Theoretical Computer Science 600 (2015) 59–69

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Complexity of the cluster deletion problem on subclasses of chordal graphs ☆

Flavia Bonomo [a], Guillermo Durán [b,c], Mario Valencia-Pabon [d,*,1]

[a] CONICET and Dep. de Computación, FCEN, Universidad de Buenos Aires, Argentina
[b] CONICET and Dep. de Matemática and Instituto de Cálculo, FCEN, Universidad de Buenos Aires, Argentina
[c] Dep. de Ingeniería Industrial, FCFM, Universidad de Chile, Santiago, Chile
[d] Université Paris-13, Sorbonne Paris Cité LIPN, CNRS UMR7030, Villetaneuse, France

ARTICLE INFO

Article history:
Received 28 October 2014
Received in revised form 30 June 2015
Accepted 1 July 2015
Available online 7 July 2015
Communicated by V.Th. Paschos

Keywords:
Block graphs
Cliques
Edge-deletion
Cluster deletion
Interval graphs
Split graphs
Submodular functions
Chordal graphs
Cographs
NP-completeness

ABSTRACT

We consider the following vertex-partition problem on graphs, known as the CLUSTER DELETION (CD) problem: given a graph with real nonnegative edge weights, partition the vertices into clusters (in this case, cliques) to minimize the total weight of edges outside the clusters. The decision version of this optimization problem is known to be NP-complete even for unweighted graphs and has been studied extensively. We investigate the complexity of the decision CD problem for the family of chordal graphs, showing that it is NP-complete for weighted split graphs, weighted interval graphs and unweighted chordal graphs. We also prove that the problem is NP-complete for weighted cographs. Some polynomial-time solvable cases of the optimization problem are also identified, in particular CD for unweighted split graphs, unweighted proper interval graphs and weighted block graphs.

1. Introduction

Clustering is an important task in the data analysis process. It can be viewed as a data modeling technique that provides an attractive mechanism for automatically finding the hidden structure of large data sets. The input to the problem is typically a set of elements and pairwise similarity values between elements. The goal is to partition these elements into subsets called *clusters* such that two meta-criteria are satisfied: *homogeneity* – elements in a given cluster are highly similar to each other; and *separation* – elements from different clusters have low similarity to each other. In the graph theoretic approach to clustering, one builds from the raw data a *similarity graph* whose vertices correspond to elements and there is an edge between two vertices if and only if the similarity of their corresponding elements exceeds a predefined threshold [13,14]. Cluster graphs have been used in a variety of applications whenever clustering of objects is studied or when consistent data

☆ Partially supported by MathAmSud Project 13MATH-07 (Argentina–Brazil–Chile–France), UBACyT Grant 20020130100808BA, CONICET PIP 112-200901-00178 and 112-201201-00450CO and ANPCyT PICT 2012-1324 (Argentina), FONDECYT Grant 1140787 and Millennium Science Institute "Complex Engineering Systems" (Chile).

* Corresponding author.
E-mail addresses: fbonomo@dc.uba.ar (F. Bonomo), gduran@dm.uba.ar (G. Durán), valencia@lipn.univ-paris13.fr (M. Valencia-Pabon).

[1] Currently "en délégation" at the INRIA Nancy-Grand Est.

http://dx.doi.org/10.1016/j.tcs.2015.07.001
0304-3975/

is sought among noisy or error-prone data [1,5]. Ideally, the resulting graph would be a *cluster graph*, that is, a graph in which every connected component is a clique (i.e., a complete subgraph). In practice, it is only close to being such, since similarity data is experimental and therefore error-prone.

The *cluster deletion* problem consists in finding the minimum number of edges that must be removed from an input graph to make the resulting graph a cluster graph. In its decision version, the cluster deletion problem has a non-negative integer parameter W and asks if one can remove a set of at most W edges from the input graph such that the resulting graph is a cluster graph. There exist several results for the cluster deletion problem (see for example [3,17,22] and references therein). The cluster deletion problem is known to be NP-complete [22] for general graphs. Moreover, Shamir et al. [22] showed that it remains NP-hard when imposing that the input graph should be clustered into exactly $d \geq 3$ components. They also showed that when the input graph is clustered into exactly 2 components, the problem is polynomial-time solvable. Komusiewicz et al. [17] proved that cluster deletion is hard for C_4-free graphs with maximum degree 4 and gave an $O(n^{1.5}\log^2 n)$ time algorithm for solving cluster deletion on graphs with maximum degree 3, where n is the number of vertices of the graph.

Based on results obtained by Demaine et al. [7] for a variant of a clustering problem, Dessmark et al. [8] provided a polynomial $O(\log n)$-approximation algorithm for the edge-weighted version of the cluster deletion problem. In this version, the edges of the graph have an associated weight and the aim is to minimize the sum of the weights of the removed edges. Considering it as a decision problem, the aim is to determine, for some input parameter W, if there is a set of edges with a total weight of at most W such that removing it from the input graph will make the resulting graph a cluster graph. Note that if we allow the weight function to be negative on some edges, we can reduce any clustering problem to a clustering problem whose input graph is a weighted complete graph by assigning a negative weight with a large enough absolute value to the edges that are missing in the original graph. Thus, the problem with arbitrary weights is NP-complete for any graph class admitting arbitrarily large cliques. We will assume throughout that all of the weight functions are nonnegative.

Dessmark et al. [8] also showed that for the unweighted version of cluster deletion on general graphs, the greedy algorithm that finds iteratively maximum cliques gives a 2-approximation algorithm to the optimal cluster deletion. The complexity of such an algorithm reflects the complexity of iteratively finding maximum cliques, so it is a polynomial-time approximation algorithm for certain graph classes. Recently, Gao et al. [11] showed that the greedy algorithm that finds iteratively maximum cliques gives an optimal solution for the class of graphs known as *cographs*. This implies that the cluster deletion problem is polynomial-time solvable on unweighted cographs. With a different approach based on modular decomposition, it is proved in [4] that the unweighted cluster deletion problem is polynomial-time solvable on a subclass of P_4-sparse graphs that strictly includes P_4-reducible graphs (which are, in turn, a superclass of cographs). Gao et al. [11] also showed that the cluster deletion problem is NP-hard on (C_5, P_5)-free graphs, on $(2K_2, 3K_1)$-free graphs and on $(C_5, P_5, \text{bull}, 4\text{-pan}, \text{fork}, \text{co-gem}, \text{co-4-pan})$-free graphs. For weighted graphs, the cluster deletion problem can be solved in polynomial time on the class of triangle-free graphs given that it is equivalent to maximum weighted matching [9]. The cluster deletion and other clustering problems have been studied extensively in the context of fixed-parameter tractability (FPT) ([6,18] and references therein). Many of the recently-developed FPT algorithms rely on being able to solve cluster deletion in polynomial-time on restricted graph structures [3].

A heuristic for solving clustering problems consists in modifying a given input graph into another graph having some nice algorithmic properties and then solving the clustering problem for the modified graph. For example, to solve a genetic clustering problem, Kaba et al. [16] transform any input graph into a chordal graph via minimal triangulations of the former one. Once the input graph has been so transformed, they exploit the algorithmic properties of chordal graphs to obtain good solutions to their clustering problem. If solving a clustering problem for a specific graph family $\mathcal{F}$ is computationally hard, however, the heuristic which first transforms the input graph into a graph in $\mathcal{F}$ and then solves the problem on the resulting graph may not be a good approach. Therefore, it is important to know how to solve a clustering problem on specific graph families before using the above-described heuristic for general input clustering graphs.

