



When Can Cluster Deletion with Bounded Weights Be Solved Efficiently?

Jaroslav Garvardt  

Institute of Computer Science, Friedrich Schiller University Jena, Germany

Christian Komusiewicz  

Institute of Computer Science, Friedrich Schiller University Jena, Germany

Nils Morawietz  

Institute of Computer Science, Friedrich Schiller University Jena, Germany

Abstract

In the NP-hard WEIGHTED CLUSTER DELETION problem, the input is an undirected graph $G = (V, E)$ and an edge-weight function $\omega : E \rightarrow \mathbb{N}$, and the task is to partition the vertex set V into cliques so that the total weight of edges in the cliques is maximized. Recently, it has been shown that WEIGHTED CLUSTER DELETION is NP-hard on some graph classes where CLUSTER DELETION, the special case where every edge has unit weight, can be solved in polynomial time. We study the influence of the value t of the largest edge weight assigned by ω on the problem complexity for such graph classes. Our main results are that WEIGHTED CLUSTER DELETION is fixed-parameter tractable with respect to t on graph classes whose graphs consist of well-separated clusters that are connected by a sparse periphery. Concrete examples for such classes are split graphs and graphs that are close to cluster graphs. We complement our results by strengthening previous hardness results for WEIGHTED CLUSTER DELETION. For example, we show that WEIGHTED CLUSTER DELETION is NP-hard on restricted subclasses of cographs even when every edge has weight 1 or 2.

2012 ACM Subject Classification Theory of computation \rightarrow Graph algorithms analysis

Keywords and phrases Graph clustering, split graphs, cographs, parameterized complexity

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2024.32

Funding *Jaroslav Garvardt*: Supported by the Carl Zeiss Foundation, Germany, within the project “Interactive Inference”.

1 Introduction

Graph-based clustering is one of the core applications of graphs in computer science. This has led to a vast number of different algorithms and problem formulations for this task. One of the most fundamental problem formulations in this context is CLUSTER DELETION. In this problem, we are given an undirected graph G and ask for a partition of its vertex set into cliques that maximizes the total number of edges inside the cliques. CLUSTER DELETION is NP-hard [26] which has motivated the application of algorithmic approaches for hard problems to CLUSTER DELETION. In particular, the parameterized complexity of CLUSTER DELETION has been intensively studied [3, 12, 14, 16, 19, 22, 28].

Another closely related line of research is to study the complexity of CLUSTER DELETION on restricted classes of input graphs.¹ On the positive side, it was shown, for example, that CLUSTER DELETION can be solved in polynomial time on subcubic graphs [22], on cographs [13], on the more general class of P_4 -sparse graphs [4], on interval graphs [23], and on several classes that generalize split graphs [23]. On the negative side, it was shown, for example, that CLUSTER DELETION remains NP-hard on planar graphs [15], on P_5 -free

¹ For a definition of all considered graph classes, see Section 2.



chordal graphs [5], and on C_4 -free graphs with maximum degree 4 [22]. For graph classes defined by a single forbidden induced subgraph with at most four vertices a dichotomy into NP-hard and polynomial-time solvable cases is known [16].

All in all, the complexity of CLUSTER DELETION is fairly well-understood by now. In many applications, however, we are interested in the edge-weighted version of the problem where the aim is to maximize the total weight of the edges inside the clusters. This problem, called WEIGHTED CLUSTER DELETION, turns out to be NP-hard for many graph classes where CLUSTER DELETION is polynomial-time solvable. For example, it has been shown that WEIGHTED CLUSTER DELETION is NP-hard on cographs [5] and on split graphs [5]. Perhaps surprisingly, the problem is even NP-hard on split graphs where the independent set contains three vertices and every vertex of the independent set is adjacent to every vertex of the clique [19]. A closer inspection of the NP-hardness proofs shows, however, that the reductions construct instances where the maximum edge weight is unbounded. For example, in the above-mentioned reduction for the restricted class of split graphs with a periphery of constant size, the maximum edge weight is n^2 where n is the number of vertices in the constructed instance. In the vein of the deconstruction of hardness proofs [21], this observation begs the question of whether one can solve instances with bounded weights more efficiently. In particular, we would like to either identify graph classes on which parameterizing the maximum edge weight t in the input graph leads to FPT-algorithms or strengthen previous NP-hardness results to also hold in the case when the maximum edge weight is constant.

Our Results. Our results are summarized in Table 1. In a nutshell, we show that we obtain FPT-algorithms with respect to t on graph classes where large cliques are rather well-separated from each other. The first example of such an FPT-algorithm for t is obtained for the class of split graphs. Another class for which we can show fixed-parameter tractability is the class of almost cluster graphs. More precisely, we provide an FPT-algorithm for the combined parameter $\text{cvd} + t$, where cvd is the vertex deletion distance of the input graph to cluster graphs. We then generalize these results to obtain algorithms for graphs with a bounded treewidth modulator to cliques (this generalizes split graphs) and for graphs which are almost split cluster graphs, that is, for the combined parameter $\text{scvd} + t$, where scvd is the vertex deletion distance to graphs where every connected component is a split graph.

On the negative side, we show that even very restricted cases of WEIGHTED CLUSTER DELETION remain NP-hard when all edges have weight 1 or 2. More precisely, we show this hardness for complete tripartite graphs and complete unipolar graphs, a subclass of the previously considered stable-like and laminar-like graphs which allow for polynomial-time algorithms in the unweighted case [23].

In addition, we show that it is presumably impossible to strengthen our FPT-algorithms for the parameter t to polynomial-size problem kernels. Finally, we show that WEIGHTED CLUSTER DELETION is NP-hard on proper interval graphs, albeit with unbounded edge weights.

Due to lack of space, some proofs, marked by (*), are deferred to a full version of this work.

2 Preliminaries and Basic Observations

Notation. For $a, b \in \mathbb{N}$ with $a \leq b$ we write $[a, b]$ for the set $\{i \in \mathbb{N} \mid a \leq i \leq b\}$ and $[n] := [1, n]$. We consider undirected graphs $G = (V, E)$ with vertex set V and edge set E . The *neighborhood* of a vertex $v \in V$ is defined as $N_G(v) := \{u \in V \mid \{u, v\} \in E\}$ and for a set

■ **Table 1** An overview over the (parameterized) complexity of WCD on the considered graph classes. Here, nd denotes the neighborhood diversity of the input graph, and cvd denotes its cluster vertex deletion number. For the first three classes, FPT refers to parameterization by maximum edge weight t plus either nd or cvd , for the other classes FPT refers to parameterization by t .

Graph class	Unweighted	Weighted	Bounded Max Weight
$\text{nd} \leq 2$	P [19]	NP-h [19]	FPT Corollary 4.8
$\text{nd} \geq 3$	FPT [19]	NP-h [19]	NP-h Theorem 3.6
bounded cvd	FPT [14]	NP-h (for $\text{cvd} = 2$) [19]	FPT Theorem 5.3
interval	P [23]	NP-h [5]	NP-h Proposition 3.4
proper interval	P [5]	NP-h Theorem 3.5	?
complete unipolar	P [23]	NP-h [19]	NP-h Proposition 3.4
co-graph	P [13]	NP-h [5]	NP-h Proposition 3.4
split	P [5]	NP-h [5]	FPT Theorem 4.6

of vertices $S \subseteq V$, we define $N_G(S) := (\bigcup_{v \in S} N_G(v)) \setminus S$ and $N_G^{\cap}(S) := (\bigcap_{v \in S} N_G(v))$. The *closed neighborhood* of a vertex $v \in V$ is defined as $N_G[v] := N_G(v) \cup \{v\}$ and for a set of vertices $S \subseteq V$, we define $N_G[S] := \bigcup_{v \in S} N_G[v]$. When the graph G is clear from the context we may omit the subscript. The *degree* of a vertex v is $|N(v)|$. For a vertex set $S \subseteq V$ we write $E(S) := \{\{u, v\} \in E \mid u, v \in S\}$ and denote with $G[S] := (S, E(S))$ the *subgraph of G induced by S* . Moreover, we define $G - S := G[V \setminus S]$.

Graph Classes and Graph Parameters. A vertex set $K \subseteq V$ that induces a complete graph is called a *clique*. A vertex set $I \subseteq V$ that induces a graph with no edges is called an *independent set*. A graph $G = (V, E)$ is a *cluster graph* if each connected component of G consists of a clique. A graph is a *cograph* if it does not contain the P_4 , the path on four vertices, as an induced subgraph. A graph is an *interval graph* if it is the intersection graph of some set of intervals on the real line. A graph is a *proper interval graph* if additionally, none of the intervals is properly contained in another one of the intervals. A graph $G = (V, E)$ is *unipolar* if V can be partitioned into C and P such that C is a clique and $G[P]$ is a cluster graph [11]. We refer to C as the *core* and to P as the *periphery* of G . Furthermore, G is *complete unipolar* if each vertex of C is adjacent to each vertex of P . A unipolar graph $G = (V, E)$ is a *split graph* if P is an independent set. Moreover, G is a *dense split graph* if each vertex of C is adjacent to each vertex of P . A graph G is a *split cluster graph* if every connected component is a split graph [6]. We note the following relations between these graph classes. A complete unipolar graph is a cograph and an interval graph but not necessarily a proper interval graph. Also, every cluster graph is a split cluster graph.

