\big)$ that does not belong to $V_R = B_j(r(j)) \setminus (C(\ell, r) \cup B_j(r))$. Every vertex of $R \cup C_R$ belongs to V_R, so that $x \notin R \cup C_R$. This means that $b(x) > r$, since $x \notin R$, and $a(x) > r$, since $x \notin C_R \cup C(\ell, r)$. Then we obtain $r < a(x) \leq b(x) \leq b(v_{j''})$, showing that

$x \in B_{j''}(r)$. Thus we reach a contradiction, because $B_{j''}(r) \subseteq B_j(r)$. Hence, the vertices described in $A(V_R)[\ell, r]$ correspond to the vertices of V_R, as desired.

With respect to $A(V_R)[\ell, r]$, it remains to show that (ℓ, r) is a bounding pair for $v_{j''}$. If $b(v_{j''}) < r$ then $C_R = \emptyset$, which means that (ℓ, r) is a bounding pair for $v_{j''}$. Next suppose that $r \leq b(v_{j''})$. If $r \leq a(v_{j''})$ then $v_{j''} \in B_j(r)$, contradicting the fact that $v_{j''} \in V_R$. Thus, we know that $a(v_{j''}) < r \leq b(v_{j''})$. If further $\ell \leq a(v_{j''})$, then $v_{j''} \in C(\ell, r)$, contradicting $v_{j''} \in V_R$. Hence, we conclude that $v_{j''}$ crosses (ℓ, r), showing that (ℓ, r) is indeed a bounding pair for $v_{j''}$.

To complete the proof, observe that no vertex of V_L belongs to the same cluster with a vertex of V_R by Lemma 3.10 (3). Thus, the optimal solutions described by $A(V_L)[\ell(j), r(j)]$ and $A(V_R)[\ell, r]$ do not overlap in $A_{i,j}[\ell, r]$. Therefore, the claimed formula holds. □

Now we are ready to obtain our main result, namely a polynomial-time algorithm for Cluster Deletion on interval graphs.

Theorem 3.14. Cluster Deletion *is polynomial-time solvable on interval graphs.*

Proof. We describe a dynamic programming algorithm that computes $A_{1,n}$ based on Lemma 3.13. In a preprocessing step, we first compute two orderings of the vertices according to their first $a(v)$ and last $b(v)$ maximal cliques. Then we visit all vertices in ascending order with respect to $b(v_j)$ and for each such vertex v_j we consider the vertices v_i with $b(v_i) \leq b(v_j)$ in descending order with respect to $b(v_i)$. In such a way, we construct the sets $V_{i,j}$. We use a table $\mathcal{T}[i, j, \ell, r]$ to store the values of each $A_{i,j}[\ell, r]$. At the end, we output the maximum value of $\mathcal{T}[1, n, n+1, n+1]$ that corresponds to $A_{1,n}[n+1, n+1]$, as already explained. Regarding the running time, observe that the number of our table entries is at most n^4, as each table index is bounded by n. Moreover, computing a single table entry requires $O(n^2)$ time, since we take the maximum of at most (ℓ, r) table entries. Therefore, the overall running time of the algorithm is $O(n^6)$. □

4 Cluster Deletion on a generalization of split graphs (starlike graphs)

A graph $G = (V, E)$ is a *split graph* if V can be partitioned into a clique C and an independent set I, where (C, I) is called a *split partition* of G. Split graphs are characterized as $(2K_2, C_4, C_5)$-free graphs [10]. They form a subclass of the larger and widely known graph class of *chordal graphs*, which are the graphs that do not contain induced cycles of length 4 or more as induced subgraphs. In general, a split graph can have more than one split partition and computing such a partition can be done in linear time [17].

Hereafter, for a split graph G, we denote by (C, I) a split partition of G in which C is a maximal clique. It is known that Cluster Deletion is polynomial-time solvable on split graphs [3]. In fact, the algorithm given in [3] is characterized by its simplicity due to the following elegant characterization of an optimal solution: if there is a vertex $v \in I$ such that $N(v) = C \setminus \{w\}$ and w has a neighbor v' in I then the non-trivial clusters of an optimal solution are $C \setminus \{w\} \cup \{v\}$ and $\{w, v'\}$; otherwise, the only non-trivial cluster of an optimal solution is C [3]. Here we study whether such a simple characterization can be extended into more general classes of split graphs. Due to Lemma 2.1, it is natural to consider true twins of $V \setminus C$, as they are grouped together in an optimal solution and they are expected not to influence the solution characterization[1]. Surprisingly, we show that Cluster Deletion remains NP-complete even on such a slight generalization of split graphs. Before presenting

[1]Note that the class of split graphs is closed under the addition of true twins in the clique.

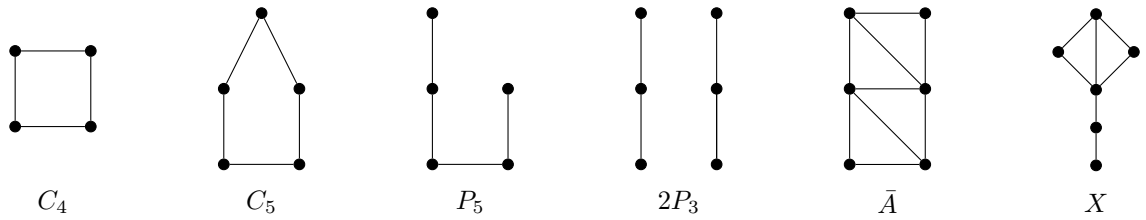

Figure 3: The list of forbidden induced subgraph characterization for starlike graphs.

our NP-completeness proof, let us first show that such graphs form a proper subclass of P_5-free chordal graphs. We start by giving the formal definition of such graphs that were introduced in [6].

Definition 4.1. *A graph $G = (V, E)$ is called* starlike *graph if its vertex set can be partitioned into C and I such that $G[C]$ is a clique and every two vertices of I are either non-adjacent or true twins in G.*

It is clear that in a starlike graph G with vertex partition (C, I) the following holds:

(i) each connected component of $G[I]$ is a clique and forms a true-twin set in G, and

(ii) contracting the connected components of $G[I]$ results in a split graph, denoted by G^*.