Some known results are summarized in Table 1; those obtained in the present work are shown in bold face. We conclude this introduction with some definitions.

Let $G = (V, E)$ be a graph. For each vertex $v \in V$, we denote as $N(v) = \{u : vu \in E\}$ the set of neighbors of v in G. Two vertices v and w are called *true twins* if $N(v) \cup \{v\} = N(w) \cup \{w\}$. A graph G is said to be *weighted* if there is a nonnegative weight function $w : E \to \mathbb{R}^+$ associated with it. For the algorithms involving weighted graphs we will assume that the weights are rational (or belong to any ordered field in which we can perform the field operations and the order comparisons algorithmically). An *unweighted* graph is a graph in which each edge has a weight equal to 1. We say that a set F of edges of a given graph has a *uniform* weight if all the edges in F have the same weight.

Let H and G be graphs. If G contains no induced subgraph isomorphic to H then G is an *H-free graph*. Let P_k (resp. C_k) denote a path (resp. cycle) on k vertices. Let $K_{m,n} = (A \cup B, E)$ denote the complete bipartite graph, where A (resp. B) is an independent set of size m (resp. n) and E is the set of all the edges with an endpoint in A and an endpoint in B. We refer to [23] for standard definitions and results in graph theory. A graph is *chordal* if and only if it does not contain a cycle of length at least four as an induced subgraph. Given a vertex partition $S = C_1, \ldots, C_k$ of a graph G, we call the *weight* of S, denoted $w(S)$, the sum of the weights of all edges $e = uv$ such that $u \in C_i$, $v \in C_j$, with $i \neq j$. An edge is called *external* with respect to the partition S if its endpoints belong to distinct sets of S, and *internal* otherwise. The cluster deletion problem for an (un)weighted graph G can be redefined as the problem of finding a clique partition of G with minimum weight. We will assume throughout that all NP-completeness results concern the decision version of the cluster deletion problem.

Table 1
A summary of some of the known complexity results, with those obtained in the present work shown in bold face. Question marks denote open problems.

Class	Cluster deletion	Weighted cluster deletion
General	NP-c [22]	NP-c [22]
Complete split	P [11]	**NP-c** (Theorem 5.2)
3-Split	**P** (Theorem 2.4)	**NP-c** (Theorem 2.1)
Split	**P** (Theorem 2.4)	**NP-c** (Theorem 2.1)
P_5-free chordal	**NP-c** (Theorem 3.2)	**NP-c** (Theorem 3.2)
Block	**P** (Theorem 4.2)	**P** (Theorem 4.2)
Interval	?	**NP-c** (Corollary 5.3)
Proper interval	**P** (Theorem 5.6)	?
Paths of cliques	**P** (Theorem 5.1)	**P** (Theorem 5.1)
Cographs	P [11]	**NP-c** (Corollary 5.7)
P_4-reducible	P [4]	**NP-c** (Corollary 5.7)
$\Delta = 3$	P [17]	?
C_4-free with $\Delta = 4$	NP-c [17]	NP-c [17]
(C_5, P_5)-free	NP-c [11]	NP-c [11]
$(2K_2, 3K_1)$-free	NP-c [11]	NP-c [11]
$(C_5, P_5$, bull, 4-pan, fork, co-gem, co-4-pan)-free	NP-c [11]	NP-c [11]

2. Split graphs

A graph $G = (V, E)$ is a *split graph* if and only if there is a partition of the vertex set V of G into a clique K and an independent set I. Another necessary and sufficient condition for a graph G to be a split graph is that G and its complement $\overline{G}$ be chordal graphs (see [10]). If each vertex of the independent set is adjacent to exactly p vertices of the clique K with $p \geq 1$, then G is called a *p-split* graph.

In this section, we prove the NP-completeness of the weighted cluster deletion problem for split graphs by a reduction from the *exact cover by 3-sets* problem (*X3C* problem for short). The formal definition of the X3C problem can be stated as follows:

Instance: A set X with $3q$ elements and a collection C of 3-element subsets of X.

Question: Does C contain an exact cover for X? In other words, is there a subset $C' \subseteq C$ such that every element of X occurs in exactly one member of C'?

The X3C problem is known to be NP-complete [12]. We may further assume that the union of the subsets in C covers X, otherwise the answer is trivially no.

Theorem 2.1 *(NP-completeness on weighted* 3*-split graphs). The cluster deletion problem is NP-complete for weighted* 3*-split graphs, even if the weight of all the internal edges of the clique is* 1 *and the weight of the edges between the clique and the independent set is uniform.*

Proof. It is easily seen that the cluster deletion problem is in NP, since we can readily verify in polynomial time whether a vertex partition of a graph is a clique partition and whether its weight is less than a given value W. Let $X = \{x_1, \ldots, x_{3q}\}$ and $C = \{c_1, \ldots, c_m\}$ be an instance of the X3C problem, where each element $c_i \in C$ is a 3-element subset of X with $m \geq q \geq 1$. We want to know if there exists a subset $C' \subseteq C$ with size q such that each element in X belongs to exactly one of the elements in C'. We will construct a weighted split graph $G = (K_X \cup I_C, E)$, where K_X induces a clique with $3q$ vertices and I_C induces an independent set with m vertices. For each element $x_i \in X$ there is a vertex x_i in K_X and for each 3-subset $c_j \in C$ there is a vertex $c_j \in I_C$. The edge set E is defined in such a way that K_X induces a complete graph, I_C is an independent set, and x_i in K_X is adjacent to c_j in I_C if and only if x_i is an element of X that belongs to the set c_j in C. We will call E_X the set of edges with both endpoints in K_X and E_C the set of edges with an endpoint in K_X and an endpoint in I_C. The weight of each edge in E_X is equal to 1 and the weight of each edge in E_C is equal to $3q$. Clearly, G is a split graph and can be obtained in polynomial time from the instance (X, C). Let $W = \binom{3q}{2} - 3q + 9q(m-q)$. We will show that there exists a subset $C' \subseteq C$, with $|C'| = q$ exactly covering X if and only if G admits a clique partition where the sum of the weights of the external edges is at most W. In other words, there exists a solution for the X3C problem if and only if there exists a solution for the cluster deletion problem for G with a weight of at most W.

Assume first that there exists $C' \subseteq C$, with $C' = \{c'_1, \ldots, c'_q\}$ such that $c'_i \cap c'_j = \emptyset$ whenever $i \neq j$, and $\bigcup_{c'_j \in C'} c'_j = X$. The clique partition for G can be constructed as follows: for each $c'_j \in C'$, choose the clique of G formed by the corresponding vertex c'_j in I_C and its neighbors in K_X. Each of the remaining $m - q$ vertices in I_C forms a clique of size one (a singleton).

The sum of the edge weights outside those cliques is exactly equal to W. The weight of the clique partition is equal to $\binom{3q}{2}$ (the weight of all the edges in K_X) minus $3q$ (each one of the q cliques with size four has a triangle in K_X) plus $9q(m-q)$ (the $3(m-q)$ edges of weight $3q$ joining the $m-q$ singletons and their neighbors in K_X are external with respect to the partition).