A vertex set $S \subseteq V$ is a *vertex cover* for a graph G if the graph $G - S$ does not contain any edges. The *vertex cover number* $\text{vc}(G)$ is the size of a smallest vertex cover for G . We say that a vertex set $M \subseteq V$ is a *cluster modulator for G* if $G - M$ is a cluster graph. The *cluster vertex deletion number* $\text{cvd}(G)$ of G is defined as the size of the smallest cluster modulator for G . The relation \sim with $u \sim v$ if $N(u) = N(v)$ or $N[u] = N[v]$ is an equivalence relation and the *neighborhood diversity* $\text{nd}(G)$ is the number of equivalence classes of this relation. The *treewidth* of a graph G , denoted $\text{tw}(G)$, is a parameter that, informally speaking, measures how close the graph is to a tree. For a formal definition of treewidth refer to [8]. We make use of the fact that $\text{vc}(G) \geq \text{tw}(G)$ for every graph G . If the graph G is clear from context, we omit it from the parameter notation.

Parameterized Complexity. An algorithm for a parameterized problem L is an FPT-algorithm, if there is a computable function f such that for every instance (I, k) the algorithm decides in $f(k) \cdot |I|^{\mathcal{O}(1)}$ time whether (I, k) is a yes-instance of L . A *polynomial (many-one) kernel* for L is an algorithm that computes for each instance (I, k) in polynomial time an equivalent instance (I', k') with $(|I'| + k') \in k^{\mathcal{O}(1)}$. A *polynomial Turing kernel* for L is an algorithm that decides whether a given instance (I, k) is a yes-instance of L in time $(|I| + k)^{\mathcal{O}(1)}$, when given access to an oracle that decides membership in L for any instance (I', k') with $(|I'| + k') \in k^{\mathcal{O}(1)}$ in constant time.

A *polynomial parameter transformation* is a reduction from a parameterized problem A to a parameterized problem B that transforms each instance (I, k) of A in polynomial time into an equivalent instance (I', k') of B with $k' \in k^{\mathcal{O}(1)}$. Note that polynomial parameter transformations are transitive. Moreover, if there is a polynomial parameter transformation from A to B , where the unparameterized versions of A and B are in NP, then A admits a polynomial (many-one/Turing) kernel if B admits a polynomial (many-one/Turing) kernel. The Exponential Time Hypothesis (ETH) conjectures that 3-SAT cannot be solved in $2^{o(|F|)}$ time where F is the input formula. For further background on these definitions, we refer the reader to the standard monographs [8, 10].

Clusterings and Formal Problem Definition. A *clustering* \mathcal{C} of a graph $G = (V, E)$ is a partition of V into subsets C_1, \dots, C_r such that each C_i is a clique. For a clustering \mathcal{C} of $G = (V, E)$ we denote with $E(\mathcal{C})$ the set of edges with both endpoints in the same cluster of \mathcal{C} . Let $\omega : E \rightarrow \mathbb{N}$ be an edge-weight function. For a set of edges $E' \subseteq E$, we define $\omega(E') := \sum_{e \in E'} \omega(e)$ and denote with $\omega(\mathcal{C}) := \omega(E(\mathcal{C}))$ the weight of a clustering \mathcal{C} .

WEIGHTED CLUSTER DELETION (WCD)

Input: A graph $G = (V, E)$, a weight function $\omega : E \rightarrow \mathbb{N}$ and a nonnegative integer k .

Question: Is there a clustering \mathcal{C} for G such that $\omega(\mathcal{C}) \geq k$?

Some of our algorithms use brute force to find a clustering for a small subgraph of the input graph and then extend this clustering in an optimal way. This is formalized as follows. Let $G = (V, E)$ be a graph, let $S \subseteq V$ and let \mathcal{C} and \mathcal{C}_S be clusterings for G and $G[S]$, respectively. We say that \mathcal{C} *extends* the clustering \mathcal{C}_S if for each cluster C of \mathcal{C}_S , there is a cluster $C' \in \mathcal{C}$ such that $C' \cap S = C$. That is, restricting the clustering \mathcal{C} to the vertices of S results in \mathcal{C}_S . Note that \mathcal{C} may contain clusters that contain no vertex of S .

Basic Observations. A clique C in a graph $G = (V, E)$ is called a *critical clique* if all vertices of C have the same closed neighborhood. Furthermore, C is a *closed critical clique* if additionally $N[C]$ is a clique in G .

► **Observation 2.1.** Let $G = (V, E)$ be a graph, let $\omega : E \rightarrow [1, t]$ be an edge-weight function, and let C be a closed critical clique in G . Then, each optimal clustering for G and ω extends $\{C\}$.

This can be seen as follows. Let \mathcal{C} be a clustering for G which contains two distinct clusters A and B that both contain vertices of C . Note that $A \cup B$ is a clique in G , since $A \cup B \subseteq N[C]$ and C is a closed critical clique. Hence, replacing the clusters A and B by $A \cup B$ yields a better clustering.

Note that if we try to find a clustering \mathcal{C} that extends a clustering \mathcal{C}_S of some subgraph $G[S]$, then no edges between vertices of distinct clusters of \mathcal{C}_S are contained in \mathcal{C} , due to the definition of extending clusterings.

► **Observation 2.2.** *Let $G = (V, E)$ be a graph, let $S \subseteq V$, let \mathcal{C}_S be a clustering of $G[S]$, and let G' be the graph obtained from G by removing all edges between vertices of distinct clusters of \mathcal{C}_S . Then G and G' share the same clusterings that extend \mathcal{C}_S .*

Hence, in the following, whenever – for a WCD instance and a clustering \mathcal{C}_S for some subgraph $G[S]$ – we search for a clustering that extends \mathcal{C}_S , we may implicitly assume that $G[S]$ is a cluster graph. In the following, we show that we can further merge each cluster of \mathcal{C}_S to a single vertex by increasing the largest assigned edge weight.

Let $G = (V, E)$ be a graph and let C be a clique in G . Then, *merging* C in G results in the graph $G' = (V', E')$, where C is replaced by a single vertex $v_C \in C$ with neighborhood $N_{G'}(v_C) := N_G^\cap(C)$. In other words, the vertex v_C keeps only the common neighborhood of C as neighbors.

► **Definition 2.3.** *Let $G = (V, E)$ be a graph, let $\omega : E \rightarrow [1, t]$ be an edge-weight function, and let C be a clique in G . The weighted merge of C in G yields the graph $G' = (V', E')$ obtained from merging C in G with edge-weight function ω' defined as follows: For each edge $e \in E'$ which is not incident with v_C , set $\omega'(e) := \omega(e)$, and for each edge $e = \{v_C, w\} \in E'$, set $\omega'(\{v_C, w\}) := \sum_{v \in C} \omega(\{v, w\})$.*

Note that the largest edge weight assigned by ω' is at most $|C| \cdot t$ and that each edge not incident with v_C is assigned a weight of at most t by ω' .

Similarly, for a vertex set $S \subseteq V$ where $G[S]$ is a cluster graph with collection of connected components \mathcal{C}_S , we define the *clustering-merge* of \mathcal{C}_S for G and ω as the consecutive merges of all clusters of \mathcal{C}_S in G and ω . Note that the largest edge weight in the resulting instance is at most $t \cdot \max_{C \in \mathcal{C}_S} |C|$, since G contains no edge with endpoints in distinct clusters of \mathcal{C}_S .

► **Lemma 2.4 (*)**. *Let $G = (V, E)$ be a graph, let $\omega : E \rightarrow [1, t]$ be an edge-weight function, and let $S \subseteq V$ such that $G[S]$ is a cluster graph with collection of connected components \mathcal{C}_S . Moreover, let $(G' = (V', E'), \omega')$ be the graph and edge-weight function obtained by the clustering-merge of \mathcal{C}_S for G and ω . There is a bijection π between the clusterings of G' and the clusterings of G that extend \mathcal{C}_S , such that for every clustering \mathcal{C}' of G'*

- $\omega'(\mathcal{C}') = \omega(\pi(\mathcal{C}')) - \omega(\mathcal{C}_S)$ and
- $\pi(\mathcal{C}')$ extends \mathcal{C}' .

The above lemma implies in particular, that we can obtain the optimal clustering of a graph G that extends some clustering \mathcal{C}_S for some $G[S]$ by performing the clustering-merge and then computing the optimal clustering in the remaining instance.

3 Hardness Results

In this section, we present a reduction from UNIFORM EXACT COVER to WCD on dense split graphs which implies several hardness results for WCD.

UNIFORM EXACT COVER

Input: A set X and a collection \mathcal{F} of size- d subsets of X .

Question: Is there a subset $\mathcal{F}' \subseteq \mathcal{F}$ such that every element of X occurs in exactly one member of \mathcal{F}' ?

