Cluster Deletion on Interval Graphs and Split Related Graphs

Athanasios L. Konstantinidis
Department of Mathematics, University of Ioannina, Greece
skonstan@cc.uoi.gr

Charis Papadopoulos
Department of Mathematics, University of Ioannina, Greece
charis@cs.uoi.gr

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 the 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.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases Cluster deletion, interval graphs, split graphs

Digital Object Identifier 10.4230/LIPIcs.MFCS.2019.12

Related Version `https://arxiv.org/abs/1904.09470`

Funding This research 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).

1 Introduction

In graph theoretic notions, *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, 5, 23]. 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.

Although Cluster Deletion is NP-hard on general graphs [24], 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. [11] showed that Cluster Deletion is NP-hard on $(C_5, P_5, \text{bull}, \text{fork}, \text{co-gem}, \text{4-pan}, \text{co-4-pan})$-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 [11], 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 [11]. In fact, the greedy approach of selecting a maximum clique provides a 2-approximation algorithm, though not necessarily in polynomial-time [7]. 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, 11].

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.

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 consider true twins at the independent set, 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. We then study two different classes of such generalization of split graphs that can be viewed as the parallel of split graphs that admit disjoint clique-neighborhood and nested clique-neighborhood. 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 functions minimization.

2 Preliminaries

All graphs considered here are simple and undirected. We refer to Diestel's classical book [8] for standard graph terminology that is undefined here. 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 $N(u) = N(v)$. 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 $\mathcal{H}$.

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 $G - F$ 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 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. Notice, however, that the weights on the vertices imply weights on the edges of the graph, as they contribute to the total cost of external and internal edges in a solution. 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.** *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 y 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 large as the number of internal edges of the optimal solution. ◀

Moreover, we prove the following generalization of Lemma 1.

▶ **Lemma 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 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.*

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. 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 [10, 13]. 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., [10, 13]). 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 said to be *guarded* by two vertices x and z if $\min\{a(x), a(z)\} \leq a(y)$ and $b(y) \leq \max\{b(x), b(z)\}$ hold. 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 $a(y) \leq i \leq b(y)$. Observe that the following statement generalizes Corollary 4, in the sense that the neighborhood of the guarded vertex y is not necessarily contained in the neighborhood of x or z.

▶ **Lemma 5.** *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 $y \in C$.*

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 (see also Figure 1):

- $U(j)$ 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 $K_{b(j)}$. For two vertices v_i, v_j with $b(v_i) \leq b(v_j)$, we define the following:

- $A_{i,j}$ 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 candidate 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 $V_{i,j} = M(j) \cup \{v_j\}$. 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 $b(v_i) \leq b(v_j)$.

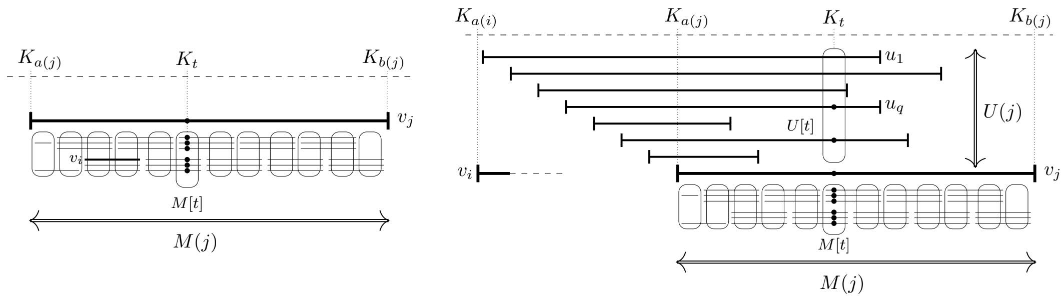

Figure 1 Illustrating the sets $M(j)$ and $U(j)$ for v_j. The left part shows the case in which $v_i \in M(j)$ (or, equivalently, $V_{i,j} = V_{j,j}$), whereas the right part corresponds to the case in which $a(v_i) < a(v_j)$.