Starlike graphs are exactly the intersection graphs of subtrees of a star [6]. In fact, a forbidden induced subgraph characterization was already given in [6]; figure 3 illustrates the induced subgraphs that are forbidden in a starlike graph. Despite the known forbidden subgraph characterization, here we give a shorter and different proof of such a characterization.

Proposition 4.2. *A graph G is starlike if and only if it does not contain any of the graphs $C_4, C_5, P_5, 2P_3, \bar{A}, X$ as induced subgraphs.*

Proof. Let $\mathcal{F}$ be the list of such subgraphs, i.e., $\mathcal{F} = \{C_4, C_5, P_5, 2P_3, \bar{A}, X\}$. We show that starlike graphs are exactly the $\mathcal{F}$-free graphs. It is clear that any subgraph of $\mathcal{F}$ does not contain true twins. Moreover, each subgraph of $\mathcal{F} \setminus \{C_4, C_5\}$ contains an induced $2K_2$, which implies that all such subgraphs of F are not starlike graphs. Thus, if a graph G contains one of the subgraphs of $\mathcal{F}$ then G is not a starlike graph.

We show that any $\mathcal{F}$-free graph G is starlike. If G is a split graph then, by definition, G is starlike. Assume that G is not a split graph. Since G does not contain C_4 or C_5 and split graphs are exactly the $(2K_2, C_4, C_5)$-free graphs, there is an induced $2K_2$ in G. Let x_1x_2 and y_1y_2 be the two edges of an induced $2K_2$. We show that the endpoints of at least one of the two edges are true twins. Assume for contradiction that neither x_1, x_2 nor y_1, y_2 are true twins in G. Let a be a neighbor of x_1 that is non-adjacent to x_2, and let b be a neighbor of y_1 that is non-adjacent to y_2. We show that the vertices of $\{a, x_1, x_2, b, y_1, y_2\}$ induce one of the subgraphs of $\mathcal{F}$, contradicting the fact that no pair of vertices form true twins. If $b \notin N(\{x_1, x_2\})$ and $a \notin N(\{y_1, y_2\})$ then there is an induced P_5 or $2P_3$ depending on whether a and b are adjacent or not. Thus, $b \in N(\{x_1, x_2\})$ or $a \in N(\{y_1, y_2\})$. Observe that if a is adjacent to at least one of y_1 or y_2 then a is adjacent to both y_1 and y_2; otherwise, $\{x_1, x_2, a, y_1, y_2\}$ induce a P_5. By symmetric arguments we know that either b is adjacent to both x_1, x_2 or to none. Without loss of generality, assume that $bx_1, bx_2 \in E(G)$.

- Suppose that a and b are non-adjacent. If $a \notin N(\{y_1, y_2\})$ then there is a P_5 induced by $\{a, x_1, b, y_1, y_2\}$. Moreover, by the previous argument, we know that if $a \in N(\{y_1, y_2\})$ then $ay_1, ay_2 \in E(G)$, which implies a C_4 in G induced by $\{a, x_1, b, y_1\}$. Thus if $ab \notin E(G)$ we obtain a induced subgraph of F.

- Suppose that a and b are adjacent. If $a \notin N(\{y_1, y_2\})$, then all six vertices induce an X graph. Otherwise, we know that $ay_1, ay_2 \in E(G)$, showing that all six vertices induce a graph $\bar{A}$, where a and b are the degree four vertices.

Thus in all cases we obtain an induced subgraph of $\mathcal{F}$, reaching to a contradiction that G being an $\mathcal{F}$-free graph. This means that for any $2K_2$ we know that at least one of the two edges contains true twin vertices in G. By iteratively picking such true twins and contracting them into a new vertex, results in a graph G^* that does not contain $2K_2$. Therefore G^* is a split graph, implying that G is a starlike graph. □

Thus by Proposition 4.2, starlike graphs form a proper subclass of P_5-free chordal graphs, i.e., of (C_4, C_5, P_5)-free graphs. Now let us show that decision version of Cluster Deletion is NP-complete on starlike graphs. This is achieved by observing that the constructed graphs given in the reduction for P_5-free graphs [3], constitute such split-related graphs. In particular, the reduction shown in [3] comes from the X3C problem: given a universe X of $3q$ elements and a collection $C = \{C_1, \ldots, C_{|C|}\}$ of 3-element subsets of X, asks whether there is a subset $C' \subseteq C$ such that every element of X occurs in exactly one member of C'. The constructed graph G is obtained by identifying the elements of X as a clique K_X and there are $|C|$ disjoint cliques $K_1, \ldots, K_{|C|}$ each of size $3q$ corresponding to the subsets of C and a vertex x of K_X is adjacent to all the vertices of K_i if and only if x belongs to the corresponding subset C_i of K_i. Then, it is not difficult to see that the vertices of each K_i are true twins and the contracted graph G^* is a split graph, showing that G is indeed a starlike graph. Therefore, by the NP-completeness given in [3], we have:

Theorem 4.3. Cluster Deletion *is NP-complete on starlike graphs.*

However, here we give a different reduction that highlights an interesting connection between edge-weighted and vertex-weighted split graphs. In the Edge Weighted Cluster Deletion problem, each edge of the input graph is associated with a weight and the objective is to construct a clustered graph having the maximum total (cumulative) weight of edges. As already explained, we can contract true twins and obtain a vertex-weighted graph as input for the corresponding Cluster Deletion. Similarly, it is known that for edge-weighted graphs the corresponding Edge Weighted Cluster Deletion remains NP-hard even when restricted to particular variations on special families of graphs [3]. In fact, it is known [3] that Edge Weighted Cluster Deletion remains NP-hard on split graphs even when

(i) all edges inside the clique have weight one,

(ii) all edges incident to a vertex $w \in I$ have the same weight q, and

(iii) $q = |C|$.

We abbreviate the latter problem by EWCD and denote by (C, I, k) an instance of the problem where (C, I) is a split partition of the vertices of G and k is the total weight of the edges in a cluster solution for G. With the following result, we show an interesting connection between the two variations of the problem when restricted to starlike graphs.