Note that we can assume that $|X|$ is divisible by d , as otherwise the instance at hand is a trivial no-instance. Our reduction is a generalization of a known reduction from X3C [5], that is, UNIFORM EXACT COVER with $d = 3$.

Construction. Let $I = (X, \mathcal{F})$ be an instance of UNIFORM EXACT COVER, where d is the size of each set of \mathcal{F} and $n := |X|$ is divisible by d . We construct from I an instance $I' = (G, \omega, k)$ of WCD as follows. Starting from an empty graph, we add to G a vertex v_x for each element $x \in X$ and make the set $V_X := \{v_x \mid x \in X\}$ a clique with $\omega(e) = 1$ for each edge e between two vertices in V_X . For each set $F \in \mathcal{F}$, we add a vertex v_F together with an edge $\{v_F, v_x\}$ with $\omega(\{v_F, v_x\}) = 2n$ for each $x \in F$ and an edge $\{v_F, v_x\}$ with $\omega(\{v_F, v_x\}) = n$ for each $x \in X \setminus F$. We define $V_{\mathcal{F}} := \{v_F \mid F \in \mathcal{F}\}$. Note that $V_{\mathcal{F}}$ is an independent set. We finish the construction by setting $k := n \cdot (2n + \frac{1}{d} \cdot \binom{d}{2})$.

Observe that the graph G in the construction above is a dense split graph with core V_X and periphery $V_{\mathcal{F}}$. Moreover, $t := 2n$ is the highest weight present in G and the core V_X consists of n vertices. Since the core of a split graph is a vertex cover, $t + \text{vc} \in \mathcal{O}(n)$, where vc denotes the vertex cover number of G . Moreover, G contains exactly $|X| + |\mathcal{F}|$ vertices.

Next, we show the equivalence of the two instances I and I' .

► **Lemma 3.1.** *I is a yes-instance of UNIFORM EXACT COVER if and only if I' is a yes-instance of WCD.*

Proof. (\Rightarrow) Let I be a yes-instance and let \mathcal{F}' be a solution for I . For each $F \in \mathcal{F}'$ let $C_F = \{v_x \in V_X \mid x \in F\} \cup \{v_F\}$. Consider the clustering $\mathcal{C} := \{C_F \mid F \in \mathcal{F}'\} \cup \{\{v_F\} \mid F \notin \mathcal{F}'\}$. Since \mathcal{F}' is a solution for I , each $x \in X$ is covered by exactly one set $F \in \mathcal{F}'$ and thus each vertex of V_X is in exactly one set of \mathcal{C} . Note that V_X , and thus also each subset of V_X , is a clique and each vertex v_F is adjacent to each vertex in V_X . Therefore, each $C_F \in \mathcal{C}$ is a clique and \mathcal{C} is a valid clustering. Moreover, we have $\omega(E(\mathcal{C})) = n \cdot (2n + \frac{1}{d} \cdot \binom{d}{2})$. Hence, I' is a yes-instance.

(\Leftarrow) Let I' be a yes-instance and let \mathcal{C} be an optimal solution for I' . Let $E_{\mathcal{C}}(X)$ denote the edges that are preserved in \mathcal{C} between vertices of V_X and let $E_{\mathcal{C}}(\mathcal{F})$ denote the edges that are preserved in \mathcal{C} between a vertex of V_X and a vertex of $V_{\mathcal{F}}$. Clearly, $\omega(E(\mathcal{C})) = \omega(E_{\mathcal{C}}(X)) + \omega(E_{\mathcal{C}}(\mathcal{F}))$.

We now show that each $v_x \in V_X$ is part of a cluster containing a vertex $v_F \in V_{\mathcal{F}}$ such that $x \in F$. Let $x \in X$ and let $C_x \in \mathcal{C}$ be the cluster containing v_x . Note that C_x can contain at most one vertex of $V_{\mathcal{F}}$, since $V_{\mathcal{F}}$ is an independent set. This implies that there is at most one edge of $E_{\mathcal{C}}(\mathcal{F})$ incident with v_x . Suppose that C_x does not contain a vertex v_F such that $x \in F$. If an edge $\{v_x, v_{F'}\} \in E_{\mathcal{C}}(\mathcal{F})$ exists, it has weight at most n , since $x \notin F'$. Let $F \in \mathcal{F}$ be an arbitrary set such that $x \in F$ and let C_F denote the cluster of \mathcal{C} that contains v_F . Moreover, let \mathcal{C}' be the clustering obtained by moving vertex v_x in the cluster C_F . Note that \mathcal{C}' is a valid clustering, since each vertex of V_X is adjacent to every other vertex of G , and that the weight of the edge $\{v_x, v_F\}$ is $2n$. Since $x \in F$ and v_x has at most $n - 1$ neighbors in $C_x \cap V_X$, we have

$$\omega(\mathcal{C}') - \omega(\mathcal{C}) \geq \omega(\{v_x, v_F\}) - \sum_{w \in C_x \setminus \{v_x\}} \omega(\{v_x, w\}) \geq 2n - (n + |C_x \cap V_X|) > 0,$$

which is a contradiction to \mathcal{C} being an optimal solution.

Thus, we can now assume that $\omega(E_{\mathcal{C}}(\mathcal{F})) = n \cdot 2n$. Furthermore, since each v_x is part of a cluster containing a vertex $v_F \in V_{\mathcal{F}}$ with $x \in F$, and each $F \in \mathcal{F}$ contains exactly d elements, at most d vertices from V_X can be in the same cluster in \mathcal{C} . More precisely, for each $x \in X$ let $C_x \in \mathcal{C}$ be the cluster containing v_x . We have $|(C_x \cap V_X) \setminus \{v_x\}| \leq d - 1$ for each $x \in X$.

Recall that edges between vertices of V_X have weight 1 and each edge in $E_{\mathcal{C}}(X)$ has two endpoints in V_X . If for some $x \in X$ we have $|(C_x \cap V_X) \setminus \{v_x\}| < d - 1$, then $\omega(E_{\mathcal{C}}(X)) = \sum_{x \in X} \frac{|(C_x \cap V_X) \setminus \{v_x\}|}{2} < \frac{n \cdot (d-1)}{2}$ and hence $\omega(E(\mathcal{C})) = \omega(E_{\mathcal{C}}(X)) + \omega(E_{\mathcal{C}}(\mathcal{F})) < \frac{n \cdot (d-1)}{2} + n \cdot 2n = k$, a contradiction to \mathcal{C} being a solution.

Therefore, for each $x \in X$ we have $|(C_x \cap V_X) \setminus \{v_x\}| = d - 1$. Hence, each cluster C of \mathcal{C} that contains at least one vertex of V_X fulfills $C = \{v_F\} \cup \{v_x \mid x \in F\}$ for some set $F \in \mathcal{F}$. Thus, the set $\mathcal{F}' := \{F \in \mathcal{F} \mid C_F \cap V_X \neq \emptyset, C_F \in \mathcal{C}, v_F \in C_F\}$ contains all elements in X exactly once and is a solution for I' . \blacktriangleleft

Lemma 3.1 and the bound on t and the vertex cover number of G directly give the following.

► **Proposition 3.2.** *There is a polynomial parameter transformation from UNIFORM EXACT COVER with parameter n (size of the universe) to WCD on dense split graphs with parameter $t + \text{vc}$, where vc denotes the vertex cover number of the input graph.*

Due to Lemma 3.1 and the fact that UNIFORM EXACT COVER cannot be solved in $2^{o(|X|+|\mathcal{F}|)} \cdot |I|^{\mathcal{O}(1)}$ time, unless the ETH fails [20]², we also derive the following.

► **Corollary 3.3.** *Even on dense split graphs, WCD cannot be solved in $2^{o(n+t)} \cdot n^{\mathcal{O}(1)}$ time, unless the ETH fails.*

We can adapt the construction above in order to show that WCD is hard on a restricted subclass of cographs even if all edges have weight 1 or 2.

► **Proposition 3.4.** *On complete unipolar graphs WCD remains NP-hard even if $t = 2$ and cannot be solved in $2^{o(\text{cvd}+t)} \cdot n^{\mathcal{O}(1)}$ time, unless the ETH fails.*

Proof. To show the statement, we adapt the construction above so that we obtain from $I' = (G = (V, E), \omega, k)$ an equivalent instance $I'' = (G' = (V', E'), \omega', k')$ of WCD where G' is a complete unipolar graph and all edges have weight 1 or 2. We obtain I'' from I' as follows. Each vertex $v_F \in V_{\mathcal{F}}$ is replaced by a clique K_F of size n with $\omega'(\{u, v\}) = 1$ for each $u, v \in K_F$. We add edges such that K_F is fully connected to V_X and set $\omega'(\{u, v_x\}) = 2$ for each $u \in K_F$ and $v_x \in V_X$ with $\omega(\{v_F, v_x\}) = 2n$ as well as $\omega'(\{u, v_y\}) = 1$ for each $u \in K_F$ and $v_y \in V_X$ with $\omega(\{v_F, v_y\}) = n$. Moreover, we set $k' = k + |\mathcal{F}| \cdot \binom{n}{2}$. Note that in G' the sets K_F are disjoint cliques, each fully connected to the clique V_X . Therefore, G' is a complete unipolar graph. Furthermore, each edge in G' has weight either 1 or 2. Moreover, since G' is unipolar, V_X is a cluster modulator of size n . Hence, $\text{cvd}(G') \leq n$.