▶ **Lemma 6.** *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. Observe that $w \in M(j)$ implies $a(v_j) \leq a(w) \leq b(w) \leq b(v_j)$. Since $v_j, w \in C$, we know that there is a maximal clique K_t for which $C \subseteq K_t$ with $a(v_j) \leq a(w) \leq t \leq b(w) \leq b(v_j)$. 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 $b(y) \leq b(v_j) = \max\{b(v_j), b(w)\}$. 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 5 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 $C \subseteq K_t$. ◀

By Lemma 6, 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 a 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 $a(u_1) \leq \cdots \leq a(u_{|U(j)|})$ (see the right part of Figure 1).

▶ **Lemma 7.** *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 5 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 the group $M[t]$, 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 8.** *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 $M[t] \subseteq C$.*

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 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 Lemmas 6 and 8. Then, for each of such candidate $M[t]$, we apply Lemma 7 to construct all subsets containing the last q vertices of $U[t]_{\leq a}$. Thus, there are at most n^2 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 $B_j(t) = \big((K_{t+1} \cup \cdots \cup K_{b(j)}) \setminus K_t\big) \cap M(j)$. The reason of breaking the vertices of the part $M(j)$ into sets $B_j(t)$ is the following.

▶ **Lemma 9.** *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 $x \in M(j)$. If $x \in M[t]$, then $x \in C$ by Lemma 6, which implies that $y \notin C$. If $x \in (M[a(v_j)] \cup \cdots \cup M[t-1]) \setminus M[t]$ then $xy \notin E(G)$. 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 $C' \cap B_j(t) \neq \varnothing$. 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 3 we reach a contradiction to x and v_j belonging to different clusters. ◀

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 10.** *Let $V_t = K_t \cap V_{i,j}$, for every $\min\{a(v_i), a(v_j)\} \leq t \leq b(v_j)$. 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 $j' = b\text{-}\max(S_L)$.*

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

Observe that all vertices of $C(\ell(j), r(j))$ induce a clique in G, because $C(\ell(j), r(j)) \subseteq K_{r(j)}$. We say that a vertex u *crosses* the pair $(\ell(j), r(j))$ if $a(u) < \ell_r(j)$ and $r(j) \leq b(u)$. 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 11.** *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 $\ell_r \leq a(y)$.*

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 $\ell(j) = a(v_j)$. 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 a bounding pair (ℓ, r) for v_j, we write $(\ell(j), r(j)) < (\ell, r)$ to denote the set of admissible pairs $(\ell(j), r(j))$ for v_j with the following restriction on $r(j)$:

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

Observe that if $b(v_j) < r$ holds, then $(\ell(j), r(j)) < (\ell, r)$ 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. 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 (observe that $v_j \in C(\ell(j), r(j))$ and $v_j \notin C(\ell, r)$).

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 as follows: Let (ℓ, r) be a bounding pair for v_j.

- $A_{i,j}[\ell, r]$ 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.

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

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

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

▶ **Theorem 13.** Cluster Deletion *is polynomial-time solvable on interval graphs.*

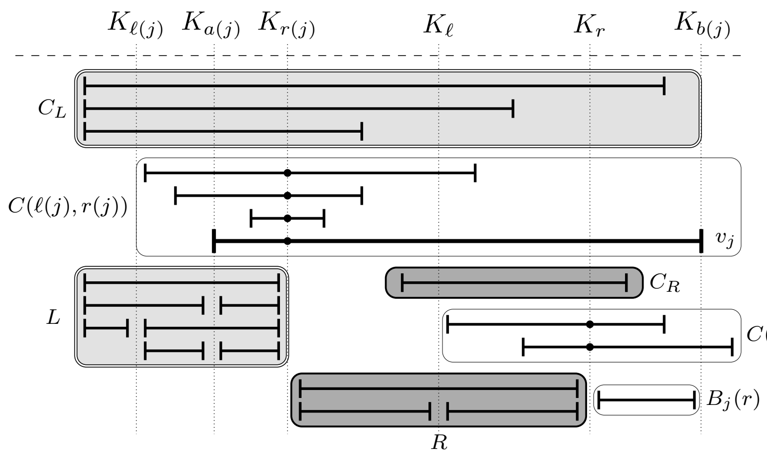

Figure 2 A partition of the set of vertices given in $A_{i,j}[\ell, r]$, where $V_L = C_L \cup L$ and $V_R = C_R \cup R$. Observe that $B_j(r(j)) = R \cup C_R \cup (C(\ell, r) \cap V_{i,j}) \cup B_j(r)$.

4 Cluster Deletion on a generalization of split 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 [9]. 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 1, it is natural to consider true twins at the independent set, as they are grouped together in an optimal solution and 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. Before presenting 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.