Theorem 4.4. *There exists a polynomial time algorithm that, given an instance (C, I, k) for EWCD, produces an equivalent instance for* Cluster Deletion *on starlike graphs.*

Proof. Let (C, I, k) be an instance of EWCD, where $G = (C \cup I, E)$ is a split graph. From G, we build a starlike graph $G' = (C' \cup I', E')$ by keeping the same clique $C' = C$, and for every vertex $w_j \in I$ we apply the following:

- We replace w_j by $q = |C|$ true twin vertices I'_j (i.e., by a q-clique) such that for any vertex $w' \in I'_j$ we have $N_{G'}(w') = N_G(w_j) \cup (I'_j \setminus \{w'\})$. That is, their neighbors outside I'_j are exactly $N_G(w_j)$. Moreover, the set of vertices $I'_1, \ldots, I'_{|I|}$ form I'.

By the above construction, it is not difficult to see that G' is a starlike graph, since the graph induced by I' is a disjoint union of cliques and two adjacent vertices of I' are true twins in G'. Also observe that the construction takes polynomial time because q is at most $n = |V(G)|$. We claim that there is an edge weighted cluster solution for G with total weight at least k if and only if there is a cluster solution for G' having at least $k + |I| \cdot \binom{q}{2}$ edges.

Assume that there is a cluster solution S for G with total weight at least k. From S, we construct a solution S' for G'. There are three types of clusters in S:

(a) Cluster formed only by vertices of the clique C, i.e., $Y \in S$, where $Y \subseteq C$. We keep such clusters in S'. We denote by t_a the total weight of clusters of type (a). Notice that since the weight of edges having both endpoints in C are all equal to one, t_a corresponds to the number of edges in Y.

(b) Cluster formed only by one vertex $w_j \in I$, i.e., $\{w_j\} \in S$. In S' we replace such cluster by the corresponding clique I'_j having exactly $\binom{q}{2}$ edges. It is clear that the total weight of such clusters do not contribute to the value of S.

(c) Cluster formed by the vertices $y_1, \ldots, y_p, w_j$, where $y_i \in C$ and $w_j \in I$. As the weights of the edges between the vertices of y_i is one, the total number of weights in such a cluster is $\binom{p}{2} + p \cdot q$. Let t_c be the total weight of clusters of type (c). In S' we replace w_j by the vertices of I'_j and obtain a cluster S' having $\binom{p}{2} + p \cdot q + \binom{q}{2}$ number of edges.

Now observe that in S we have $t_a + t_c$ total weight, which implies $t_a + t_c \geq k$. Thus, in S' we have at least $t_a + t_c + |I| \cdot \binom{q}{2}$ edges, giving the desired bound.

For the opposite direction, assume that there is a cluster solution S' of G' having at least $k + |I| \cdot \binom{q}{2}$ edges. All vertices of I'_j are true twins and, by Lemma 2.1, we know that they belong to the same cluster in S'. Thus, any cluster of S' has one of the following forms:

(i) Y', where $Y' \subseteq C'$,

(ii) I'_j,

(iii) $I'_j \cup \{y'_1, \ldots, y'_p\}$, where $y'_i \in C'$.

This means that all internal edges having both endpoints in I' contribute to the value of S' by $|I| \cdot \binom{q}{2}$. Moreover, observe that for any internal edge of S' of the form $y'w'$ with $y' \in C'$ and $w' \in I'_j$, we know that there are exactly q internal edges incident to y' and the q vertices of I'_j. Thus, internal edges $y'w'$ of S' correspond to exactly one internal edge yw_j of S having weight q, where $y = y'$ (recall that $C = C'$) and w_j is the vertex of I associated with I_j. Hence, all internal edges outside each I'_j in S' correspond to either a weighted internal edge in S or to the same unweighted edge of the clique C in S. Therefore, there is an edge weighted solution S having weight at least k. □

4.1 Polynomial-time algorithms on subclasses of starlike graphs

Due to the hardness result given in Theorem 4.4, it is natural to consider subclasses of starlike graphs related to their analogue subclasses of split graphs. We consider two such subclasses. The first one corresponds to the starlike graphs in which the vertices of I have no common neighbor in the clique, unless they are true or false twins. The second one is related to the true twin extension of *threshold graphs* (i.e., split graphs in which the vertices

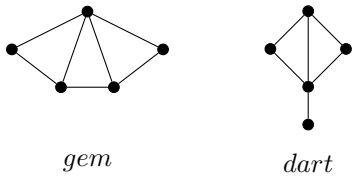

Figure 4: Forbidden induced subgraphs of stable-like graphs that are starlike graphs.

of the independent set have nested neighborhood) and form the starlike graphs in which the vertices of the I have nested neighborhood. The third one comprises a generalization of the formers and consists of the starlike graphs that are obtained from vertex-disjoint threshold graphs with a common clique. We formally define such graphs and give polynomial-time algorithms for Cluster Deletion on the considered graph classes. For a vertex $x \in I$ we write $N_C(x)$ to denote the set $N(x) \cap C$ and for a vertex $a \in C$ we write $N_I(a)$ to denote the set $N(a) \cap I$.

Definition 4.5. *A starlike graph G with partition (C, I) on its vertices is called* stable-like *graph if*

- $\forall x, y \in I$*: either* $N_C(x) \cap N_C(y) = \emptyset$ *or* $N_C(x) = N_C(y)$.

It is not difficult to see that in a stable-like graph, any two vertices of I having a common neighbor in C have exactly the same neighborhood in C. Before presenting our linear-time algorithm, we first give a forbidden induced subgraph characterization for the class of stable-like graphs. The graphs *gem* and *dart* are shown in Figure 4.

Proposition 4.6. *A graph G is stable-like if and only if it does not contain any of the graphs $C_4, C_5, P_5, 2P_3, gem, dart$ as induced subgraphs.*

Proof. We first show that if G is stable-like then it does not contain any graph of the given list as an induced subgraph. Since G is a starlike graph, by Proposition 4.2 G does not contain any of $C_4, C_5, P_5, 2P_3$ as induced subgraphs. Moreover, it is not difficult to see that in any proper partition (C, I) of a *gem* or a *dart* there are no two vertices $x, y \in I$ for which $N_C(x) \cap N_C(y) = \emptyset$ or $N_C(x) = N_C(y)$. Thus, the claimed list is indeed forbidden for stable-like graphs.