It remains to show that I' and I'' are equivalent instances. Observe that in G' for each $F \in \mathcal{F}$ the clique K_F is a closed critical clique and thus in every optimal clustering for G' each vertex of K_F is part of the same cluster due to Observation 2.1. Since the edges within these critical cliques all have weight 1, they have a total weight of $|\mathcal{F}| \cdot \binom{n}{2}$. Let $S := \bigcup_{F \in \mathcal{F}} K_F$ and let $\mathcal{C}_S := \bigcup_{F \in \mathcal{F}} \{K_F\}$. Per construction \mathcal{C}_S is a clustering for $G'[S]$. Observe that (G, ω) can be obtained by the clustering-merge of \mathcal{C}_S for G' and ω' . Clearly, $G'[S]$ is a cluster graph and each cluster K_F of \mathcal{C}_S is merged into the vertex v_F in G with $N_G(v_F) = V_X = N_{G'}^{\cap}(K_F)$. Moreover, we have $\omega(\{v_F, v_x\}) = \sum_{v \in K_F} \omega'(\{v_F, v\})$.

By the above facts, Lemma 2.4 and Observation 2.1 imply that I' and I'' are equivalent instances. \blacktriangleleft

In addition, we obtain the following hardness results on graph classes that are unrelated to complete unipolar graphs.

► **Theorem 3.5 (*)**. *WCD is NP-hard on proper interval graphs.*

² Note that Theorem 3.1 in [20] shows an ETH lower bound for 3D-MATCHING, which is a special case of UNIFORM EXACT COVER.

► **Theorem 3.6 (*)**. *WCD is NP-hard on complete tripartite graphs even for $t = 2$.*

Note that complete tripartite graphs are cographs with neighborhood diversity 3. Hence, we obtain NP-hardness also for this class of graphs.

4 Split Graphs with Bounded Weights

We now show that WCD can be solved in FPT-time on split graphs when parameterized by the largest edge weight t . The algorithm is based on several properties concerning the interaction of the optimal clustering with the core of the split graphs. These properties will also be helpful for our algorithms on generalizations of split graphs.

As an auxiliary result, we first present an algorithm for WCD on split graphs, when parameterized by the size of the core of the split graph. The algorithm uses dynamic programming over subsets of the core.

► **Lemma 4.1**. *Let $G = (V, E)$ be a split graph with core C and let ω be an edge-weight function. One can find an optimal clustering for G and ω in $3^{|C|} \cdot n^{\mathcal{O}(1)}$ time.*

Proof. Let $P := \{p_1, \dots, p_{|P|}\}$ be the periphery of G . We describe a dynamic program that finds an optimal clustering for G and ω in $3^{|C|} \cdot n^{\mathcal{O}(1)}$ time.

The dynamic programming table T has entries of type $T[i, S]$ for each $S \subseteq C$ and each $i \in [0, |P|]$ and stores the weight of an optimal clustering for $G[S \cup \{p_j \mid 1 \leq j \leq i\}]$. Hence, the base case for $i = 0$ and each $S \subseteq C$ is defined as $T[i, S] := \omega(E(S))$. This is correct, since C is a clique in G . For each other entry of the dynamic programming table, that is, for each $i \in [1, |P|]$ and each $S \subseteq C$, the recurrence to compute the entry $T[i, S]$ is

$$T[i, S] := \max_{\substack{S' \subseteq S \\ S' \subseteq N(p_i)}} \omega(E(S' \cup \{p_i\})) + T[i-1, S \setminus S'].$$

Intuitively, we search for the best way to assign a subset S' of S to the cluster with vertex p_i and combine this with an optimal clustering for $G[(S \setminus S') \cup \{p_j \mid 1 \leq j < i\}]$ and ω . This is correct, since P is an independent set in G , and thus, in each valid clustering for G , the cluster containing p_i contains no other vertex of P . The total weight of an optimal clustering for G and ω is stored in $T[|P|, C]$ and a corresponding clustering can be found via traceback.

It thus remains to show the running time. Since each entry $T[i, S]$ can be computed in $2^{|S|} \cdot n^{\mathcal{O}(1)}$ time and there are $\binom{|C|}{\ell}$ subsets S of C of size exactly ℓ , all entries of the dynamic programming table T can be computed in $\sum_{\ell=0}^{|C|} \binom{|C|}{\ell} \cdot 2^\ell \cdot n^{\mathcal{O}(1)} = 3^{|C|} \cdot n^{\mathcal{O}(1)}$ time. ◀

Next, we show the first property for optimal clusterings of split graphs. Roughly speaking, the lemma shows that for each optimal clustering \mathcal{C} and each subset $\mathcal{C}' \subseteq \mathcal{C}$, at least one of \mathcal{C}' and $\mathcal{C} \setminus \mathcal{C}'$ contains at most $2t$ vertices of the core. This for example implies that the vertices of the core are only contained in $\mathcal{O}(t)$ clusters in any optimal clustering.

► **Lemma 4.2**. *Let $G = (V, E)$ be a split graph and let $\omega : E \rightarrow [1, t]$ be an edge-weight function. Moreover, let \mathcal{C} be an optimal clustering for G and ω . Then, for each subset $\mathcal{C}' \subseteq \mathcal{C}$, \mathcal{C}' contains at most $2t$ vertices of the core of G or $\mathcal{C} \setminus \mathcal{C}'$ contains at most $2t$ vertices of the core of G .*

Proof. Let C and P denote the core and the periphery of G , respectively. We show the contrapositive, that is, we show that \mathcal{C} is not an optimal clustering for G and ω , if there is a subset $\mathcal{C}' \subseteq \mathcal{C}$, such that \mathcal{C}' and $\mathcal{C} \setminus \mathcal{C}'$ contain at least $2t + 1$ vertices of the core C

each. To this end, we show that keeping the whole core as a cluster yields a better clustering than \mathcal{C} . That is, we show that $\mathcal{C}^* := \{C\} \cup \{\{p\} \mid p \in P\}$ is a better clustering than \mathcal{C} . Note that \mathcal{C}^* is a valid clustering for G . It thus remains to show that $\omega(\mathcal{C}^*) - \omega(\mathcal{C}) > 0$. To this end, we first analyze the edges of $E(\mathcal{C}) \setminus E(\mathcal{C}^*)$. Since G is a split graph, each cluster of \mathcal{C} contains at most one vertex of the periphery P . Hence, $E(\mathcal{C}) \setminus E(\mathcal{C}^*)$ contains for each vertex $v \in C$ at most one edge incident with v . Since each edge has weight at most t , this implies that $\omega(E(\mathcal{C}) \setminus E(\mathcal{C}^*)) \leq |C| \cdot t$. Next, we analyze the edges of $E(\mathcal{C}^*) \setminus E(\mathcal{C})$. Let C_1 denote the vertices of C that are contained in C' and let C_2 denote the vertices of C that are contained in $C \setminus C'$ and assume without loss of generality that $|C_1| \geq |C_2|$. Hence, $|C_1| \geq \frac{|C|}{2}$. Moreover, note that all edges of $E(C_1, C_2)$ are contained in $E(\mathcal{C}^*) \setminus E(\mathcal{C})$ which implies that $E(\mathcal{C}^*) \setminus E(\mathcal{C})$ contains at least $|E(C_1, C_2)| = |C_1| \cdot |C_2| \geq \frac{|C|}{2} \cdot (2t + 1) > |C| \cdot t$ edges of weight at least one each. Consequently,

$$\omega(\mathcal{C}^*) - \omega(\mathcal{C}) = \omega(E(\mathcal{C}^*) \setminus E(\mathcal{C})) - \omega(E(\mathcal{C}) \setminus E(\mathcal{C}^*)) > |C| \cdot t - |C| \cdot t = 0.$$

This implies that \mathcal{C} is not an optimal clustering for G and ω . ◀

This has the following implications for any optimal clustering \mathcal{C} : There is at most one cluster of size more than $2t$ in \mathcal{C} and there are $\mathcal{O}(t)$ clusters of size more than one in \mathcal{C} . We also derive the following.