Now assume conversely that G has a clique partition with a weight of at most W. We must prove that there exists $C' \subseteq C$ with $|C'| = q$ such that C' is an exact cover for X. We begin by analyzing the structure of the optimal solutions of the optimization version of the cluster deletion problem for G.

Claim 2.2. *Let S be a clique partition of G that is an optimal solution for the cluster deletion problem in G in its optimization version. Then no clique in S is formed only by vertices in K_X.*

Note first of all that it is not possible to have two cliques $A_1, A_2 \in S$ formed only by vertices in K_X. If the case were otherwise we could define the clique partition $S' = S \setminus \{A_1, A_2\} \cup \{A_1 \cup A_2\}$, but then clearly $w(S) - w(S') = |A_1|.|A_2| \geq 1$, which contradicts the optimality of S. Thus, every clique of S contains at most one vertex of I_C, so there are m cliques (possibly singletons) each containing a vertex of I_C. Let $S = A_1, B_1, \ldots, B_m$ be the clique partition of G and let $w(S)$ be the weight of S. Assume that clique A_1 is formed only by vertices in K_X and each clique B_j, $1 \leq j \leq m$, contains the vertex c_j of I_C and zero, one, two or three vertices in K_X.

Let c_j be in I_C. We consider the following cases:

- $|B_j| < 3$ and there is a neighbor y of c_j in A_1. Let S' be the clique partition $S' = S \setminus \{A_1, B_j\} \cup \{A_1 \setminus \{y\}, B_j \cup \{y\}\}$. It is readily seen that $w(S') = w(S) - (3q+2) + (|A_1| - 1) = w(S) + |A_1| - 3q - 3$. However, $|A_1| \leq 3q - 2$. Therefore, $w(S') < w(S)$, which contradicts the optimality of S.
- Similarly to the previous case, $|B_j| = 2$ (resp. $|B_j| = 1$) and there is a neighbor y of c_j in A_1. Let S' be the clique partition $S' = S \setminus \{A_1, B_j\} \cup \{A_1 \setminus \{y\}, B_j \cup \{y\}\}$. Observe that $w(S') = w(S) + |A_1| - 3q - 2 < w(S)$ (resp. $w(S') = w(S) + |A_1| - 3q - 1 < w(S)$), which is a contradiction to the optimality of S.

Since S is an optimal solution, it follows that no vertex in I_C is adjacent to a vertex in A_1. But by construction, this implies that X is not covered by C, which contradicts our assumption about the instances of X3C. This ends the proof of this claim. $\diamond$

By the above claim, there must be an optimal solution S for the cluster deletion problem of G (optimization version) of the form $S = B_1, \ldots, B_m$, where each clique B_j contains exactly one vertex c_j of I_C, for $1 \leq j \leq m$. Let t_i be the number of cliques of S with i vertices in K_X, for $i = 0, \ldots, 3$.

Claim 2.3. $t_3 > 0$.

Suppose that $t_3 = 0$. If $t_2 \neq 0$, then there is a clique $B_j = \{c_j, x_{j_1}, x_{j_2}\}$ in S. By construction, c_j has another neighbor $y \in K_X$. If $y \in B_i$ with $|B_i| = 3$ (resp. $|B_i| = 2$), we then obtain another clique partition $S' = S \setminus \{B_i, B_j\} \cup \{B_i \setminus \{y\}, B_j \cup \{y\}\}$ such that $w(S') = w(S) - (3q+2) + 3q + 1 = w(S) - 1 < w(S)$ (resp. $w(S) - (3q+2) + 3q = w(S) - 2 < w(S)$), which contradicts the optimality of S. If $t_3 = t_2 = 0$, then there exist $B_j, B_i, B_s \in S$ such that $B_j = \{c_j, x_{j_1}\}$, $B_i = \{c_i, x_{j_2}\}$ and $B_s = \{c_s, x_{j_3}\}$, where x_{j_1}, x_{j_2} and x_{j_3} are the three neighbors of c_j in K_X. We then obtain another clique partition $S' = S \setminus \{B_j, B_i, B_s\} \cup \{B_j \cup \{x_{j_2}, x_{j_3}\}, B_i \setminus \{x_{j_2}\}, B_s \setminus \{x_{j_3}\}\}$ such that $w(S') = w(S) - (6q+3) + 6q = w(S) - 3 < w(S)$, which again contradicts the optimality of S. Therefore, $t_3 > 0$, which ends the proof of this claim. $\diamond$

The weight $w(S)$ of S can be written as $w(S) = \binom{3q}{2} - (3t_3 + t_2) + (t_2 + 2t_1 + 3t_0)3q$. Moreover, $m = t_3 + t_2 + t_1 + t_0$ and $3q = 3t_3 + 2t_2 + t_1$. Therefore, $3(m-q) = t_2 + 2t_1 + 3t_0$ and $3t_3 + t_2 = 3q - t_2 - t_1$. Then, since $3q \geq 3t_3 + t_2$ it follows that $W \leq w(S)$, with equality if and only if $t_2 = t_1 = 0$ and $t_3 = q$. Indeed, recall that $W = \binom{3q}{2} - 3q + 9q(m-q)$ and thus, if $W = w(S)$ then $3t_3 + 2t_2 + t_1 = 3q = 3t_3 + t_2$, which implies that $t_2 = -t_1$. Since $t_2, t_1 \geq 0$ it must be the case that $t_2 = t_1 = 0$, which forces t_3 to be equal to q. So if G admits a clique partition S with weight W, then there is a solution to the X3C problem. This ends the proof of the theorem. $\square$

For the unweighted case, we will show that the problem can be easily solved on split graphs.

Theorem 2.4 *(Polynomiality on unweighted split graphs). The cluster deletion problem is polynomial-time solvable for unweighted split graphs. Indeed, if (K, I) is a split partition of a graph G such that K is a maximal clique of G, then $\{K\} \cup \{\{v\} : v \in I\}$ is an optimal solution unless there is a vertex v_1 in I adjacent to all but one vertex w in K and that vertex w has a neighbor v_2 in I. In that case, an optimal solution is $\{\{v_1\} \cup (K - \{w\}), \{w, v_2\}\} \cup \{\{v\} : v \in I, v \neq v_1, v_2\}$.*

Proof. Let G be a split graph and (K, I) be a split partition of G such that K is a maximal clique of G. Note that the cluster deletion problem can be seen as the problem consisting in maximizing the number of internal edges in a clique partition S. In order to break ties, we maximize the number of internal edges joining vertices of K in G. We will refer to $S_0 = \{K\} \cup \{\{v\} : v \in I\}$ as the *standard partition*.

It is clear that no optimal solution has two different cliques strictly contained in K. So, every possibly optimal partition that is not the standard one contains at least one clique with a nonempty intersection with both K and I.

Suppose that we have an optimal solution S with respect to the above criterion that is strictly better than S_0. If S contains a clique $A \subsetneq K$ such that $|A| = a \geq 1$ and a clique B containing one vertex of I and $b \geq 1$ vertices of K, then $S' = S \setminus \{A, B\} \cup \{A \cup (B \cap K), B \cap I\}$ is another clique partition which compared to S contains ab new internal edges that join two vertices of K in G but no longer contains b edges that join vertices of K with vertices in I. Thus, it is either strictly better than S or preferable to it by the tie-breaking rule, thereby contradicting the optimality of S. Therefore, since as stated above clique B exists, we may assume that every clique of S contains a vertex of I.