For the opposite direction, we show that any starlike graph that is not stable-like contains a *gem* or a *dart* as an induced subgraph. Then, by Proposition 4.2 we obtain the claimed list of forbidden induced subgraphs. Let G be a starlike graph that is not stable-like, with partition (C, I) such that $|C|$ is maximum. By definition, we know that there are two vertices $x, y \in I$ such that $N_C(x) \cap N_C(y) \neq \emptyset$ and $N_C(x) \neq N_C(y)$.

- Assume that $N_C(x) \not\subseteq N_C(y)$ and $N_C(y) \not\subseteq N_C(x)$. Let $a \in N_C(x) \setminus N_C(y)$, $b \in N_C(y) \setminus N_C(x)$, and $c \in N_C(x) \cap N_C(y)$. Notice that all three vertices exist because of our assumptions. Then, $xy \notin E(G)$ because there is no C_4 in a starlike graph which means that the vertices of $\{x, y, a, b, c\}$ induce a *gem* in G.

- Assume that $N_C(x) \subsetneq N_C(y)$. There are two vertices $b, c \in C$ such that $b \in N_C(y) \setminus N_C(x)$ and $c \in N_C(x) \cap N_C(y)$. We show that y is non-adjacent to all the vertices of C. For this, observe that if $C \subseteq N_C(y)$ then $(C \cup y, I \setminus \{y\})$ is a partition of the vertices of G that respects Definition 4.1 and properties (i) and (ii). By the maximality of C, we obtain that there is a vertex $z \in C$ such that $z \notin N_C(y)$. Since $N_C(x) \subsetneq N_C(y)$, we know that $z \notin N_C(x)$. Thus the vertices of $\{x, y, b, c, z\}$ induce a *gem* whenever $xy \in E(G)$ or a *dart* whenever $xy \notin E(G)$.

Therefore, in all cases we obtain a *gem* or a *dart* as an induced subgraph. □

Theorem 4.7. Cluster Deletion *can be solved in time $O(n+m)$ for a stable-like graph on n vertices and m edges.*

Proof. Let G be a stable-like graph with partition (C, I). First observe that if G is disconnected then I contains isolated cliques, i.e., true twins having no neighbor in C. Thus we can restrict ourselves to a connected graph G, since by Lemma 2.1 each isolated clique is contained in exactly one cluster of an optimal solution. We now show that all vertices of C that have a common neighbor in I are true twins. Let u and v be two vertices of C such that $x \in N(u) \cap N(v) \cap I$. All vertices of $C \setminus \{u, v\}$ are adjacent to both u and v. Assume that there is a vertex $y \in I$ that is adjacent to u and non-adjacent to v. If $xy \in E(G)$ then by the definition of starlike graphs x and y are true twins which contradicts the assumption of $xv \in E(G)$ and $yv \notin E(G)$. Otherwise, x and y are non-adjacent and since $N_C(x) \cap N_C(y) \neq \emptyset$ we reach a contradiction to the definition of stable-like graphs. Thus, all vertices of C that have a common neighbor in I are true twins.

We partition the vertices of C into true twin classes $C_1, \ldots, C_k$, such that each C_i contains true twins of C. From the previous discussion, we know that any vertex of I is adjacent to all the vertices of exactly one class C_i; otherwise, there are vertices of different classes in C that have common neighbor. For a class C_i, we partition the vertices of $N(C_i) \cap I$ into true twin classes $I_i^1, \ldots, I_i^q$ such that $|I_i^1| \geq \cdots \geq |I_i^q|$.

We claim that in an optimal solution S, the vertices of each class I_i^j with $j \geq 2$ constitute a cluster. To see this, observe first that the vertices of I_i^j, $1 \leq j \leq q$, are true twins, and by Lemma 2.1 they all belong to the same cluster of S. Also, by Lemma 2.1 we know that all the vertices of C_i belong to the same cluster of S. Moreover, all vertices between different classes I_i^j,$I_i^{j'}$ are non-adjacent and are C_i-compatible. Since every vertex of I_i^j is non-adjacent to all the vertices of $V(G) \setminus (I_i^j \cup C_i)$, we know that any cluster of S that contains I_i^j is of the form either $I_i^j \cup C_i$ or I_i^j. Assume that there is a cluster that contains $I_i^j \cup C_i$ with $j \geq 2$. Then, we substitute the vertices of I_i^j by the vertices of I_i^1 and obtain a solution of at least the same size, because $|I_i^1| \geq |I_i^j|$ implies $\binom{|C_i|+|I_i^1|}{2} \geq \binom{|C_i|+|I_i^j|}{2}$. Thus, all vertices of each class I_i^j with $j \geq 2$ constitute a cluster in an optimal solution S.

This means that we can safely remove the vertices of I_i^j with $j \geq 2$, by constructing a cluster that contains only I_i^j. Hence, we construct a graph G^* from G, in which there are only matched pair of k classes (C_i, I_i) such that (i) all sets C_i, I_i are non-empty except possibly the set I_k, (ii) $N(C_i) \cap I = I_i$, (iii) $N(I_i) = C_i$, (iv) $G^*[C_i \cup I_i]$ is a clique, and (v) $G^*[C_1 \cup \cdots \cup C_k]$ is a clique. Our task is to solve Cluster Deletion on G^*, since for the rest of the vertices we have determined their cluster. By Lemma 2.1, observe that if the vertices of $C_i \cup C_j$ belong to the same cluster then the vertices of each I_i and I_j constitute two respectively clusters. Thus, for each set of vertices I_i we know that either one of $C_i \cup I_i$ or I_i constitutes a cluster in S. This boils down to compute a set M of matched pairs (C_i, I_i) from the k classes, having the maximum value

$$\sum_{(C_i,I_i)\in M} \binom{|C_i|+|I_i|}{2} + \binom{\sum_{C_j \notin M} |C_j|}{2} + \sum_{I_j \notin M} \binom{|I_j|}{2}.$$

Let (C_i, I_i) and (C_j, I_j) be two pairs of classes such that $|C_i| + |I_i| \leq |C_j| + |I_j|$. We show that if $(C_j, I_j) \notin M$ then $(C_i, I_i) \notin M$. Assume for contradiction that $(C_j, I_j) \notin M$ and $(C_i, I_i) \in M$. Observe that $|I_j| < \sum_{C_t \notin M \setminus C_j} |C_t|$, because I_j is C_j-compatible. Similarly, we know that $\sum_{C_t \notin M \setminus C_j} |C_t| + |C_j| \leq |I_i|$. This however, shows that $|C_j| + |I_j| < |I_i|$, contradicting the fact that $|C_i| + |I_i| \leq |C_j| + |I_j|$. Thus $(C_j, I_j) \notin M$ implies $(C_i, I_i) \notin M$.