► **Lemma 4.3.** *Let $G = (V, E)$ be a split graph with core C and let $\omega : E \rightarrow [1, t]$ be an edge-weight function. Moreover, let \mathcal{C} be an optimal clustering for G and ω . If $|C| > 6t$, then there is a cluster $C^* \in \mathcal{C}$ that misses at most $2t$ vertices of C , that is, $|C^* \cap C| \geq |C| - 2t$.*

Proof. Let C^* be a cluster of \mathcal{C} that contains the most vertices of C . First, we show that C^* contains at least $2t + 1$ vertices of C . Assume towards a contradiction that C^* contains at most $2t$ vertices of C , which then implies that every cluster of \mathcal{C} contains at most $2t$ vertices of C . Let C' be an arbitrary subset of \mathcal{C} such that C' contains at least $2t + 1$ vertices of C and no proper subset of C' contains at least $2t + 1$ vertices of C . Note that this implies that C' contains at most $4t$ vertices of C , since each cluster of \mathcal{C} contains at most $2t$ vertices of C . Since C has size at least $6t + 1$, this implies that $C \setminus C'$ contains at least $2t + 1$ vertices of C . Due to Lemma 4.2, this contradicts the fact that \mathcal{C} is an optimal clustering for G and ω . Hence, C^* contains at least $2t + 1$ vertices of C .

Now, consider the subset of clusters $\mathcal{C}' := \{C^*\}$. Since \mathcal{C} is an optimal clustering for G and ω and \mathcal{C}' contains more than $2t$ vertices of C , Lemma 4.2 implies that $C \setminus \mathcal{C}'$ contains at most $2t$ vertices of C . Consequently, C^* contains at least $|C| - 2t$ vertices of C . ◀

Based on this lemma, we can now show that there are only linearly many options for the largest cluster of any optimal clustering, if the core has size at least $2t^2 + 4t + 1$.

► **Lemma 4.4.** *Let $G = (V, E)$ be a split graph with core C and periphery P , let $\omega : E \rightarrow [1, t]$ be an edge-weight function, and let \mathcal{C} be an optimal clustering for G and ω . If $|C| \geq 2t^2 + 4t + 1$, then either \mathcal{C} contains the cluster C or there is some periphery vertex $v \in P$ with degree at least $|C| - 2t$ such that \mathcal{C} contains the cluster $N[v]$.*

Proof. Assume towards a contradiction that the statement does not hold. Then, \mathcal{C} contains neither the cluster C nor the cluster $N[v]$ for any periphery vertex $v \in P$ with degree at least $|C| - 2t$. Let C^* be a cluster of \mathcal{C} with the most vertices of C . Since C has size at least $6t + 1$ and \mathcal{C} is an optimal clustering for G and ω , Lemma 4.3 implies that C^* contains at least $|C| - 2t \geq 2t^2 + 1$ vertices of C .

32:10 When Can Cluster Deletion with Bounded Weights Be Solved Efficiently?

Let v^* be the unique periphery vertex of C^* if such a vertex exists and an arbitrary vertex of C^* , otherwise. In both cases, there is a vertex $v \in (N(v^*) \cap C) \setminus C^*$, since C is not a cluster in \mathcal{C} and for no periphery vertex $w \in P$ with degree at least $|C| - 2t$, $N[w]$ is a cluster of \mathcal{C} . We show that we can obtain a better clustering by moving vertex v to C^* . Note that $C^* \cup \{v\}$ is a clique in G , since v is adjacent to v^* and all vertices of $(C^* \setminus \{v^*\}) \cup \{v\}$ are from C . Let \mathcal{C}' be the clustering for G obtained from \mathcal{C} by moving vertex v to C^* . We show that \mathcal{C}' is a better clustering for G and ω than \mathcal{C} . To this end, we analyze the total weight incident with vertex v under both \mathcal{C} and \mathcal{C}' . Let C_v be the cluster of \mathcal{C} containing v . Since $\mathcal{C} \setminus \{C^*\}$ contains at most $2t$ vertices of C , C_v has size at most $2t + 1$. Hence, v is incident with at most $2t$ edges in $E(C)$, each of weight at most t . Moreover, v is incident with $|C^*| \geq 2t^2 + 1$ edges of weight at least one each in $E(C')$. Hence, \mathcal{C}' is a better clustering for G and ω than \mathcal{C} . This contradicts the fact that \mathcal{C} is an optimal clustering for G and ω . ◀

With this property at hand, we are now able to show that WCD can be solved in $2^{\mathcal{O}(t)} \cdot n^{\mathcal{O}(1)}$ time on split graphs with a small or a very large core.

► **Lemma 4.5.** *Let $G = (V, E)$ be a split graph with core C and let $\omega : E \rightarrow [1, t]$ be an edge-weight function. One can find an optimal clustering for G and ω in $2^{\mathcal{O}(t)} \cdot n^{\mathcal{O}(1)}$ time if $|C| \leq 6t$ or $|C| \geq 2t^2 + 4t + 1$.*

Proof. If $|C| \leq 6t$, then we can find an optimal clustering for G and ω in $3^{|C|} \cdot n^{\mathcal{O}(1)} = 2^{\mathcal{O}(t)} \cdot n^{\mathcal{O}(1)}$ time due to Lemma 4.1.

Otherwise, if $|C| \geq 2t^2 + 4t + 1$, Lemma 4.4 implies that each optimal clustering for G and ω contains a cluster C^* such that $C^* = C$ or $C^* = N[v]$ for some periphery vertex $v \in P$ with degree at least $|C| - 2t$. Since these are at most $|P| + 1 \in \mathcal{O}(n)$ possibilities, we can perform an initial branching for the choice of C^* . For each such choice for C^* , we find an optimal clustering \mathcal{C}^* for $G - C^*$ and ω and return the best clustering $\mathcal{C}^* \cup \{C^*\}$ over all choices of C^* . Note that this algorithm is correct due to Lemma 4.4. The initial branching can be done in $n^{\mathcal{O}(1)}$ time and for each branching-instance we can find an optimal solution for $G - C^*$ and ω in $2^{\mathcal{O}(t)} \cdot n^{\mathcal{O}(1)}$ time, since $G - C^*$ is a split graph with a core $C \setminus C^*$ of size at most $2t$.

Hence, in both cases, we find an optimal solution for I in the desired running time. ◀

Thus, to obtain an algorithm for WCD on split graphs, we now show how to solve the case that the core has size at least $6t + 1$ and at most $2t^2 + 4t$ and use this to bound the total running time.

► **Theorem 4.6.** *WCD can be solved in $t^{\mathcal{O}(t)} \cdot n^{\mathcal{O}(1)}$ time on split graphs, where t denotes the largest edge weight.*

Proof. Let $I = (G = (V, E), \omega : E \rightarrow [1, t], k)$ be an instance of WCD, where G is a split graph with core C and periphery P . We show how to obtain an optimal clustering for G and ω in time $t^{\mathcal{O}(t)} \cdot n^{\mathcal{O}(1)}$.

By Lemma 4.5, we can achieve this running time if $|C| \leq 6t$ or $|C| \geq 2t^2 + 4t + 1$. Hence, in the following, we assume that $6t + 1 \leq |C| \leq 2t^2 + 4t$. Due to Lemma 4.3, this implies that each optimal clustering for G and ω contains a cluster C^* such that $|C^* \cap C| \geq |C| - 2t$. Based on this fact, we can find an optimal clustering for G and ω by iterating over all $\hat{C} \subseteq C$ of size at least $|C| - 2t$ and finding for each cluster $C^* \in \{\hat{C}\} \cup \{\hat{C} \cup \{v\} \mid v \in P, \hat{C} \subseteq N(v)\}$ an optimal clustering for $G - C^*$ and ω . Over all such choices of C^* , we return the best clustering $\{C^*\} \cup \mathcal{C}^*$, where \mathcal{C}^* is an optimal clustering for $G - C^*$. Due to Lemma 4.3, this algorithm finds an optimal clustering for G and ω .

It remains to show the running time. Since there are at most $|C|^{2t} = (2t^2 + 4t + 1)^{2t} \in t^{\mathcal{O}(t)}$ distinct subsets \widehat{C} of C with $|\widehat{C}| \geq |C| - 2t$ and for each such subset we consider at most $|P| + 1 \in \mathcal{O}(n)$ possible clusters C^* , we have to find an optimal clustering for at most $t^{\mathcal{O}(t)} \cdot n^{\mathcal{O}(1)}$ subgraphs $G - C^*$ of G . For each choice for C^* , $G - C^*$ is a split graph with a core of size $|C \setminus C^*| \leq 2t$. We can thus find an optimal clustering for $G - C^*$ in $2^{\mathcal{O}(t)} \cdot n^{\mathcal{O}(1)}$ time due to Lemma 4.5. Hence, the total running time is $t^{\mathcal{O}(t)} \cdot 2^{\mathcal{O}(t)} \cdot n^{\mathcal{O}(1)} = t^{\mathcal{O}(t)} \cdot n^{\mathcal{O}(1)}$. ◀

On dense split graphs, we obtain an even faster algorithm.

► **Theorem 4.7.** *WCD can be solved in $2^{\mathcal{O}(t)} \cdot n^{\mathcal{O}(1)}$ time on dense split graphs, where t denotes the largest edge weight.*

Proof. Let $I = (G = (V, E), \omega : E \rightarrow [1, t], k)$ be an instance of WCD, where G is a dense split graph with core C and periphery P . We show how to obtain an optimal clustering for G and ω in time $2^{\mathcal{O}(t)} \cdot n^{\mathcal{O}(1)}$.