Suppose now that S contains a clique A with $|A \cap K| = a \geq 2$ and a clique B with $|B \cap K| = b \geq 2$. Then $S' = S \setminus \{A, B\} \cup \{(A \cap K) \cup (B \cap K), A \cap I, B \cap I\}$ is another clique partition which compared to S contains ab new internal edges that join two vertices of K in G but no longer contains $a + b$ edges that join vertices of K with vertices of I. Since $a, b \geq 2$, S' is either strictly better than S or preferable to it by the tie-breaking rule, thus contradicting the optimality of S.

Finally suppose that there are three cliques A, B, C containing a vertex in I and $a, b, c \geq 1$ vertices in K, respectively. Then $S' = S \setminus \{A, B, C\} \cup \{(A \cap K) \cup (B \cap K) \cup (C \cap K), A \cap I, B \cap I, C \cap I\}$ is another clique partition which compared to S contains $ab + ac + bc$ new internal edges that join two vertices of K in G but no longer contains $a + b + c$ edges that join vertices of K with vertices of I. Since $ab \geq a$, $bc \geq b$, and $ca \geq c$, then S' is either strictly better than S or preferable to it by the tie-breaking rule, thereby contradicting the optimality of S.

We conclude that S contains exactly two cliques A, B such that $|A \cap I| = |B \cap I| = 1$, $|A \cap K| \geq 1$ and $|B \cap K| = 1$. So, there is a vertex v_1 in I adjacent to all but one vertex w in K which has a neighbor v_2 in I, and $S = \{\{v_1\} \cup (K - \{w\}), \{w, v_2\}\} \cup \{\{v\} : v \in I, v \neq v_1, v_2\}$. □

3. Chordal graphs

Chordal graphs are a class of graphs that have been extensively studied thanks to their peculiar clique-based structure, which lends itself to efficient solutions of algorithmic problems [2].

To prove the main result of this section, we first demonstrate a simple general lemma.

Lemma 3.1 *(True twins). Let G be graph and v, z be true twins in G. Then, for every optimal solution of the unweighted cluster deletion problem, v and z belong to the same clique of the partition.*

Proof. Suppose, on the contrary, that there is an optimal clique partition S such that v belongs to a clique C_1 and z belongs to a different clique C_2. Without loss of generality, we may assume that $|C_1| \leq |C_2|$. But then $S' = S \setminus \{C_1, C_2\} \cup \{C_1 \setminus \{v\}, C_2 \cup \{v\}\}$ is another clique partition such that $w(S') < w(S)$, which is a contradiction. □

Theorem 3.2 *(NP-completeness on unweighted chordal graphs). The cluster deletion problem is NP-complete for unweighted P_5-free chordal graphs.*

Proof. This proof is based on the proof of Theorem 2.1. The reduction is again from the X3C problem. Let $X = \{x_1, \ldots, x_{3q}\}$ and $C = \{c_1, \ldots, c_m\}$ be an instance of the X3C problem, where each element $c_i \in C$ is a 3-element subset of X, with $m \geq q \geq 1$. We want to know if there exists a subset $C' \subseteq C$ of size q such that each element in X belongs to exactly one of the elements in C'. We construct a graph $G = (V, E)$ whose vertex set V is formed by $m + 1$ disjoint cliques of size $3q$, namely $K_X, K_{c_1}, \ldots, K_{c_m}$, such that the vertices of K_X correspond to the elements of X, the clique K_{c_i} corresponds to the element c_i of C, for $i = 1, \ldots, m$, and vertex x_j in K_X is adjacent to all the vertices of K_{c_i} if and only if the element x_j of X belongs to the set c_i of C. Clearly, this construction can be done in polynomial time from the instance (X, C). Now, since for each $1 \leq i \leq m$, all the $3q$ vertices of the clique K_{c_i} are true twins, by Lemma 3.1 they must belong to the same clique in an optimal partition S for the optimization version of the cluster deletion problem. Therefore, we can contract each clique K_{c_i}, with $1 \leq i \leq m$, into a single vertex c_i and replace each subset of $3q$ edges between K_{c_i} and the vertex $x_j \in K_X$ by a single edge with weight $3q$, for each x_j in c_i. Hence, we obtain the weighted split graph constructed in the proof of Theorem 2.1. This shows (a) that graph G is a P_5-free chordal graph since split graphs also are and the true twins contraction neither eliminates chordless cycles nor induces paths on five vertices; and (b) that the problem is indeed NP-complete. □

4. Block graphs

A graph G is a *block* graph if it is a connected graph and every *block* (i.e. maximal 2-connected component) is a clique. Block graphs form a subclass of chordal graphs.

The first result in this section concerns weighted 1-split graphs, a special subclass of block graphs. First, we will show how to use submodular function minimization in order to solve the cluster deletion problem on 1-split graphs. Then we will explain how to reduce the problem on weighted block graphs to the problem on weighted 1-split graphs.

Given a finite nonempty set V of cardinality n, a function f defined on all the subsets of V is called *submodular* if it satisfies $f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y)$, for every $X, Y \subseteq V$. In [15] and [21], the authors present combinatorial

polynomial-time algorithms for finding a minimizer of a general submodular function provided an oracle for evaluating the function value is available. The number of oracle calls is bounded by a polynomial in the size of the underlying set.

Theorem 4.1 *(Polynomiality on weighted* 1*-split graphs). The cluster deletion problem is polynomial-time solvable for weighted* 1*-split graphs.*

Proof. Let G be a 1-split graph with split partition (K, I). Consider an optimal solution S of the cluster deletion problem. Clearly, S contains at most one clique composed only of vertices of K, otherwise a new partition obtained by replacing two such cliques by their union would contradict the solution's optimality. If this clique exists, denote it by K_1. All other cliques in S are either isolated vertices of I or cliques of size 2 with one vertex in K and one vertex in I. Also, if $\{u, v\} \in S$ with $u \in K$, $v \in I$, then necessarily $w(uv) \geq w(uw)$, for all $w \in N(v) \cap I$, since minimizing the weight of the external edges of the clique partition is equivalent to maximizing the weight of the internal ones. We can then preprocess the graph, identifying a subset of vertices of I that will be singletons in the solution and leaving a graph with vertex partition (K, I') in which each vertex v of K has at most one neighbor u in I' (one of the neighbors of v maximizing the weight of the edge vu). Moreover, by definition of 1-split graph, each vertex of I has at most one neighbor in K.

If the subset K_0 of vertices of K having no neighbors in I' is nonempty, it will be part of the clique K_1 in the solution. We name the set $K \setminus K_0$ as $\{v_1, \ldots, v_r\}$ and their respective neighbors in I' as $\{u_1, \ldots, u_r\}$. A candidate solution must be $\{\{v_i, u_i\}\}_{i \notin R} \cup \{K_1 \cup \{v_i : i \in R\}\} \cup \{\{u_i\}\}_{i \in R}$, where R is a subset of $\{1, \ldots, r\}$ and is totally determined by that subset R.

The subset R is not necessarily a proper subset of $\{1, \ldots, r\}$ and may be empty. In what follows we will prove that the function f that assigns to R the weight difference between the candidate solution associated with R and the solution associated with the empty set is submodular, and thus the set R that minimizes that function can be found in polynomial time.

