



Correlation Clustering with Vertex Splitting

Matthias Bentert 

University of Bergen, Norway

Alex Crane  

University of Utah, Salt Lake City, UT, USA

Pål Grønås Drange  

University of Bergen, Norway

Felix Reidl  

Birkbeck, University of London, UK

Blair D. Sullivan  

University of Utah, Salt Lake City, UT, USA

Abstract

We explore CLUSTER EDITING and its generalization CORRELATION CLUSTERING with a new operation called *permissive vertex splitting* which addresses finding overlapping clusters in the face of uncertain information. We determine that both problems are NP-hard, yet they exhibit significant differences in terms of parameterized complexity and approximability. For CLUSTER EDITING WITH PERMISSIVE VERTEX SPLITTING, we show a polynomial kernel when parameterized by the solution size and develop a polynomial-time 7-approximation. In the case of CORRELATION CLUSTERING, we establish para-NP-hardness when parameterized by the solution size and demonstrate that computing an $n^{1-\varepsilon}$ -approximation is NP-hard for any constant $\varepsilon > 0$. Additionally, we extend an established link between CORRELATION CLUSTERING and MULTICUT to the setting with permissive vertex splits.

2012 ACM Subject Classification Theory of computation → Facility location and clustering; Theory of computation → Parameterized complexity and exact algorithms; Mathematics of computing → Approximation algorithms; Theory of computation → Problems, reductions and completeness

Keywords and phrases graph modification, cluster editing, overlapping clustering, approximation, parameterized complexity

Digital Object Identifier 10.4230/LIPIcs.SWAT.2024.8

Related Version *Full Version:* <https://arxiv.org/abs/2402.10335>

Funding *Alex Crane, Felix Reidl, and Blair D. Sullivan:* Gordon & Betty Moore Foundation’s Data Driven Discovery Initiative under award GBMF4560 to Blair D. Sullivan.

Matthias Bentert: European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No. 819416).

1 Introduction

Discovering clusters, or communities, is a core task in understanding the vast amounts of relational data available. One limitation of many traditional clustering algorithms is the necessity of specifying a desired number of clusters as part of the input. The problem known as CLUSTER EDITING avoids this by instead aiming to minimize the number of edge insertions and removals necessary to transform the input into a *cluster graph* (a disjoint union of cliques). This problem has been heavily studied in the graph-algorithms community, and was first proved to be fixed-parameter tractable with respect to the number of edge modifications (k) by Cai in 1996 [12]. The running time has significantly improved since, with the best known algorithm running in $O(1.62^k(n + m))$ time [11]. The problem also admits a polynomial kernel with $2k$ vertices [16].



© Matthias Bentert, Alex Crane, Pål Grønås Drange, Felix Reidl, and Blair D. Sullivan; licensed under Creative Commons License CC-BY 4.0

19th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2024).

Editor: Hans L. Bodlaender; Article No. 8; pp. 8:1–8:17



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

8:2 Correlation Clustering with Vertex Splitting

Formally in CLUSTER EDITING, we consider a complete graph where each edge is labeled as positive (which we imagine as colored *blue*) or negative (colored *red*) and we ask for the minimum number of edges whose color must be changed so that there is a partition of the vertex set where all edges within each part are blue, and all edges between parts are red. This convention of an edge-labeled complete graph will be useful in our setting and easily maps onto the more common formalism for CLUSTER EDITING with an incomplete, uncolored graph as input (imagine the graph edges as blue and its non-edges as red). We also note that other conventions for labelling positive/negative edges exist in the literature, e.g. using labels like $\langle + \rangle$ and $\langle - \rangle$.

In practice, the positive or negative association between objects is usually computed using a similarity metric which we can think of as an oracle function which, given two objects, computes a score that expresses the (dis)similarity of the inputs. For large-scale data, the assumption of complete information is then unrealistic for two reasons: First, the quadratic complexity of computing all pairwise associations is prohibitively expensive. Second, the similarity oracle may be unable to provide a clear answer for certain pairs – suggesting that objects can either be grouped together or kept separate, depending on other parts of the data or even external domain context.

Consequently, we should also consider cases in which the input is an incomplete graph with positive and negative labels on the existing edges and no information about pairs not joined by an edge. This scenario has previously been investigated by Demaine et al. [20] by allowing “0-weight edges” (zero-edges) in their cluster-editing framework¹. For clarity, we will refer to the problem where zero-edges (non-edges) are allowed as CORRELATION CLUSTERING and to the problem where the input graph is *complete* – i.e. every vertex pair is connected either by a blue or a red edge – as CLUSTER EDITING.