Due to Lemma 4.5, we can achieve this running time if $|C| \leq 6t$. Hence, assume in the following, that $|C| > 6t$. Let \mathcal{C} be an optimal clustering for G and ω . Due to Lemma 4.3, \mathcal{C} contains a cluster C^* such that $|C^* \cap C| \geq |C| - 2t > 4t$. We show that $C^* := N[p] = \{p\} \cup C$ for some periphery vertex p . To this end, we first show that C^* contains all core vertices. Let $C' := C \setminus C^*$ denote the core vertices that are not in C^* . Since G is a dense split graph, each vertex of C' is adjacent to each vertex of C^* . Hence, moving all vertices of C' to the cluster C^* yields a valid clustering \mathcal{C}' . If $C' \neq \emptyset$, this clustering improves over \mathcal{C} , since each vertex of C' was adjacent to at most one periphery vertex, which implies that

$$\omega(E(\mathcal{C}) \setminus E(\mathcal{C}')) \leq |C'| \cdot t < |C'| \cdot 4t \leq |C'| \cdot |C^*| \leq \omega(E(\mathcal{C}') \setminus E(\mathcal{C})).$$

Since \mathcal{C} is an optimal clustering for G and ω , this implies that $C' = \emptyset$ and thus C^* contains all core vertices. Moreover, due to the optimality of \mathcal{C} , C^* contains one periphery vertex, as otherwise adding an arbitrary periphery vertex to C^* would yield a better valid clustering, since G is a dense split graph. Hence, $C^* = N[p] = C \cup \{p\}$ for some periphery vertex $p \in P$. This implies that C^* is the only cluster of \mathcal{C} that contains any edges, which further implies that $\omega(\mathcal{C}) = \omega(E(C^*)) = \omega(E(N[p]))$.

Hence, to find an optimal clustering for G and ω it suffices to find a periphery vertex $p \in P$ that maximizes $\omega(E(N[p]))$. This can be done in polynomial time. ◀

The result also gives a dichotomy with respect to the neighborhood diversity: Theorem 3.6 showed NP-hardness for neighborhood diversity 3 and $t = 2$. We now show an FPT-algorithm with respect to t for neighborhood diversity at most 2. Note that the graphs with neighborhood diversity at most 2 are a subset of the cographs but not a subset of the split graphs, since complete bipartite graphs have neighborhood diversity 2.

► **Corollary 4.8.** *On graphs with neighborhood diversity at most 2, WCD can be solved in $2^{\mathcal{O}(t)} \cdot n^{\mathcal{O}(1)}$ time.*

Proof. Note that if a graph is already a cluster graph, an optimal clustering of that graph simply contains all edges. The only graphs of neighborhood diversity at most 2 that are not cluster graphs are complete bipartite graphs and dense split graphs. On bipartite graphs, WCD asks simply for a maximum weight matching which can be solved in polynomial time [24]. For dense split graphs, Theorem 4.7 implies that we can solve WCD in the stated running time. ◀

5 Further FPT-Algorithms

Bounded-Treewidth Modulators to Clique. In the following, we extend ideas of the FPT-algorithm for split graphs to a generalization of split graphs. Namely, we show that WCD can be solved in FPT-time when parameterized by $t + r$ on graphs when a treewidth- r clique modulator is provided. Here, a *clique modulator* in a graph G is a vertex set M such that $G - M$ is a complete graph. The treewidth of a clique modulator M is defined to be the treewidth of $G[M]$. We now consider the parameter treewidth r of a given clique modulator of G and show that WCD admits an FPT-algorithm for this parameter.

► **Lemma 5.1 (*)**. *Let G be a graph with edge-weight function $\omega : E \rightarrow [1, t]$ and let M be a treewidth- r modulator in G to a clique C . Then, if $|C| \geq 2 \cdot (t \cdot (r + 1))^2 + 4t \cdot (r + 1) + 1$, any optimal clustering \mathcal{C} for G and ω contains the cluster C or there is some clique $K \subseteq M$ such that \mathcal{C} contains the cluster $K \cup (N^\cap(K) \cap C)$ and this cluster contains at least $|C| - 2(t \cdot (r + 1))$ vertices of C .*

Note that this statement is a direct generalization of Lemma 4.4, since the periphery of a split graph G is a treewidth-0 modulator to the core of G .

With this lemma at hand, we are now able to present an algorithm for WCD when parameterized by t and the treewidth of a given clique modulator.

► **Theorem 5.2.** *WCD can be solved in time $2^{\mathcal{O}(t^2 \cdot r^2)} \cdot n^{\mathcal{O}(1)}$ when given a treewidth- r clique modulator M for the input graph G .*

Proof. First, assume that $C := V \setminus M$ has size at most $2(t \cdot (r + 1))^2 + 4t \cdot (r + 1)$ then the input graph has treewidth $\mathcal{O}(t^2 \cdot r^2)$ and we can solve WCD on this graph in time $2^{\mathcal{O}(t^2 \cdot r^2)} \cdot n^{\mathcal{O}(1)}$ [25].³

Otherwise, C has size at least $2(t \cdot (r + 1))^2 + 4t \cdot (r + 1) + 1$. Then, by Lemma 5.1, for each optimal clustering \mathcal{C} , either (a) \mathcal{C} contains C or (b) there is a clique K in $G[M]$ such that \mathcal{C} has a cluster consisting of K plus all of the (at least $|C| - 2(t \cdot (r + 1))$ many) common neighbors of K in C . Now observe that in Case (a) we may simply remove C from G and compute an optimal clustering for $G - C = G[M]$ which has treewidth at most r . This can be done in $2^{\mathcal{O}(r)} \cdot n^{\mathcal{O}(1)}$ time [25]. Otherwise, in Case (b), since $G[M]$ has treewidth at most r , we can enumerate all cliques K of $G[M]$ in $2^{\mathcal{O}(r)} \cdot n^{\mathcal{O}(1)}$ time, for example using the fact that $G[M]$ has degeneracy at most r . Now, for each clique K , we consider the case that the optimal clustering contains a cluster C' consisting of K and of all common neighbors of K in C . To compute the optimal clustering in that case, we may remove C' and find an optimal clustering for the remaining graph $G - C' = G[M \cup C \setminus C']$. This graph has treewidth at most $2(t \cdot (r + 1)) + r$ since $|C \setminus C'| \leq 2(t \cdot (r + 1))$ and $G[M]$ has treewidth r . Thus, an optimal clustering for this graph can be computed in time $2^{\mathcal{O}(t \cdot r)} \cdot n^{\mathcal{O}(1)}$. The total running time for Case (b) is thus $2^r \cdot n^{\mathcal{O}(1)} \cdot 2^{\mathcal{O}(t \cdot r)} \cdot n^{\mathcal{O}(1)}$. ◀

Parameterization by (Split) Cluster Vertex Deletion Number. Next, we focus on the cluster vertex deletion number cvd of the input graph, that is, the minimum number of vertices to remove from G to obtain a cluster graph. Recently, it was shown that CLUSTER DELETION admits an FPT-algorithm when parameterized by cvd [14]. In contrast, WCD

³ The algorithm described in [25] solves the unweighted problem via dynamic programming on tree decompositions; it can be easily adapted to the weighted case by summing up over edge weights instead of counting edges inside clusters.

is known to be NP-hard even on graphs with $\text{cvd} = 2$ [19]. This motivates the study of the combined parameter cvd and the maximum edge weight t . We show an FPT-algorithm for WCD parameterized by this combined parameter.

To this end, we first provide some additional notation. Let $G = (V, E)$ be a graph. For a cluster modulator M , let $\mathcal{B}(M)$ be the collection of connected components of $G - M$, that is, the clusters of the cluster graph $G - M$. The individual clusters of $\mathcal{B}(M)$ are referred to as *bags*. If the cluster modulator is clear from the context, we may also only write \mathcal{B} .

► **Theorem 5.3.** *WCD can be solved in $(t \cdot \text{cvd})^{\mathcal{O}(t \cdot \text{cvd})} \cdot n^{\mathcal{O}(1)}$ time.*

Proof. Let $I := (G := (V, E), \omega, k)$ be an instance of WCD with $\omega : E \rightarrow [1, t]$. The algorithm consists of two steps. First, we compute a minimum-size cluster modulator M for G . Second, we iterate over all possible clusterings of $G[M]$ and compute for each such clustering \mathcal{C}_M the best clustering of G that extends \mathcal{C}_M . To solve the latter task, we use dynamic programming over subsets to find an optimal way to distribute the vertices of the bags of $\mathcal{B}(M)$ among the clusters of \mathcal{C}_M .