The function f can be computed in polynomial time for a subset R, and is defined as

$$f(R) = \sum_{i \in R} w(v_i u_i) - \sum_{i,j \in R} w(v_i v_j) - \sum_{i \in R, z \in K_0} w(v_i z)$$

Let $R, T \subseteq \{1, \ldots, r\}$. We will show that $f(R) + f(T) \geq f(R \cup T) + f(R \cap T)$ provided that the weights of the edges joining two vertices of K are nonnegative, which holds under our assumptions. It is easily seen that

$$\sum_{i \in R} w(v_i u_i) + \sum_{i \in T} w(v_i u_i) = \sum_{i \in R \cup T} w(v_i u_i) + \sum_{i \in R \cap T} w(v_i u_i)$$

and that

$$\sum_{i \in R, z \in K_0} w(v_i z) + \sum_{i \in T, z \in K_0} w(v_i z) = \sum_{i \in R \cup T, z \in K_0} w(v_i z) + \sum_{i \in R \cap T, z \in K_0} w(v_i z).$$

We therefore have to show that

$$\sum_{i,j \in R} w(v_i v_j) + \sum_{i,j \in T} w(v_i v_j) \leq \sum_{i,j \in R \cup T} w(v_i v_j) + \sum_{i,j \in R \cap T} w(v_i v_j).$$

The inequality holds because

$$\sum_{i,j \in R \cup T} w(v_i v_j) + \sum_{i,j \in R \cap T} w(v_i v_j) - \sum_{i,j \in R} w(v_i v_j) - \sum_{i,j \in T} w(v_i v_j) = \sum_{i \in R \setminus T, j \in T \setminus R} w(v_i v_j) \geq 0. \qquad \square$$

If the weight of the internal edges is 1, the algorithm is very simple. We denote the vertices in $K \setminus K_0$ as $\{v_1, \ldots, v_r\}$ and their respective neighbors in I as $\{u_1, \ldots, u_r\}$, in such a way that $w(v_1 u_1) \leq \cdots \leq w(v_r u_r)$. Then the only sets that are candidates for minimizing f are the empty set and the sets $\{1, \ldots, j\}$, for $1 \leq j \leq r$.

Based on the solution of cluster deletion for weighted 1-split graphs, we now solve the problem for weighted block graphs.

Theorem 4.2 *(Polynomiality on weighted block graphs). The cluster deletion problem is polynomial-time solvable for weighted block graphs.*

Proof. Let $G = (V, E)$ be a weighted block graph. An *end-block* of a graph is a block containing exactly one cut-vertex. It is known that every connected graph that is not 2-connected has an end-block. Inductively, the blocks of G can be enumerated as $B_1, \ldots, B_r$ in such a way that B_i is an end-block of the graph induced by $B_i \cup B_{i+1} \cup \cdots \cup B_r$. We will process the blocks in that order by solving a subproblem at each iteration and thus reduce the graph to a simpler one. Then we will reconstruct the solution for the original graph based on the solution of each subproblem and the recursive solution of the reduced graph.

Given the order in which the blocks were chosen, when we process block B_i for $i < r$ it will have exactly one cut-vertex v joining it with the non-processed blocks. The graph G_0 will be G and we will create a graph G_i ($i \geq 1$) from G_{i-1} by replacing the connected component of $G_{i-1} \setminus \{v\}$ that contains $B_i \setminus \{v\}$ with a vertex u adjacent to v. We will then assign a suitable weight to edge vu. Inductively, this means that when block B_i is processed, the connected component H of $G_{i-1} \setminus \{v\}$ that contains $B_i \setminus \{v\}$ is a 1-split graph, and so is the graph induced by $V(H) \cup \{v\}$. We will define $w(uv) = \sum_{y \in H, vy \in E} w(vy) + w(S_H) - w(S_{H \cup \{v\}})$, where $S_{H \cup \{v\}}$ and S_H are optimal solutions for the cluster deletion problem on the 1-split graph induced by $V(H) \cup \{v\}$ and H, respectively. If the weight $w(uv)$ is negative we do not create the edge since no optimal solution will use it (i.e., no optimal solution will have a clique containing both v and vertices of H). Note that we can find $S_{H \cup \{v\}}$ and S_H by Theorem 4.1.

Claim 4.3. *Let S_i be an optimal solution of the cluster deletion problem for G_i. If $\{u, v\} \in S_i$, then $S_i \setminus \{\{u, v\}\} \cup S_{H \cup \{v\}}$ is an optimal solution for G_{i-1}. If $\{u, v\} \notin S_i$, then $S_i \setminus \{\{u\}\} \cup S_H$ is an optimal solution for G_{i-1}.*

Let S_{i-1} be an optimal solution of the cluster deletion problem for G_{i-1}. Let S^1_{i-1} be the subset of cliques of S_{i-1} containing vertices of H and $S^2_{i-1} = S_{i-1} \setminus S^1_{i-1}$. Then v either does or does not belong to a clique in S^1_{i-1}. Since v is a cut-vertex of G_{i-1} it is clear that under the first alternative S^1_{i-1} is an optimal solution for the cluster deletion problem on the graph induced by $V(H) \cup \{v\}$ while under the second, S^1_{i-1} is an optimal solution for the cluster deletion problem on graph H. Also, under the first alternative $S^2_{i-1} \cup \{v, u\}$ will be a feasible solution for G_i with weight $w(S_{i-1}) - w(S_{H \cup \{v\}})$ while under the second, $S^2_{i-1} \cup \{u\}$ will be a feasible solution for G_i with weight $w(S_{i-1}) - w(S_H) - \sum_{y \in H, vy \in E} w(vy) + w(uv) = w(S_{i-1}) - w(S_{H \cup \{v\}})$. Therefore, if S_i is an optimal solution of the cluster deletion problem for G_i, then $w(S_i) \leq w(S_{i-1}) - w(S_{H \cup \{v\}})$, which implies that $w(S_i) + w(S_{H \cup \{v\}}) \leq w(S_{i-1})$.

We now prove that the weight of the solutions proposed for G_{i-1} under each alternative is exactly $w(S_i) + w(S_{H \cup \{v\}})$, making them both optimal.

If $\{u, v\} \in S_i$, it is easily seen that $w(S_i \setminus \{\{u, v\}\} \cup S_{H \cup \{v\}}) = w(S_i) + w(S_{H \cup \{v\}})$. If $\{u, v\} \notin S_i$, then, when considering $S_i \setminus \{\{u\}\} \cup S_H$ in G_{i-1}, we do not have to delete the edge uv but do have to delete every edge joining v with vertices in H so that $w(S_i \setminus \{\{u\}\} \cup S_H) = w(S_i) - w(uv) + \sum_{y \in H, vy \in E} w(vy) + w(S_H) = w(S_i) + w(S_{H \cup \{v\}})$. This ends the proof of this claim. $\diamond$

When block B_r is processed, the graph G_{r-1} is a 1-split graph so we can also apply the algorithm of Theorem 4.1 in order to obtain an optimal partition for G_{r-1}. Claim 4.3 describes an optimal partition of G_{i-1} in terms of an optimal partition of G_i. Applying this claim iteratively for $i = r-1, \ldots, 1$, we construct an optimal solution for the graph $G_0 = G$. $\square$

Notice that if the graph G is unweighted, the 1-split graphs in which we need to solve the subproblems have a weight of 1 on every internal edge. In this case, as we noted before Theorem 4.2, the algorithm is very simple.