This means that we can consider the k pair of classes (C_i, I_i) in a decreasing order according to their number of vertices $|C_i| + |I_i|$. With a simple dynamic programming algorithm, starting from the largest ordered pair (C_1, I_1) we know that either (C_1, I_1) belongs to M or not. In the former, we add $\binom{|C_1|+|I_1|}{2}$ to the optimal value of $(C_2, I_2), \ldots, (C_k, I_k)$ and in the latter we know that no pair belongs to M giving a total value of $\binom{\sum |C_i|}{2} + \sum \binom{|I_i|}{2}$. By choosing the maximum between the two values, we construct a table of size k needed for the dynamic programming. Computing the twin classes and the partition (C, I) takes linear time in the size of G and sorting the pair of classes can be done $O(n)$ time, since $\sum(|C_i| + |I_i|)$ is bounded by n. Thus, the total running time is $O(n + m)$, as the dynamic programming for computing M requires $O(n)$ time. Therefore, all steps can be carried out in linear time for a stable-like graph G. □

We next define the analogue of threshold graphs in terms of starlike graphs.

Definition 4.8. *A starlike graph G with partition (C, I) on its vertices is called* threshold-like *graph if*

- $\forall x, y \in I$: $N_C(x) \subseteq N_C(y)$.

It is not difficult to see that the class of threshold-like graphs and stable-like graphs are unrelated. Threshold-like graphs are also known as *starlike-threshold graphs* under the notions of intersection graphs [6]. Although the absence of an induced P_4 follows from the results of [6], we give the following short proof for completeness.

Lemma 4.9. *Let G be a threshold-like graph. Then G is a P_4-free graph.*

Proof. We show that there is no induced path on four vertices, P_4, in G. Assume for contradiction that there is a $P_4 = v_1v_2v_3v_4$ in G. Since $G[C]$ is a clique and $G[I]$ is a disjoint union of cliques, at least one of v_1, v_4, say v_1, belongs to I. If $v_4 \in C$ then $v_2 \in I$ because $v_4v_2 \notin E(G)$, which gives a contradiction as $v_1v_2 \in E(G)$ and v_1, v_2 are not true twins. Otherwise, we have $v_4 \in I$, so that $v_2, v_3 \in C$ because v_1, v_2 and v_3, v_4 are not true twins G. The latter, results again in a contradiction because $N_C(v_1) \nsubseteq N_C(v_4)$ and $N_C(v_4) \nsubseteq N_C(v_1)$. Therefore, G is a P_4-free graph. □

By Lemma 4.9 and the $O(n^2)$-time algorithm on P_4-free graphs (also known as *cographs*) [12, 23], Cluster Deletion is polynomial-time solvable on threshold-like graphs.

Next we proceed with a subclass of starlike graphs that generalizes the previous two classes, as it contains both the class of stable-like graphs and the class of threshold-like graphs.

Definition 4.10. *A starlike graph G with partition (C, I) on its vertices is called* laminar-like *graph*[2] *if*

1. *$\forall x, y \in I$: either $N_C(x) \cap N_C(y) = \emptyset$ or $N_C(x) \subseteq N_C(y)$, and*
2. *$\forall a, b \in C$: either $N_I(a) \cap N_I(b) = \emptyset$ or $N_I(a) \subseteq N_I(b)$.*

We start by characterizing the laminar-like graphs in terms of disjoint threshold-like graphs.

Lemma 4.11. *A graph $G = (V, E)$ is a laminar-like graph with partition (C, I) if and only if $V(G)$ can be partitioned into vertex-disjoint threshold-like graphs $G_i = (C_i \cup I_i, E_i)$ such that $C = \cup C_i$, $I = \cup I_i$, and $E(G) = E(C) \cup (\cup E_i)$.*

[2]The term *laminar* comes from the notion of laminar family of sets: a family of sets is called *laminar* if any two of its sets are either disjoint or one includes the other.

Proof. Given a laminar-like graph G with partition (C, I), we partition the vertices of G according to whether the vertices of I have a common neighbor in C. Let I_i be a subset of I that contains all vertices $x, y \in I$ such that $N_C(x) \cap N_C(y) \neq \emptyset$ or $N_C(x) = N_C(y)$. Let also $C_i = N_C(I_i)$. We claim that $G_i = G[C_i \cup I_i]$ is a threshold-like graph. By the first property of Definition 4.10, for any two vertices $x, y \in I_i$ we have $N_C(x) \subseteq N_C(y)$. Let $z \in I_i$. Then $N_C(z) \subseteq N_C(y)$ by the construction of I_i. Assume for contradiction that $N_C(x) \nsubseteq N_C(z)$ and $N_C(z) \nsubseteq N_C(x)$. Let $a \in N_C(x) \setminus N_C(z)$ and $b \in N_C(z) \setminus N_C(x)$. Then observe that $y \in N_I(a) \cap N_I(b)$. Thus by the second property of Definition 4.10 we reach a contradiction to $N_I(a) \subseteq N_I(b)$. Therefore the vertices of I_i can be ordered as $w_1, \ldots, w_{|I_i|}$ such that $N_C(w_1) \subseteq \cdots \subseteq N_C(w_{|I_i|})$ which means that G_i is indeed a threshold-like graph. Moreover consider any two subgraphs $G_i = (C_i, I_i)$ and $G_j = (C_j, I_j)$ that are constructed as explained above. Then it is clear that $I_i \cap I_j = \emptyset$ and $C_i \cap C_j = \emptyset$, since the construction partitions I into equivalent classes of I. In particular, for every two vertices $w \in I_i$ and $w' \in I_j$ we have $N_C(w) \cap N_C(w') = \emptyset$. What is left to show is that there are no edges between the vertices of I_i and I_j. For this, observe that if there is an edge between $w \in I_i$ and $w' \in I_j$ then w and w' are true twins, as G is a starlike graph. Therefore we have $N_C(w) = N_C(w')$ which means that both w, w' belong to the same set I_i.