▶ **Definition 14.** *A graph $G = (V, E)$ is called split-twin graph if its vertex set can be partitioned into C and I such that $G[C]$ is a clique and the vertices of each connected component of $G[I]$ form true twins in G.*

It is clear that in a split-twin graph G 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^*. Figure 3 illustrates the induced subgraphs that are forbidden in a split-twin graph.

▶ **Proposition 15.** *A graph G is split-twin 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.*

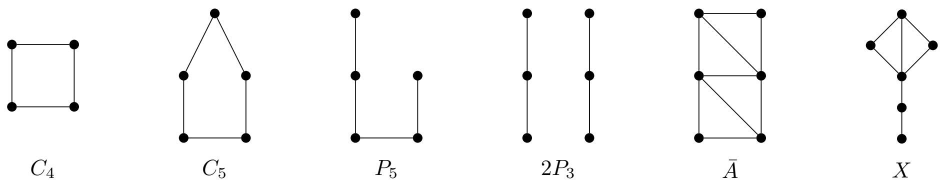

Figure 3 The list of forbidden induced subgraph characterization for split-twin graphs.

Thus by Proposition 15, split-twin 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 split-twin 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 split-twin graph. Therefore, by the NP-completeness given in [3], we have:

▶ **Theorem 16.** Cluster Deletion *is NP-complete on split-twin 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 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) $q = |C|$ [3]. 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 split-twin graphs.

▶ **Theorem 17.** *There exists a polynomial time algorithm that, given an instance (C, I, k) for EWCD, produces an equivalent instance for* Cluster Deletion *on split-twin graphs.*

Proof. From G, we build a split-twin 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 split-twin 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 solution S' in G' having at least $k + |I| \cdot \binom{q}{2}$ edges. All vertices of I'_j are true twins and, by Lemma 1, 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 C in S. Therefore, there is an edge weighted solution S having weight at least k. ◀

4.1 Polynomial-time algorithms on subclasses of split-twin graphs

Due to the hardness result given in Theorems 16 and 17, it is natural to consider subclasses of split-twin graphs related to their analogue subclasses of split graphs. We consider two such subclasses. The first one corresponds to the split-twin graphs such that the vertices of I have no common neighbor in the clique, unless they are true or false twins. The second subclass corresponds to threshold graphs (i.e., split graphs in which the vertices of the independent set have nested neighborhood) and form the split-twin graphs in which the vertices of I have a nested neighborhood. We formally define such graphs and give polynomial-time algorithms for CLUSTER DELETION. For a vertex $x \in I$ we write $N_C(x)$ to denote the set $N(x) \cap C$.

▶ **Definition 18.** *A split-twin graph G with partition (C, I) on its vertices is called 1-split-twin graph if for any two vertices $x, y \in I$, either $N_C(x) \cap N_C(y) = \varnothing$ or $N_C(x) = N_C(y)$.*

It is not difficult to see that in a 1-split-twin graph, any two vertices of I having a common neighbor in C have the same neighborhood in C. Close related to 1-split-twin graphs, are the *1-split graphs* which are the edge-weighted split graphs in which every vertex of the independent set is adjacent to exactly one vertex of the clique. It is known that the (edge-weighted) CLUSTER DELETION is solved in polynomial-time on 1-split graphs [3]. Let us explain how to use the algorithm on a 1-split graph to obtain a polynomial-time algorithm on a 1-split-twin graph $G = (C, I)$. Observe that the contracted graph G^* is 1-split. Let xy be an edge of G^*. Denote by $w(x)$ and $w(y)$ the weights assigned to x and y that correspond to the sizes of their true twins classes in G. From the vertex-weighted 1-split graph G^*, construct an edge-weighted 1-split graph H^* by removing the vertex weights and for each edge xy assign weight $w(x) \cdot w(y)$. Then, given a solution of the edge-weighted H^* taken from the algorithm of [3], we obtain a solution for Cluster Deletion on G by adding the internal edges corresponding to each contracted vertex.