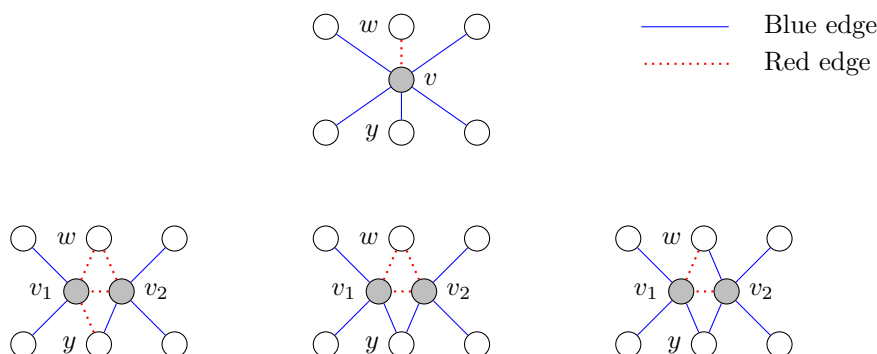
The approximability of both CLUSTER EDITING and CORRELATION CLUSTERING are well-studied. First considered by Bansal, Blum, and Chawla [8], under the name *correlation clustering*², CLUSTER EDITING admits a 1.73-approximation [17] when minimizing the number of disagreements (red edges within and blue edges between clusters). Other variants of CLUSTER EDITING which maximize the number of agreements or the correlation (agreements minus disagreements) admit a PTAS (polynomial-time approximation scheme) and a $\Omega(\log n)$ -approximation, respectively [8, 14]. In the more general setting of minimizing disagreements for CORRELATION CLUSTERING (i.e., when zero-edges are present but never constitute a disagreement), an $O(\log n)$ -approximation is known [20]. This result arises from the strong relation between CORRELATION CLUSTERING and MULTICUT³. The connection was first observed with MULTIWAY CUT by Bansal, Blum, and Chawla [8], before an approximation-preserving reduction from MULTICUT to CORRELATION CLUSTERING was given independently by both Charikar, Guruswami, and Wirth [13] and Demaine et al. [20]. The connection to MULTICUT also implies that no constant-factor approximation is possible for CORRELATION CLUSTERING, unless the Unique Games Conjecture is false [15].

These algorithmic advances provide a positive outlook on applying these clustering variants in practice, however, we need to also investigate whether the proposed clustering model could be improved. In particular, we need to question the underlying assumption that real-world data segregates into neat, disjoint clusters. The following domain examples illustrate why this assumption is probably too optimistic:

¹ In the version discussed by Demaine et al. [20], real weights are assigned to edges, reflecting the certainty level of the oracle in determining the similarity between objects. We only consider weights in $\{-1, 0, 1\}$, a common restriction in the literature.

² There is significant inconsistency in the literature regarding the nomenclature of these problems; as stated, we reserve the name CORRELATION CLUSTERING for the problem where the input is incomplete.

³ Given a set of pairs of terminals, $(s_1, t_1), (s_2, t_2), \dots, (s_p, t_p)$, find a set of at most k edges such that after removing these edges, every pair (s_i, t_i) is disconnected



■ **Figure 1** A vertex v in an (incomplete) correlation graph (top). The bottom row gives toy examples of exclusive (left), inclusive (center), and permissive (right) vertex splits of v into v_1 and v_2 . For clarity, some red edges incident to v_1 and v_2 are omitted from each figure on the bottom row.

- Document classification: Individual documents often span multiple topics and should therefore belong to multiple topic-clusters;
- Sentiment analysis: A single piece of text can express very different emotions (e.g. sadness mixed with humor);
- Community detection: Individuals typically participate in multiple communities, such as family, professional, and hobbyist groups.
- Language processing: Homonyms like “bat” should belong both to an “animal” cluster as well as a “sports-equipment” cluster.

Hence, the emphasis in clustering has recently shifted towards algorithms for *overlapping clustering* [3, 4, 5, 6, 7, 9, 10, 18, 19, 22, 23, 24, 25, 26, 27, 28]. These models move away from the requirement that data must be partitioned into disjoint subsets by considering a variety of definitions for clusters which may intersect. One natural approach is to edit to a more general target graph class (instead of a cluster graph, consider minimizing the number of edge modifications required to achieve some more complex structure that exhibits strong community structure but allows overlap), but it is difficult to define generalizations that align with many applications.

Motivated by this, Abu-Khizam et al. [3] proposed an alternative model for overlapping clustering based on the concept of *splitting* a vertex into two new vertices, representing an object having two distinct roles within a dataset. This approach led to the problem CLUSTER EDITING WITH VERTEX SPLITTING, where edges can be added or deleted, and vertices can be split. Here, *splitting* a vertex v means replacing it with two copies, v_1 and v_2 , ensuring the union of their (blue) neighbor sets equals the original vertex’s (blue) neighbor set. In fact, Abu-Khizam et al. [3] propose two different vertex splitting operations: one (*exclusive* splitting) where v_1 and v_2 are required to have disjoint (blue) neighborhoods, and another (*inclusive* splitting) where they are allowed to share (blue) neighbors. See Figure 1 for an example. Abu-Khizam et al. [1] show that CLUSTER EDITING WITH VERTEX SPLITTING is NP-hard and has a $6k$ -vertex kernel, where k is the number of edits (edge modifications/vertex splits) allowed. The approximability of this problem remains unknown.