5. Interval graphs

Another interesting subclass of chordal graphs is the class of interval graphs. A graph G is an *interval graph* if G is the intersection graph of a finite family of intervals of the real line, and it is a *proper interval graph* if it admits an intersection model in which no interval properly contains another. A *unit interval graph* is the intersection graph of a finite family of intervals of the real line, all of the same length. Proper interval graphs and unit interval graphs are equivalent classes and are also equivalent to the class of claw-free interval graphs [20] (the *claw* is the complete bipartite graph $K_{1,3}$).

A restricted subclass of unit interval graphs is the class of *paths of cliques*. A graph is a *path of cliques* if after contracting true twins into a single vertex, the resulting graph is a single path. In other words, its vertex set can be partitioned into sets $A_1, \ldots, A_n$ in such a way that any pair of vertices v, w such that $v \in A_i$ and $w \in A_j$ is adjacent if and only if either $i = j$ or $i = j + 1$ or $i = j - 1$. Paths of cliques are also known in the literature as *line graphs of multipaths*.

In a clustering context, if the measured data arrive during a time-line as a sequence of sets and it is desired to cluster the data on the basis of both a similarity function, defined by intrinsic properties of the data, and closeness in time, defined as arriving either in the same set or in consecutive sets, these paths of cliques will arise.

In what follows, we prove a result using an approach similar to that used for Theorem 4.1. We begin by defining an initial solution, then represent every solution by a subset of a set, and finally show that the function that assigns to every subset the weight difference between its associated solution and the initial one is submodular.

Theorem 5.1 *(Polynomiality on weighted paths of cliques). The cluster deletion problem is polynomial-time solvable for weighted paths of cliques.*

Proof. Let $A_1, \ldots, A_n$ be the vertex set partition of a path of cliques $G = (V, E)$. Without loss of generality, we may assume that n is even, if necessary adding a set A_{n+1} with only one vertex adjacent to every vertex in A_n with edges of zero weight. Let $n = 2r$ and define the initial solution as the cliques $A_{2k-1} \cup A_{2k}$ for $1 \leq k \leq r$. Every vertex of A_i, for $1 < i < n$, has two possibilities: to be either part of a clique contained in $A_{i-1} \cup A_i$ or part of a clique contained in $A_i \cup A_{i+1}$. Hence, any

solution is completely defined by the subset S of $V \setminus (A_1 \cup A_n)$ that is moved from the initial solution to a clique contained in $A_{2j} \cup A_{2j+1}$, for some integer $1 \leq j \leq r-1$. We define $f(S)$ for a subset S of $V \setminus (A_1 \cup A_n)$ as the difference between the weight of the solution associated with S and the weight of the initial solution. The optimal solution will be given by the subset S that minimizes the function f. We now show that S is a submodular function, implying that a minimizer can be found in strongly polynomial time.

The function f can be expressed as $f(S) = \sum_{1 \leq k \leq r} f^k(S)$, where

$$f^k(S) = \sum_{\substack{v \in S \cap A_{2k} \\ u \in A_{2k-1}}} w(vu) + \sum_{\substack{v \in S \cap A_{2k-1} \\ u \in A_{2k}}} w(vu) + \sum_{\substack{v \in S \cap A_{2k-1} \\ u \in A_{2k-1} \setminus S}} w(vu) + \sum_{\substack{v \in S \cap A_{2k} \\ u \in A_{2k} \setminus S}} w(vu) - \sum_{\substack{v \in S \cap A_{2k} \\ u \in S \cap A_{2k+1}}} w(vu)$$
$$- \sum_{\substack{v \in S \cap A_{2k-1} \\ u \in S \cap A_{2k}}} w(vu).$$

To simplify this expression we let $A_{2r+1} = \emptyset$. Note that the last term in the sum prevents double counting by the first two terms.

For each value of k, define $f_1^k(S), \ldots, f_6^k(S)$ as the six terms of $f^k(S)$. It can easily be seen that for $i = 1, 2$ and S, T subsets of $V \setminus (A_1 \cup A_n)$, it holds $f_i^k(S) + f_i^k(T) = f_i^k(S \cup T) + f_i^k(S \cap T)$. We will prove that for $i = 3, \ldots, 6$, $f_i^k(S) + f_i^k(T) \geq f_i^k(S \cup T) + f_i^k(S \cap T)$.

For S, T subsets of $V \setminus (A_1 \cup A_n)$, and by decomposing S as $(S \setminus T) \cup (S \cap T)$ (resp. T as $(T \setminus S) \cup (S \cap T)$); $V \setminus S$ as $(T \setminus S) \cup (V \setminus (S \cup T))$ (resp. $V \setminus T$ as $(S \setminus T) \cup (V \setminus (S \cup T))$); $S \cup T$ as $(S \setminus T) \cup (T \setminus S) \cup (S \cap T)$; and $V \setminus (S \cap T)$ as $(T \setminus S) \cup (S \setminus T) \cup (V \setminus (S \cup T))$, it can be seen that

$$f_3^k(S) + f_3^k(T) - f_3^k(S \cup T) - f_3^k(S \cap T) = 2 \sum_{\substack{v \in (S \setminus T) \cap A_{2k-1} \\ u \in (T \setminus S) \cap A_{2k-1}}} w(vu) \geq 0$$

because the weights are nonnegative, so f_3^k is submodular. The proof for f_4^k is identical.

Recall that $f_5^k(S) = -\sum_{v \in S \cap A_{2k}, u \in S \cap A_{2k+1}} w(vu)$. By using again the decomposition S as $(S \setminus T) \cup (S \cap T)$ (resp. T as $(T \setminus S) \cup (S \cap T)$); and $S \cup T$ as $(S \setminus T) \cup (T \setminus S) \cup (S \cap T)$, it can be seen that

$$f_5^k(S) + f_5^k(T) - f_5^k(S \cup T) - f_5^k(S \cap T) = \sum_{\substack{v \in (S \setminus T) \cap A_{2k} \\ u \in (T \setminus S) \cap A_{2k+1}}} w(vu) + \sum_{\substack{v \in (T \setminus S) \cap A_{2k} \\ u \in (S \setminus T) \cap A_{2k+1}}} w(vu) \geq 0$$

because the weights are nonnegative, so f_5^k is submodular. The proof for f_6^k is identical. Finally, the sum of submodular functions is submodular, and this completes the proof. □

A split graph is called *complete* if each vertex of the independent set is adjacent to all the vertices of the clique. By modifying the proof of Theorem 2.1 slightly, we can prove the following.

Theorem 5.2 *(NP-completeness on weighted complete split graphs). The cluster deletion problem is NP-complete for weighted complete split graphs even if the weight of all the internal edges of the clique is* 1.

Proof. We again use a reduction of the X3C problem. Let $X = \{x_1, \ldots, x_{3q}\}$ and $C = \{c_1, \ldots, c_m\}$ be an instance of the X3C problem where each element $c_i \in C$ is a 3-element subset of X, with $m \geq q \geq 1$. We want to know if there exists a subset $C' \subseteq C$ with size q such that each element in X belongs to exactly one of the elements in C'. We construct an edge-weighted complete split graph $G = (K_X \cup I_C, E)$, where K_X induces a clique with $3q$ vertices, I_C induces an independent set with m vertices, and each vertex of C is adjacent to every vertex in K_X. To each element $x_i \in X$ we associate a vertex x_i in K_X, and to each 3-subset $c_j \in C$ we associate a vertex $c_j \in I_C$. The weight of the edges with both endpoints in K_X is 1 and the weight of an edge $x_i c_j$ with x_i in K_X and c_j in I_C is $\beta = \binom{3q}{2} + 3m(q-1) + 1$ if the element x_i of X belongs to the set c_j of C, and 1 otherwise. Clearly, its construction can be done in polynomial time from (X, C). Let $W = \binom{3q}{2} - 3q + 3(m-q)\beta + 3m(q-1)$. We will show that there exists a subset $C' \subseteq C$, with $|C'| = q$, exactly covering X if and only if G admits a clique partition where the sum of the weights of the edges outside the cliques is at most W. In other words, there exists a solution for the X3C problem if and only if there exists a clique partition of G with a weight of at most W.