Let M be a minimum-size cluster modulator for G with collection of bags \mathcal{B} and let \mathcal{C}_M be a fixed clustering of $G[M]$. We fix an arbitrary ordering of the bags of \mathcal{B} and let B_i denote the i th bag according to this fixed ordering. The dynamic programming table T has entries of type $T[i, \mathcal{C}'_M]$ with $i \in [0, |\mathcal{B}|]$ and $\mathcal{C}'_M \subseteq \mathcal{C}_M$ and stores the total edge weight of an optimal way to distribute the vertices of the first i bags among the clusters of \mathcal{C}'_M . Hence, the base case for $i = 0$ and each $\mathcal{C}'_M \subseteq \mathcal{C}_M$ is defined as $T[0, \mathcal{C}'_M] := \omega(\mathcal{C}'_M)$. The key observation for the dynamic program is the fact that no cluster in any clustering for G can contain vertices of more than one bag. Hence, to find an optimal way to distribute the vertices of the first i bags among the clusters of \mathcal{C}'_M it is sufficient to check for each subset $\mathcal{C}''_M \subseteq \mathcal{C}'_M$ for an optimal distribution of the vertices of the i th bag among the clusters of \mathcal{C}''_M and combine it with an optimal way to distribute the vertices of the first $i - 1$ bags among the clusters of $\mathcal{C}'_M \setminus \mathcal{C}''_M$. This leads to the following recurrence, where $\text{OPT}(B_i, \mathcal{C}''_M)$ denotes an optimal way to distribute the vertices of the i th bag among the clusters of \mathcal{C}''_M :

$$T[i, \mathcal{C}'_M] := \max_{\mathcal{C}''_M \subseteq \mathcal{C}'_M} \text{OPT}(B_i, \mathcal{C}''_M) + T[i - 1, \mathcal{C}'_M \setminus \mathcal{C}''_M].$$

By the above argumentation, this recurrence is correct. Hence, a total weight of a best clustering for G that extends \mathcal{C}_M is stored in $T[|\mathcal{B}|, \mathcal{C}_M]$. Moreover, a corresponding clustering can be found via trace back.

It thus remains to show the running time of the dynamic program. To this end, we first need to show that the value $\text{OPT}(B_i, \mathcal{C}''_M)$ for each $i \in [1, |\mathcal{B}|]$ and each $\mathcal{C}''_M \subseteq \mathcal{C}_M$ can be computed in the desired running time with respect to $\text{cvd} + t$.

Let $G_i := G[B_i \cup \bigcup_{C' \in \mathcal{C}''_M} C']$. Due to Observation 2.2, we can assume without loss of generality that $G_i[M]$ is a cluster graph. Hence, G_i is unipolar, since $G_i - M$ is a clique on the vertices of B_i . Let (G'_i, k') be the graph and edge-weight function obtained from performing the clustering-merge of \mathcal{C}''_M for G_i and ω . Note that G'_i is a split graph and the largest assigned weight by ω' is at most $|M| \cdot t = \text{cvd} \cdot t$. Hence, we can find an optimal clustering \mathcal{C}'_i for G'_i and ω' in time $(\text{cvd} \cdot t)^{\mathcal{O}(\text{cvd} \cdot t)} \cdot n^{\mathcal{O}(1)}$ due to Theorem 4.6. By Lemma 2.4, we can then find a best clustering for G_i and ω that extends \mathcal{C}''_M in polynomial time. Hence, for each $i \in [1, |\mathcal{B}|]$ and each $\mathcal{C}''_M \subseteq \mathcal{C}_M$, the value $\text{OPT}(B_i, \mathcal{C}''_M)$ can be computed in $(\text{cvd} \cdot t)^{\mathcal{O}(\text{cvd} \cdot t)} \cdot n^{\mathcal{O}(1)}$ time.

Concluding, the whole algorithm runs in the desired running time, since (a) M can be computed in $1.811^{\text{cvd}} \cdot n^{\mathcal{O}(1)}$ time [27], (b) all clusterings for $G[M]$ can be enumerated in $|M|^{|M|} = \text{cvd}^{\text{cvd}}$ time, (c) for each clustering \mathcal{C}_M for $G[M]$, the dynamic programming table T has $2^{|\mathcal{C}_M|} \cdot n \leq 2^{\text{cvd}} \cdot n$ entries, and (d) each such entry can be computed in $(\text{cvd} \cdot t)^{\mathcal{O}(\text{cvd} \cdot t)} \cdot n^{\mathcal{O}(1)}$ time. ◀

32:14 When Can Cluster Deletion with Bounded Weights Be Solved Efficiently?

We can extend the idea of the above algorithm to the even smaller parameter split cluster vertex deletion number scvd , that is the minimum number of vertices to remove from G such that in the remaining graph each connected component is a split graph. More formally, let $G = (V, E)$ be a graph. We say that a vertex set $M \subseteq V$ is a *split cluster modulator* for G if $G - M$ is a split cluster graph. The *split cluster vertex deletion number* scvd of G is defined as the size of the smallest split cluster modulator for G . For a split cluster modulator M , let $\mathcal{B}(M)$ be the collection of connected components of $G - M$, that is, the maximal induced split graphs of the split cluster graph $G - M$. The individual maximal induced split graphs of $\mathcal{B}(M)$ are referred to as *bags*. If the split cluster modulator is clear from the context, we may also only write \mathcal{B} .

Note that scvd is a smaller parameter than cvd , since each cluster graph is also a split cluster graph. We now show that WCD can be solved in FPT-time on general graphs when parameterized by the largest edge weight t and scvd . To this end, we first give an algorithm for computing a clustering when given a vertex set S such that $G - S$ is a split graph with core C and periphery P . In that case, $S \cup P$ is a treewidth- $|S|$ modulator to the core of $G - S$ which is a clique. Thus, Theorem 5.2 implies the following.

► **Corollary 5.4.** *Let $G = (V, E)$ be a graph and let $\omega : E \rightarrow [1, t]$ be an edge-weight function. Let $S \subseteq V$ such that $G - S$ is a split graph. One can find an optimal clustering for G and ω in $2^{\mathcal{O}((t+|S|)^2)} \cdot n^{\mathcal{O}(1)}$ time.*

We now use this algorithm as a subroutine in an algorithm similar to the one of Theorem 5.3.

► **Theorem 5.5.** *WCD can be solved in $2^{\mathcal{O}(\text{scvd}^2 \cdot t^2)} \cdot n^{\mathcal{O}(1)}$ time.*

Proof. After computing a smallest split cluster modulator M in $2^{\mathcal{O}(\text{scvd})} \cdot n^{\mathcal{O}(1)}$ time [6], we perform the same dynamic program as in the proof of Theorem 5.3 over the bags resulting from the modulator M . Note that each bag is now a split graph instead of a clique. The only difference then lies in computing the value $\text{OPT}(B_i, \mathcal{C}_M'')$ for each bag B_i and each $\mathcal{C}_M'' \subseteq \mathcal{C}_M$. Due to Corollary 5.4, this can be done in $2^{\mathcal{O}((t+|M|)^2)} \cdot n^{\mathcal{O}(1)}$ time. Hence, the whole algorithm also runs in $2^{\mathcal{O}((t+|M|)^2)} \cdot n^{\mathcal{O}(1)}$ time. ◀

6 Kernelization lower bounds

Given the FPT-algorithms for parameter t presented in Section 4, a natural next question is to ask for a polynomial kernel for t . In this section, we show that WCD on (dense) split graphs does not admit a polynomial kernel when parameterized by $t + \text{vc}$, unless $\text{NP} \subseteq \text{coNP/poly}$. Moreover, we show that even a polynomial Turing kernel for WCD on (dense) split graphs when parameterized by $t + \text{vc}$ is unlikely, by showing that the problem is WK[1]-hard. Roughly speaking, WK[1]-hardness for a parameterized problem means that a polynomial Turing kernel for this problem is unlikely [18].

To show our kernelization lower bounds, we present a chain of polynomial parameter transformations starting from SET COVER when parameterized by the size of the universe.

SET COVER

Input: A set X , a collection \mathcal{F} of subsets of X , and an integer k .

Question: Is there a subset $\mathcal{F}' \subseteq \mathcal{F}$ of size at most k , such that every element of X occurs in at least one set of \mathcal{F}' ?

Since SET COVER when parameterized by the size of the universe, is WK[1]-hard [18] and does not admit a polynomial kernel, unless $\text{NP} \subseteq \text{coNP/poly}$ [9], this then implies that WCD on (dense) split graphs when parameterized by $t + \text{vc}$ is WK[1]-hard and does not admit a polynomial kernel, unless $\text{NP} \subseteq \text{coNP/poly}$.

► **Lemma 6.1** ([18]). *There is a polynomial parameter transformation from SET COVER parameterized by the size of the universe to EXACT COVER parameterized by the size of the universe.*

This statement follows, since both SET COVER and EXACT COVER parameterized by the size of the universe are WK[1]-complete, and the class of WK[1]-complete problems is closed under polynomial parameter transformations [18].

We now present a polynomial parameter transformation from EXACT COVER to UNIFORM EXACT COVER, both parameterized by size of the universe.