For the opposite direction, assume that we are given vertex-disjoint threshold-like graphs $G_i = (C_i \cup I_i, E_i)$. We consider the graph G obtained from the union of G_i by adding all edges among the vertices of $\cup C_i$. As there are all the edges among the vertices of C_i and C_j, we have that $C = \cup C_i$ is a clique. Moreover, each class of true twins of I_i remains a class of true twins in G. Thus G is starlike graph. We show that G is indeed a laminar-like graph by verifying the two properties of Definition 4.10. For any two vertices $x, y \in I_i$ we have $N_C(x) \subseteq N_C(y)$ by Definition 4.8. If $x \in I_i$ and $y \in I_j$ then $N_C(x) \cap N_C(y) = \emptyset$, since there are no edges between the vertices of I_i and I_j. Similarly, for any two vertices $a, b \in C_i$ we have $N_{I_i}(a) \subseteq N_{I_i}(b)$ which means that $N_I(a) \subseteq N_I(b)$ because every vertex of $G - G_i$ is either adjacent to both a and b or non-adjacent to both a and b. Moreover, for two vertices $a \in C_i$ and $b \in C_j$, we have $N_I(a) \cap N_I(b) = \emptyset$ because $I_i \cap I_j = \emptyset$ and both a and b are adjacent to every vertex of C. Therefore G is a laminar-like graph. □

We next show a polynomial-time algorithm for solving Cluster Deletion on laminar-like graphs which form the more general subclass of the considered subclasses of starlike graphs. Towards this, we apply Lemma 4.11, obtain an optimal solution in each G_i, and then apply the algorithm given in Theorem 4.7.

Theorem 4.12. Cluster Deletion *can be solved in time $O(n^2)$ for a laminar-like graph on n vertices.*

Proof. Let G be a laminar-like graph. We first compute the true twin classes and the partition (C, I) of G which can be done in linear time. By the true twin classes and Lemma 4.11, we compute the threshold-like induced subgraphs G_i of G. For doing so, all vertices of I, denoted by I_i, having a common neighbor in C belong to the same graph G_i, whereas all vertices of I having no neighbor in C belong to the same graph, that we denote by G_0. Observe that the vertices of I_i define the set $C_i = N_C(I_i)$. Moreover, all adjacent vertices of I_0 are true twins in G and $N_C(I_0) = \emptyset$. Thus each connected component of G_0 is already a clique in G and forms a cluster in any optimal solution.

Consider a threshold-like graph G_i with partition (C_i, I_i). To ease the notation, we let $H = G_i$ and (A, B) be the partition (C_i, I_i). By Lemma 4.9, H is a P_4-free graph. For P_4-free graphs, it is known that greedily selecting maximum cliques results in an optimal solution for Cluster Deletion [12]. Let $S(H) = (S_1, \ldots, S_k)$ be the clusters of an optimal solution of H such that S_i is a maximum clique of the graph $H - (S_1 \cup \cdots \cup S_{i-1})$, for $1 \leq i \leq k$ with $S_0 = \emptyset$. We call $S(H)$ a *greedy-optimal* solution of H. Observe that all true

twins of H belong to the same cluster S_i by the greedy choice of a maximum clique. We partition the vertices of each cluster $S_i \in S(H)$ with respect to (A, B). In particular, for every $1 \leq i \leq k$, we define $A_i = S_i \cap A$ and $B_i = S_i \cap B$. Due to the construction of H, in which $N_C(B) = A$, notice that all sets B_i are non-empty, whereas a set A_i may be empty. We prove the following claim.

Claim 4.13. *Let $S(H) = (S_1, \ldots, S_k)$ be a greedy-optimal solution of the threshold-like graph H. For every $S_i = (A_i, B_i)$, $1 \leq i \leq k$, the following hold:*

1. *B_i constitutes a class of true twins.*
2. *For every $i < j \leq k$ with $S_j = (A_j, B_j)$, we have $|A_i| + |B_i| \geq |W| + |B_j|$, where $W = (A_i \cup \cdots \cup A_k) \cap N_H(B_j)$.*
3. *Removing all edges with one endpoint in A_i and the other endpoint in B_j, for $j \neq i$, results in a stable-like graph.*

Proof: Let $C(H) = A_1 \cup \cdots \cup A_k$ and $I(H) = B_1 \cup \cdots \cup B_k$. For the first statement, observe that every pair of adjacent vertices in $I(H)$ are true twins since H is a starlike graph. Thus, by Lemma 2.1, every set B_i constitutes a class of true twins.

For the second statement, let H' be the graph obtained from H by removing the vertices of $(A_1, B_1), \ldots, (A_{i-1}, B_{i-1})$. As (A_i, B_i) is a maximum clique in H' by the greedy choice for S_i, $|A_i| + |B_i|$ is greater or equal than the size of any other (maximal) clique in H'. Any maximal clique containing B_j in H', consists of B_j together with adjacent vertices of $A_i \cup \cdots \cup A_k$. Therefore the second statement follows.