Assume first that there exists $C' \subseteq C$, with $C' = \{c'_1, \ldots, c'_q\}$ such that $c'_i \cap c'_j = \emptyset$ whenever $i \neq j$, and $\bigcup_{c'_j \in C'} c'_j = X$. We construct a clique partition of G as follows: for each $c'_j \in C'$, with $c'_j = \{x_{j_1}, x_{j_2}, x_{j_3}\}$, choose the clique $\{c'_j, x_{j_1}, x_{j_2}, x_{j_3}\}$ in G. Each one of the remaining $m - q$ vertices in I_C forms a clique of size 1. It is easily seen that the sum of the edge weights outside those cliques is exactly equal to W.

Now assume conversely that G admits a clique partition with a weight of at most W. We will prove that there exists $C' \subseteq C$, with $|C'| = q$, such that C' is an exact cover for X. To do this, we first analyze the structure of such a partition. Note

that β is greater than the number of edges of weight 1 in G, so a solution with weight W has as its external edges exactly $3(m-q)$ edges of weight β. It cannot have less than $3(m-q)$ external edges of weight β because each K_X vertex belongs to a clique of the partition with at most one vertex of I_C. So, every vertex of K_X belongs to a clique with exactly one vertex of I_C, and is joined to that vertex by an edge of weight β. This means that all the edges of weight 1 and one endpoint in I_C are external edges and that each clique of the partition contains at most three vertices of K_X. By the value of W we can see that each clique should contain exactly three vertices of K_X, and this solution, following the general lines of the proof of Theorem 2.1, is a solution of the X3C instance, thus ending the proof of the present theorem. □

Complete split graphs are also interval graphs but not (in general) unit interval graphs. Indeed, if the size of the independent set is at least 3 and the clique is nonempty, we obtain a *claw* (the complete bipartite graph $K_{1,3}$), that is not a unit interval graph. We therefore have the following corollary.

Corollary 5.3 *(NP-completeness on weighted interval graphs). The cluster deletion problem is NP-complete for weighted interval graphs.*

Turning now to the case of unweighted unit interval graphs, we will show that the cluster deletion problem is polynomial-time solvable. But first we must state some known results and prove a lemma describing the structure of an optimal solution.

Theorem 5.4. *(See Roberts, 1969 [20].) A graph G is a unit interval graph if and only if its vertices can be linearly ordered such that, for each clique M of G, the vertices contained in M are consecutive.*

Such an ordering is called a *canonical ordering* of the vertices.

Lemma 5.5 *(Consecutiveness for unweighted unit interval graphs). Let G be an unweighted unit interval graph and $v_1, \ldots, v_n$ be a canonical ordering of the vertices of G. Then there is an optimal solution of the cluster deletion problem for G such that each clique of the solution consists of consecutive vertices in that ordering.*

Proof. Let us define, for each clique B of an optimal solution S, $m(B) = \max\{j : v_j \in B\}$. Now let $B_1, \ldots, B_k$ be the cliques of the solution such that $m(B_i) \leq m(B_{i+1})$, for $i = 1, \ldots, k-1$. Suppose that not all the cliques consist of consecutive vertices, and let i be a minimum such that either $i < m(B_1)$ and $v_i \notin B_1$, or $m(B_{j-1}) < i < m(B_j)$ but $v_i \notin B_j$, for some j. Let j' be such that $v_i \in B_{j'}$. Then, by the choice of i, all the vertices of $B_{j'}$ have subindex greater than i and $j' > j$ so that $m(B_{j'}) > m(B_j)$. Since v_i is adjacent to $v_{m(B_{j'})}$ and G is a proper interval graph with canonical ordering $v_1, \ldots, v_n$, the vertices $v_i, \ldots, v_{m(B_{j'})}$ form a clique, and in particular, $B_{j'} \cup \{v_{m(B_j)}\}$ is a clique and v_i is adjacent to $v_{m(B_j)}$. So, independently of i being greater or less than the minimum index of a vertex in B_j, $B_j \cup \{v_i\}$ is a clique. Then, either $|B_j| \geq |B'_j|$ and $S \setminus \{B_j, B_{j'}\} \cup \{B_j \cup \{v_i\}, B_{j'} \setminus \{v_i\}\}$ is a clique partition whose weight is less than $w(S)$, or $|B_j| < |B'_j|$ and $S \setminus \{B_j, B_{j'}\} \cup \{B_j \setminus \{v_{m(B_j)}\}, B_{j'} \cup \{v_{m(B_j)}\}\}$ is a clique partition whose weight is less than $w(S)$, which in both cases are contradictions. □

Theorem 5.6 *(Polynomiality on unweighted unit interval graphs). The cluster deletion problem can be solved in polynomial time on unweighted unit interval graphs.*

Proof. Using Lemma 5.5, we can easily develop a dynamic programming algorithm. For $i = 0, 1, \ldots, n$, let $f(i)$ be the value of an optimal cluster deletion solution for the subgraph of G induced by $v_1, \ldots, v_i$. Then $f(0) = f(1) = 0$ and, for $i > 1$, $f(i)$ is the minimum, over all j such that $\{v_j, \ldots, v_i\}$ is a clique (i.e., either $j = i$ or $v_j v_i \in E(G)$) of $f(j-1)$ plus the amount of vertices joining $\{v_1, \ldots, v_{j-1}\}$ with $\{v_j, \ldots, v_i\}$ (which is 0 if $j = 1$). By keeping the number j that yields the minimum $f(i)$, we can also reconstruct the partition itself. □

A very similar approach is used in [19] to solve the *cluster editing* problem on unit interval graphs, where edge insertions and deletions can be performed in order to obtain a cluster graph.

General interval graphs do not have the same clique structure as unit interval graphs and for weighted unit interval graphs, Lemma 5.5 does not hold. An example of this is the graph P_6^2, whose vertices are $v_1, \ldots, v_6$ and v_i is adjacent to v_j if and only if $1 \leq |i-j| \leq 2$. It is easily seen that the only possible canonical orderings for P_6^2 are $v_1, \ldots, v_6$ or $v_6, v_5, \ldots, v_1$. Let w be defined on the edges of P_6^2 such that $w(v_2v_4) = w(v_3v_5) = 100$ and $w(e) = 1$ for every other edge e. Any solution to the cluster deletion problem that does not contain $\{v_2, v_4\}$ and $\{v_3, v_5\}$ as cliques has a weight of at least 100 so the optimal solution is to have $\{v_2, v_4\}$, $\{v_3, v_5\}$ and isolated vertices with a weight of 7.

The example shows that the idea behind Theorem 5.6 cannot be generalized in a straightforward manner. Therefore, the computational complexity of the cluster deletion problem on unweighted interval graphs and on weighted unit interval graphs remains unknown.