► **Proposition 6.2** (*). *There is a polynomial parameter transformation from EXACT COVER parameterized by the size of the universe to UNIFORM EXACT COVER parameterized by the size of the universe.*

Based on the observed polynomial parameter transformation from EXACT COVER parameterized by size of the universe to WCD on dense split graph when parameterized by $t + vc$ in Proposition 3.2, we conclude the following due to the chain of polynomial parameter transformations (Lemma 6.1, Proposition 6.2, and Proposition 3.2) and the kernelization lower bounds for SET COVER.

► **Theorem 6.3.** *WCD on dense split graphs when parameterized by $t + vc$ does not admit a polynomial kernel, unless $\text{NP} \subseteq \text{coNP}/\text{poly}$. Moreover, WCD on dense split graphs is WK[1]-hard when parameterized by $t + vc$.*

Additionally, due to Lemma 6.1, Proposition 6.2, and the kernelization lower bounds for SET COVER, we also derive the following.

► **Corollary 6.4.** *UNIFORM EXACT COVER when parameterized by the size of the universe does not admit a polynomial kernel, unless $\text{NP} \subseteq \text{coNP}/\text{poly}$. Moreover, UNIFORM EXACT COVER is WK[1]-hard when parameterized by the size of the universe.*

7 Conclusion

The most immediate open question is whether WEIGHTED CLUSTER DELETION is polynomial-time solvable for constant values of t when the input graph is a proper interval graph. Another direction would be to improve the running time for the FPT-algorithm on split graphs to close the gap between the upper and lower bound. Finally it would be interesting to consider other NP-hard edge-weighted problems using the parameter t . A good candidate would be CLUSTER EDITING [1, 26] where the task is to obtain a cluster graph by modifying as few edges as possible. The weighted version of this problem has also received a lot of attention over the years [2, 7]. As for WEIGHTED CLUSTER DELETION, such a study would need to focus on graph classes where unweighted CLUSTER EDITING is polynomial-time solvable. Another direction could be to consider problems related to CLUSTER DELETION, where each cluster is demanded to fulfill some relaxed cluster definition, for example being a so-called s -plex [17].

References

- 1 Sebastian Böcker and Jan Baumbach. Cluster editing. In Paola Bonizzoni, Vasco Brattka, and Benedikt Löwe, editors, *Proceedings of the 9th Conference on Computability in Europe (CiE '13)*, volume 7921 of *Lecture Notes in Computer Science*, pages 33–44. Springer, 2013. doi:10.1007/978-3-642-39053-1_5.

32:16 When Can Cluster Deletion with Bounded Weights Be Solved Efficiently?

- 2 Sebastian Böcker, Sebastian Briesemeister, Quang Bao Anh Bui, and Anke Truß. Going weighted: Parameterized algorithms for cluster editing. *Theoretical Computer Science*, 410(52):5467–5480, 2009. doi:10.1016/J.TCS.2009.05.006.
- 3 Sebastian Böcker and Peter Damaschke. Even faster parameterized cluster deletion and cluster editing. *Information Processing Letters*, 111(14):717–721, 2011. doi:10.1016/J.IPL.2011.05.003.
- 4 Flavia Bonomo, Guillermo Durán, Amedeo Napoli, and Mario 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. *Information Processing Letters*, 115(6-8):600–603, 2015. doi:10.1016/J.IPL.2015.02.007.
- 5 Flavia Bonomo, Guillermo Durán, and Mario Valencia-Pabon. Complexity of the cluster deletion problem on subclasses of chordal graphs. *Theoretical Computer Science*, 600:59–69, 2015. doi:10.1016/j.tcs.2015.07.001.
- 6 Sharon Bruckner, Falk Hüffner, and Christian Komusiewicz. A graph modification approach for finding core-periphery structures in protein interaction networks. *Algorithms for Molecular Biology*, 10:16, 2015. doi:10.1186/S13015-015-0043-7.
- 7 Yixin Cao and Jianer Chen. Cluster editing: Kernelization based on edge cuts. *Algorithmica*, 64(1):152–169, 2012. doi:10.1007/S00453-011-9595-1.
- 8 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
- 9 Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Kernelization lower bounds through colors and ids. *ACM Transactions on Algorithms*, 11(2):13:1–13:20, 2014. doi:10.1145/2650261.
- 10 Rodney G. Downey and Michael Ralph Fellows. *Fundamentals of Parameterized Complexity*. Springer Science & Business Media, 2013.
- 11 Elaine M. Eschen and Xiaoqiang Wang. Algorithms for unipolar and generalized split graphs. *Discrete Applied Mathematics*, 162:195–201, 2014. doi:10.1016/J.DAM.2013.08.011.
- 12 Wen-Yu Gao and Hang Gao. $2k$ -vertex kernels for cluster deletion and strong triadic closure. *Journal of Computer Science and Technology*, 38:1431–1439, 2023. doi:10.1007/s11390-023-1420-1.
- 13 Yong Gao, Donovan R. Hare, and James Nastos. The cluster deletion problem for cographs. *Discrete Mathematics*, 313(23):2763–2771, 2013. doi:10.1016/j.disc.2013.08.017.
- 14 Jaroslav Garvardt, Nils Morawietz, André Nichterlein, and Mathias Weller. Graph clustering problems under the lens of parameterized local search. In Neeldhara Misra and Magnus Wahlström, editors, *Proceedings of the 18th International Symposium on Parameterized and Exact Computation (IPEC '23)*, volume 285 of *LIPICs*, pages 20:1–20:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.IPEC.2023.20.
- 15 Petr A. Golovach, Pinar Heggernes, Athanasios L. Konstantinidis, Paloma T. Lima, and Charis Papadopoulos. Parameterized aspects of strong subgraph closure. *Algorithmica*, 82(7):2006–2038, 2020. doi:10.1007/S00453-020-00684-9.
- 16 Niels Grüttemeier and Christian Komusiewicz. On the relation of strong triadic closure and cluster deletion. *Algorithmica*, 82(4):853–880, 2020. doi:10.1007/S00453-019-00617-1.
- 17 Jiong Guo, Christian Komusiewicz, Rolf Niedermeier, and Johannes Uhlmann. A more relaxed model for graph-based data clustering: s -plex cluster editing. *SIAM Journal on Discrete Mathematics*, 24(4):1662–1683, 2010. doi:10.1137/090767285.
- 18 Danny Hermelin, Stefan Kratsch, Karolina Soltys, Magnus Wahlström, and Xi Wu. A completeness theory for polynomial (Turing) kernelization. *Algorithmica*, 71(3):702–730, 2015. doi:10.1007/S00453-014-9910-8.
- 19 Giuseppe F. Italiano, Athanasios L. Konstantinidis, and Charis Papadopoulos. Structural parameterization of cluster deletion. In Chun-Cheng Lin, Bertrand M. T. Lin, and Giuseppe Liotta, editors, *Proceedings of the 17th International Conference and Workshops on Algorithms and Computation (WALCOM '23)*, volume 13973 of *Lecture Notes in Computer Science*, pages 371–383. Springer, 2023. doi:10.1007/978-3-031-27051-2_31.

- 20 Klaus Jansen, Felix Land, and Kati Land. Bounding the running time of algorithms for scheduling and packing problems. *SIAM J. Discret. Math.*, 30(1):343–366, 2016. doi:10.1137/140952636.
- 21 Christian Komusiewicz, Rolf Niedermeier, and Johannes Uhlmann. Deconstructing intractability – A multivariate complexity analysis of interval constrained coloring. *Journal of Discrete Algorithms*, 9(1):137–151, 2011. doi:10.1016/J.JDA.2010.07.003.
- 22 Christian Komusiewicz and Johannes Uhlmann. Cluster editing with locally bounded modifications. *Discrete Applied Mathematics*, 160(15):2259–2270, 2012. doi:10.1016/J.DAM.2012.05.019.
- 23 Athanasios L. Konstantinidis and Charis Papadopoulos. Cluster deletion on interval graphs and split related graphs. *Algorithmica*, 83(7):2018–2046, 2021. doi:10.1007/s00453-021-00817-8.
- 24 Harold W. Kuhn. The Hungarian Method for the Assignment Problem. In *50 Years of Integer Programming 1958-2008 - From the Early Years to the State-of-the-Art*, pages 29–47. Springer, 2010. doi:10.1007/978-3-540-68279-0_2.
- 25 Sebastian Ochs. Cluster deletion on unit disk graphs. Master’s thesis, Philipps-Universität Marburg, 2023. URL: https://www.fmi.uni-jena.de/fmi_femedia/fakultaet/institute-und-abteilungen/informatik/algorithm-engineering/master-thesis-sebastian-ochs.pdf.
- 26 Ron Shamir, Roded Sharan, and Dekel Tsur. Cluster graph modification problems. *Discrete Applied Mathematics*, 144(1-2):173–182, 2004. doi:10.1016/J.DAM.2004.01.007.
- 27 Dekel Tsur. Faster parameterized algorithm for cluster vertex deletion. *Theory of Computing Systems*, 65(2):323–343, 2021. doi:10.1007/s00224-020-10005-w.
- 28 Dekel Tsur. Cluster deletion revisited. *Information Processing Letters*, 173:106171, 2022. doi:10.1016/J.IPL.2021.106171.