For the third statement we consider the graph H'' with vertex set $C(H) \cup I(H)$ and edge set formed by making $C(H)$ and each (A_i, B_i) a clique, for $1 \leq i \leq k$. In order to show that H'' is a stable-like graph, observe that every pair of adjacent vertices in $I(H)$ are true twins since they belong to the same set B_i. Thus H'' is a starlike graph. To conclude, we need to prove that for any two vertices x, y of $I(H)$ either $N_{C(H)}(x) \cap N_{C(H)}(y) = \emptyset$ or $N_{C(H)}(x) = N_{C(H)}(y)$. If $x, y \in B_i$ then $N_{C(H)}(x) = N_{C(H)}(y) = A_i$ by construction, and if $x \in B_i$ and $y \in B_j$ then $N_{C(H)}(x) = A_i$ and $N_{C(H)}(y) = A_j$ so that $A_i \cap A_j = \emptyset$. Therefore, H'' is indeed a stable-like graph. $\diamondsuit$

Next we show that there is an optimal solution for G that *respects* the internal clusters of a greedy-optimal solution of H. That is, every cluster (A_i, B_i) in H remains a cluster in G, or is split into two clusters $A_i \cup Z$ and B_i where Z is a set of vertices of C.

Claim 4.14. *Let $S(H) = (S_1, \ldots, S_k)$ be a greedy-optimal solution of H and let $S_i = (A_i, B_i)$ be a cluster of $S(H)$, $1 \leq i \leq k$. There is an optimal solution $S(G)$ of G such that either $A_i \cup B_i \in S(G)$, or $A_i \cup Y, B_i \in S(G)$, where $Y \subseteq C$.*

Proof: Let (C, I) be the partition of the vertices of G into a clique C and a union of cliques I. Let also $C(H) = A_1 \cup \cdots \cup A_k$ and $I(H) = B_1 \cup \cdots \cup B_k$. Recall that every vertex of B_i is non-adjacent to any vertex of $G - H$, whereas every vertex of A_i is adjacent to every vertex of C. Thus for any vertex $z \in V(G) \setminus V(H)$ we know that the vertices $\{z\} \cup A_i \cup B_i$ do not induce a clique in G. If there is no cluster of $S(G)$ that contains vertices of both $G - H$ and H, then every cluster (A_i, B_i) of H is a cluster of G, since $S(H)$ is an optimal solution of H. In what follows, we assume that a set Z of vertices of $G - H$ together with a set X of vertices of H constitutes a cluster in $S(G)$. It is clear $Z \subseteq C \setminus C(H)$ and $X \subseteq C(H)$. Observe also that all the vertices of Z are adjacent to every vertex of $C(H) = A_1 \cup \cdots \cup A_k$ and non-adjacent to any vertex of $I(H) = B_1 \cup \cdots \cup B_k$ by Lemma 4.11. First we claim that there is no cluster $Z' \cup X'$ in $S(G)$ with $Z' \subseteq C \setminus (V(H) \cup Z)$ and $X' \subseteq C(H) \setminus X$. To see this, notice that all the vertices of Z are $(Z' \cup X')$-compatible and all the vertices of Z' are $(Z \cup X)$-compatible which by Lemma 2.3 is not possible.

We consider the graph $F = G[Z \cup V(H)]$. It is not difficult to see that F is a threshold-like graph, since H is a threshold-like graph and all the vertices of Z are adjacent to every vertex of $C(H)$ and non-adjacent to any vertex of $I(H)$. In particular, $(C(H), I(H) \cup Z)$ is a partition of the vertices of F where the vertices of Z are true twins. Let S_ℓ be the cluster of $S(H)$ with the smallest $1 \leq \ell \leq k$ such that $|A_\ell| + |B_\ell| < |Z| + |A_\ell| + \cdots + |A_k|$. We show that there is a greedy-optimal solution $(S'_1, \ldots, S'_{k+1})$ of F such that:

- $S'_i = S_i$, for every $1 \leq i \leq \ell - 1$,
- $S'_\ell = (A_\ell \cup \cdots \cup A_k, Z)$, and
- $S'_{j+1} = (\emptyset, B_j)$, for every $\ell \leq j \leq k$.

For this, observe that for any S_i, $1 \leq i \leq \ell - 1$, we have $|A_i| + |B_i| \geq |Z| + |A_i| + \cdots + |A_k|$ by the choice of S_ℓ. Thus, by Claim 4.13 (2), S_i is a maximum clique of $F - (S_1 \cup \cdots \cup S_{i-1})$, so that $S'_i = S_i$. Due to the greedy choice and Claim 4.13 (2), we know that the described S'_ℓ is the maximum clique of $F - (S_1 \cup \cdots \cup S_{\ell-1})$. Thus S'_ℓ is indeed a cluster of a greedy-optimal solution of F. Moreover all vertices of B_j, $\ell \leq j \leq k$, form true twins by Claim 4.13 (1). Since the vertices of each B_j have no neighbors in $F - (S'_1 \cup \cdots \cup S'_\ell \cup B_j)$, every B_j constitutes a cluster. Therefore there is a greedy-optimal solution of F with the claimed properties. As the clusters in F remain clusters in G, we conclude the claim. ♢

Let us now describe the remaining steps of our algorithm. Assume that every graph G_i is an induced threshold-like subgraph of G as given in Lemma 4.11.

1. For every G_i, compute a greedy-optimal solution $S(G_i) = (S^i_1, \ldots, S^i_{k_i})$.
2. Construct the graph G' from G by removing all edges among the vertices of S^i_p and S^i_q, for every G_i and $1 \leq p, q \leq k_i$ with $p \neq q$.
3. Run the algorithm described in Theorem 4.7 on G' and return the obtained solution.

For the correctness, observe that Claim 4.13 (3) shows that every induced subgraph of G' on the vertices of $V(G_i)$ is indeed a stable-like graph. Since the vertices of each set I_i and I_j of G_i and G_j, respectively, have no common neighbor in G', we conclude that G' is indeed a stable-like graph. Moreover Claim 4.14 implies that the constructed solution is an optimal solution of G, as required. Regarding the running time, observe that a greedy-optimal solution on each P_4-free graph G_i can be computed in $O(n_i^2)$ time where $n_i = |V(G_i)|$ [12, 23]. The removal of the described edges and the algorithm given in Theorem 4.7 takes linear time. Therefore the total running of the algorithm is $O(n^2)$. □

5 Concluding remarks

It is notable that our algorithm for interval graphs, heavily relies on the linear structure obtained from their clique paths. Such an observation, leads us to consider few open questions regarding two main directions. On the one hand, it seems tempting to adjust our algorithm for other vertex partitioning problems on interval graphs within a more general framework, as already have been studied for particular graph properties [5, 13, 20, 21, 28]. On the other hand, it is reasonable to ask whether our approach works for Cluster Deletion on graphs admitting similar linear structure such as permutation graphs, or graphs having bounded linear related parameter. Towards the latter direction, observe that Cluster Deletion can be solved in linear time on graphs of bounded treewidth [8, 25].