Complete split graphs are also cographs (i.e., P_4-free graphs). The cluster deletion problem on unweighted cographs was solved in polynomial time by Gao et al. in [11]. As a corollary of Theorem 5.2, we have the following complexity result for the weighted case.

Corollary 5.7 *(NP-completeness on weighted cographs). The cluster deletion problem is NP-complete for weighted cographs.*

6. Further results and open problems

In Theorem 2.1 we showed that the cluster deletion problem is NP-complete for weighted 3-split graphs even if the weight of all the internal edges of the clique is 1 and the weight of the edges between the clique and the independent set is uniform. We have seen also in Theorem 4.1 that the cluster deletion problem is polynomial-time solvable for weighted 1-split graphs. As for 2-split graphs, we will now show that under the conditions of Theorem 2.1 (the weight of all the internal edges of the clique is 1 and the weight of the edges between the clique and the independent set is uniform), the problem is polynomial-time solvable.

Theorem 6.1 *(Polynomiality on restricted weighted 2-split graphs). The cluster deletion problem is polynomial-time solvable for weighted 2-split graphs if the weight of all the internal edges of the clique is* 1 *and the weight of the edges between the clique and the independent set is uniform.*

Proof. Let G be a 2-split graph with split partition (K, I) and β be the weight of the edges between K and I. Let us create a graph G' with vertex set K', where K' is the subset of vertices of K that have at least one neighbor in I, such that two vertices are adjacent in G' if they have a common neighbor in I in the graph G. Minimizing the sum of the weights of the external edges of a clique partition of G is equivalent to maximizing the sum of the weights of the internal edges of the partition. Let S be an optimal solution to the cluster deletion problem in G. There are four possible classes of cliques in S: those containing one vertex of I and two vertices of K, those containing one vertex of I and one vertex of K, those that consist of a single vertex of I, and those completely included in K, and by optimality there is at most one clique that is completely included in K. Suppose S contains a cliques composed of one vertex of I and two vertices of K, and b cliques composed of one vertex of I and one vertex of K. Then the sum of the weights of the internal edges of S is $a(2\beta+1)+b\beta+c(c-1)/2$ $(*)$, where $0 \leq a \leq \nu(G')$ ($\nu(G')$ is the value of a maximum matching of G'), $0 \leq b \leq |K'|-2a$, and $c = |K|-2a-b$. It is easily seen that in an optimal solution, either $a = \nu(G')$ or $b = 0$. In the first case, after the substitution $a = \nu(G')$, the coefficient of b in the expression $(*)$ is positive, so the maximum is attained either by $b = 0$ or by $b = |K'|-2\nu(G')$. In the second case, after the substitution $b = 0$, the coefficient of a in the expression $(*)$ is positive, so the maximum is attained either by $a = 0$ or by $a = \nu(G')$. The problem is therefore reduced to solving maximum matching in G' and then computing the value of $(*)$ for the three possible optimal solutions $a = 0, b = 0$; $a = \nu(G'), b = 0$; and $a = \nu(G'), b = |K'|-2\nu(G')$. □

We leave as an open problem the determination of the computational complexity of the cluster deletion problem in general weighted 2-split graphs, or in 2-split graphs when the weight of all the internal edges of the clique is 1 but the weights of the edges between the clique and the independent set is arbitrary and not necessarily uniform.

Acknowledgements

The authors are indebted to the anonymous referees for their insightful comments, corrections, and observations that helped to improve this paper. We also want to thank Kenneth Rivkin for his many useful suggestions.

References

[1] N. Bansal, A. Blum, S. Chawla, Correlation clustering, Mach. Learn. 56 (1–3) (2004) 89–113, Extended abstract appeared in FOCS 2002, pp. 238–247.
[2] J.R.S. Blair, B. Peyton, An introduction to chordal graphs and clique trees, in: Graph Theory and Sparse Matrix Computation, in: The IMA Volumes in Mathematics and Its Applications, vol. 56, 1993, pp. 1–29.
[3] S. Böcker, P. Damaschke, Even faster parametrized cluster deletion and cluster editing, Inform. Process. Lett. 111 (2011) 717–721.
[4] F. Bonomo, G. Durán, A. Napoli, 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, Inform. Process. Lett. 115 (2015) 600–603.
[5] M. Charikar, V. Guruswami, A. Wirth, Clustering with qualitative information, in: Proc. of 44th Annu. IEEE Symp. Foundations of Computer Science, FOCS 2003, 2003, pp. 524–533.
[6] P. Damaschke, O. Mogren, Editing simple graphs, J. Graph Algorithms Appl. 18 (4) (2014) 557–576.
[7] E.D. Demaine, D. Emanuel, A. Fiat, N. Immorlica, Correlation clustering in general weighted graphs, Theoret. Comput. Sci. 361 (2006) 172–187.
[8] A. Dessmark, A. Lingas, E.M. Lundell, M. Persson, J. Jansson, On the approximability of maximum and minimum edge clique partitions problems, Internat. J. Found. Comput. Sci. 18 (2) (2007) 217–226.
[9] J. Edmonds, Maximum matching and a polyhedron with 0, 1-vertices, J. Res. Natl. Bur. Stand., B Math. Math. Phys. 69B (1965) 125–130.
[10] S. Földes, P.L. Hammer, Split graphs, in: Proc. of 8th South-Eastern Conference on Combinatorics, Graph Theory and Computing, in: Congressus Numerantium, vol. 19, 1977, pp. 311–315.
[11] Y. Gao, D.R. Hare, J. Nastos, The cluster deletion problem for cographs, Discrete Math. 313 (2013) 2763–2771.

[12] M.R. Garey, D.S. Johnson, Computers and Intractability, Freeman, San Francisco, 1979.
[13] P. Hansen, B. Jaumard, Cluster analysis and mathematical programming, Math. Program. 79 (1997) 191–215.
[14] J. Hartigan, Clustering Algorithms, Wiley, New York, 1975.
[15] S. Iwata, L. Fleischer, S. Fujishige, A combinatorial strongly polynomial algorithm for minimizing submodular functions, J. ACM 48 (2001) 761–777.
[16] B. Kaba, N. Pinet, G. Lelandais, A. Sigayret, A. Berry, Clustering gene expression data using graph separators, In Silico Biol. 7 (4–5) (2007) 433–452.
[17] C. Komusiewicz, J. Uhlmann, Cluster editing with locally bounded modifications, Discrete Appl. Math. 160 (15) (2012) 2259–2270.
[18] I. Kovac, I. Seleceniova, M. Steinova, On the clique editing problem, in: Proc. of MFCS 2014, in: LNCS, vol. 8635, 2014, pp. 469–480.
[19] B. Mannaa, Cluster editing problem for points on the real line: a polynomial time algorithm, Inform. Process. Lett. 1610 (2010) 961–965.
[20] F.S. Roberts, Indifference graphs, in: F. Harary (Ed.), Proof Techniques in Graph Theory, Academic Press, 1969, pp. 139–146.
[21] A. Schrijver, A combinatorial algorithm minimizing submodular functions in strongly polynomial time, J. Combin. Theory Ser. B 80 (2000) 346–355.
[22] R. Shamir, R. Sharan, D. Tsur, Cluster graph modification problems, Discrete Appl. Math. 144 (1–2) (2004) 173–182.
[23] D.B. West, Introduction to Graph Theory, 2nd edition, Prentice-Hall, 2001.