Notice, however, that the running time is bounded by the polynomial-time algorithm of 1-split graphs. In particular the described algorithm of 1-split graphs is accomplished through a minimizer of a general submodular function provided a given oracle for evaluating the function value [3]. This means that through such an approach it is unlikely to achieve a better running time for 1-split-twin graphs, unless there is a faster algorithm with less number of oracle calls for finding a minimizer of a general submodular function. With our next result we provide a simpler and faster (linear-time) algorithm for Cluster Deletion on 1-split-twin graphs that avoids the usage of submodular functions minimization.

▶ **Theorem 19.** Cluster Deletion *is linear-time solvable on 1-split-twin graphs.*

Proof. Let G be a 1-split-twin 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 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 split-twin 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 $N_C(x) \cap N_C(y) \neq \varnothing$ we reach a contradiction to the definition of 1-split-twin 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 1 they all belong to the same cluster of S. Also, by Lemma 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 1, 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 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), 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 1-split-twin graph G. ◀

▶ **Definition 20.** *A split-twin graph G with partition (C, I) on its vertices is called threshold-twin graph if the vertices of I can be ordered $w_1, \ldots, w_{|I|}$ such that for any $w_i, w_j \in I$ with $i < j$, we have $N_C(w_i) \subseteq N_C(w_j)$.*

For the next result, we prove that there is no P_4 in a threshold-twin graph (P_4-free graphs are closed under true twins addition). Thus, by the algorithm given in [11], we have:

▶ **Theorem 21.** Cluster Deletion *is polynomial-time solvable on threshold-twin graphs.*

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 [4, 12, 20, 21, 25]. 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 as a vertex partitioning problem can be solved in linear time on graphs of bounded treewidth by using Courcelle's machinery [6].

Although for other structural parameters it seems rather difficult to obtain 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 we can append degree-one vertices. This comes in conjunction with the 1-split-twin graphs, as they can be seen as a degree-one extension of a clique.

References

1 N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56:89–113, 2004.

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.*, 115:600–603, 2015.

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

4 B. Bui-Xuan, J. A. Telle, and M. Vatshelle. Fast dynamic programming for locally checkable vertex subset and vertex partitioning problems. *Theor. Comput. Sci.*, 511:66–76, 2013.

5 M. Charikar, V. Guruswami, and A. Wirth. Clustering with qualitative information. In *Proceedings of FOCS 2003*, pages 524–533, 2003.

6 B. Courcelle. The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Information and Computation*, 85:12–75, 1990.

7 A. Dessmark, J. Jansson, A. Lingas, E.-M. Lundell, and M. Persson. On the approximability of maximum and minimum edge clique partition problems. *Int. J. Found. Comput. Sci.*, 18:217–226, 2007.

8 R. Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate Texts in Mathematics*. Springer, 2012.

9 S. Földes and P. L. Hammer. Split graphs. *Congressus Numerantium*, 19:311–315, 1977.

10 D. R. Fulkerson and O. A. Gross. Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, 15:835–855, 1965.

11 Y. Gao, D. R. Hare, and J. Nastos. The cluster deletion problem for cographs. *Discrete Mathematics*, 313:2763–2771, 2013.

12 M. U. Gerber and D. Kobler. Algorithms for vertex-partitioning problems on graphs with fixed clique-width. *Theor. Comp. Science*, 299:719–734, 2003.

13 P. C. Gilmore and A. J. Hoffman. A characterization of comparability graphs and of interval graphs. *Canadian Journal of Mathematics*, 16:539–548, 1964.

14 P. A. Golovach, P. Heggernes, A. L. Konstantinidis, P. T. Lima, and C. Papadopoulos. Parameterized aspects of strong subgraph closure. In *Proceedings of SWAT 2018*, pages 23:1–23:13, 2018.

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

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

17 P. L. Hammer and B. Simeone. The splittance of a graph. *Combinatorica*, 1:275–284, 1981.

18 P. Hansen and B. Jaumard. Cluster analysis and mathematical programming. *Math. Programming*, 79:191–215, 1997.

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*, pages 171–183, 2010.

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

22 C. Komusiewicz and J. Uhlmann. Cluster editing with locally bounded modifications. *Discrete Applied Mathematics*, 160:2259–2270, 2012.

23 S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.

24 R. Shamir, R. Sharan, and D. Tsur. Cluster graph modification problems. *Discrete Applied Mathematics*, 144:173–182, 2004.

25 J. A. Telle and A. Proskurowski. Algorithms for Vertex Partitioning Problems on Partial k-Trees. *SIAM J. Discrete Math.*, 10:529–550, 1997.