Although for other structural parameters it seems rather difficult to obtain a similar result, it is still interesting to settle the complexity of Cluster Deletion on distance

hereditary graphs that admit constant clique-width [15]. In fact, we would like to settle the case in which from a given cograph (P_4-free graph) we can append degree-one vertices. This comes in conjunction with the starlike graphs, as they can be seen as a degree-one extension of a clique.

Acknowledgements

We would like to thank Paloma T. Lima for bringing to our attention the class of starlike graphs introduced in [6].

References

[1] N. Bansal, A. Blum, and S. Chawla, *Correlation clustering*, Machine Learning, 56 (2004), pp. 89–113.

[2] F. Bonomo, G. Durán, A. Napoli, and M. Valencia-Pabon, *A one-to-one correspondence between potential solutions of the cluster deletion problem and the minimum sum coloring problem, and its application to P_4-sparse graphs*, Inf. Proc. Lett., 115 (2015), pp. 600–603.

[3] F. Bonomo, G. Durán, and M. Valencia-Pabon, *Complexity of the cluster deletion problem on subclasses of chordal graphs*, Theor. Comp. Science, 600 (2015), pp. 59–69.

[4] A. Brandstädt, V. B. Le, and J. Spinrad, *Graph Classes: A Survey*, Society for Industrial and Applied Mathematics, 1999.

[5] B. Bui-Xuan, J. A. Telle, and M. Vatshelle, *Fast dynamic programming for locally checkable vertex subset and vertex partitioning problems*, Theor. Comput. Sci., 511 (2013), pp. 66–76.

[6] M. R. Cerioli and J. L. Szwarcfiter, *Characterizing intersection graphs of substars of a star*, Ars Comb., 79 (2006).

[7] M. Charikar, V. Guruswami, and A. Wirth, *Clustering with qualitative information*, in Proceedings of FOCS 2003, 2003, pp. 524–533.

[8] B. Courcelle, *The monadic second-order logic of graphs i: Recognizable sets of finite graphs*, Information and Computation, 85 (1990), pp. 12–75.

[9] A. Dessmark, J. Jansson, A. Lingas, E. Lundell, and M. Persson, *On the approximability of maximum and minimum edge clique partition problems*, Int. J. Found. Comput. Sci., 18 (2007), pp. 217–226.

[10] S. Földes and P. L. Hammer, *Split graphs*, Congressus Numerantium, 19 (1977), pp. 311–315.

[11] D. R. Fulkerson and O. A. Gross, *Incidence matrices and interval graphs*, Pacific Journal of Mathematics, 15 (1965), pp. 835–855.

[12] Y. Gao, D. R. Hare, and J. Nastos, *The cluster deletion problem for cographs*, Discrete Mathematics, 313 (2013), pp. 2763–2771.

[13] M. U. Gerber and D. Kobler, *Algorithms for vertex-partitioning problems on graphs with fixed clique-width*, Theor. Comp. Science, 299 (2003), pp. 719 – 734.

[14] P. A. Golovach, P. Heggernes, A. L. Konstantinidis, P. T. Lima, and C. Papadopoulos, *Parameterized aspects of strong subgraph closure*, Algorithmica, 82 (2020), pp. 2006–2038.

[15] M. C. Golumbic and U. Rotics, *On the clique-width of some perfect graph classes*, Int. J. Found. Comput. Sci., 11 (2000), pp. 423–443.

[16] N. Grüttemeier and C. Komusiewicz, *On the relation of strong triadic closure and cluster deletion*, in Proceedings of WG 2018, 2018, pp. 239–251.

[17] P. L. Hammer and B. Simeone, *The splittance of a graph*, Combinatorica, 1 (1981), pp. 275–284.

[18] P. Hansen and B. Jaumard, *Cluster analysis and mathematical programming*, Math. Programming, 79 (1997), pp. 191–215.

[19] J. Hartigan, *Clustering Algorithms*, Wiley, New York, 1975.

[20] P. Heggernes, D. Lokshtanov, J. Nederlof, C. Paul, and J. A. Telle, *Generalized graph clustering: recognizing (p, q)-cluster graphs*, in Proceedings of WG 2010, 2010, pp. 171–183.

[21] I. A. Kanj, C. Komusiewicz, M. Sorge, and E. J. van Leeuwen, *Solving partition problems almost always requires pushing many vertices around*, in Proceedings of ESA 2018, 2018, pp. 51:1–51:14.

[22] C. Komusiewicz and J. Uhlmann, *Cluster editing with locally bounded modifications*, Discrete Applied Mathematics, 160 (2012), pp. 2259–2270.

[23] A. L. Konstantinidis, S. D. Nikolopoulos, and C. Papadopoulos, *Strong triadic closure in cographs and graphs of low maximum degree*, Theor. Comput. Sci., 740 (2018), pp. 76–84.

[24] A. L. Konstantinidis and C. Papadopoulos, *Cluster deletion on interval graphs and split related graphs*, in Proceedings of MFCS 2019, 2019, pp. 12:1–12:14.

[25] A. L. Konstantinidis and C. Papadopoulos, *Maximizing the strong triadic closure in split graphs and proper interval graphs*, Discret. Appl. Math., 285 (2020), pp. 79–95.

[26] S. E. Schaeffer, *Graph clustering*, Computer Science Review, 1 (2007), pp. 27–64.

[27] R. Shamir, R. Sharan, and D. Tsur, *Cluster graph modification problems*, Discrete Applied Mathematics, 144 (2004), pp. 173–182.

[28] J. A. Telle and A. Proskurowski, *Algorithms for vertex partitioning problems on partial* k*-trees*, SIAM J. Discrete Math., 10 (1997), pp. 529–550.