A significant limitation of both existing notions of vertex splitting is that they require red edges to be preserved by both copies of a split vertex. For example, consider a red edge uv in data arising from word classification, where u and v correspond to “bat” and “cat”, respectively. It could be that the edge was produced by our oracle as a result of “bat” being interpreted as a piece of sports equipment, not an animal. However, when “bat” is split

so that each meaning has its own vertex, we wish to retain the red edge only on one of the copies of v (the one *not* corresponding to the small flying mammal, as this does have similarities with a cat). Motivated by this, we introduce a new operation called *permissive vertex splitting* which allows replacing a vertex v with two copies v_1 and v_2 with the restriction that if uv is a blue edge (or red edge, respectively), then at least one of uv_1 and uv_2 is a blue edge (red edge, respectively). Beyond that, we are free to choose what to do with the newly-created neighborhoods. We call the new problem variant, where edges can be added or deleted and vertices can be permissively split, CORRELATION CLUSTERING WITH PERMISSIVE VERTEX SPLITTING (CCPVS). We show that sequences of permissive vertex splits solving this problem correspond directly to a natural notion of overlapping clustering (see Definition 4), adding to the motivation for this definition of splitting.

Extending the prior work relating CORRELATION CLUSTERING to MULTICUT, we show that CCPVS can be reduced to the new problem MULTICUT WITH VERTEX SPLITTING (MCVS) and vice versa, meaning that the computational complexities of these problems are essentially the same. We then show that MCVS, and hence also CCPVS, are para-NP-hard (with respect to solution size), and NP-hard to approximate within an $n^{1-\epsilon}$ factor for any $\epsilon > 0$. Because of the inherent hardness of CCPVS, we then turn our attention to the setting where there are no zero-edges, i.e., to CLUSTER EDITING WITH PERMISSIVE VERTEX SPLITTING (CEPVS). We show that this problem remains NP-hard, but on the positive side admits a polynomial kernel (and thus is fixed-parameter tractable). Finally, we give a polynomial-time algorithm which provides a 7-approximation for CEPVS.

2 Preliminaries

We refer the reader to the textbook by Diestel [21] for standard graph-theoretic definitions and notation. A *star* is a tree with exactly one internal vertex. In particular, a star has at least two leaves. A *red clique* is a clique in which all edges are red. A *blue clique* is defined similarly. For a positive integer n , we denote by $[n] = \{1, 2, \dots, n\}$ the set of all positive integers up to n . An *incomplete correlation graph* is a simple, unweighted, and undirected graph $G = (V, B, R)$ with two disjoint edge relations B (blue) and R (red). If such a graph is complete, i.e., $B \cup R = \binom{V}{2}$, then we call it a *correlation graph*. For a vertex $v \in V$ we write $N^R(v)$ to denote the set of neighbors adjacent to v via red edges (*red neighbors*) and $N^B(v)$ for those adjacent via blue edges (*blue neighbors*). A *cluster graph* is a correlation graph in which the blue edges form vertex-disjoint cliques (and thus all edges between the cliques are red). We can now formally define our vertex-splitting operation.

► **Definition 1.** A permissive vertex split of a vertex v in an (incomplete) correlation graph G is the replacement of v in G with two new vertices v_1 and v_2 such that

- $N^R(v) \subseteq N^R(v_1) \cup N^R(v_2)$, and
- $N^B(v) \subseteq N^B(v_1) \cup N^B(v_2)$,

In other words, we create a new graph where every red (blue) neighbor of v is a red (blue) neighbor of at least one of v_1 or v_2 . All other “edges” incident to v_1 and v_2 can be chosen arbitrarily. In particular, in incomplete correlation graphs, we can assume that all these other edges are neutral (i.e., the “edges” do not exist), while in correlation graphs, it is usually simpler to make these edges either red or blue to keep the graph complete. Notably, the edge v_1v_2 can always be assumed to be a red edge as splitting a vertex into two vertices that end up in the same (blue) connected component is never advantageous. For the remainder of this text, unless otherwise specified all vertex splits are permissive. Given a sequence $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_k)$ of k vertex splits performed on an (incomplete) correlation

graph, we denote the resulting (incomplete) correlation graph by $G_{|\sigma}$. Each vertex u in $G_{|\sigma}$ corresponds to exactly one vertex v in G . We say that v is u 's *ancestor*, and that u is a *descendant* of v . If $u = v$, then u and v are *unsplit* vertices. Otherwise we say that v is a *split* vertex and that u is the descendant of a split vertex.

► **Definition 2.** *An erroneous cycle is a simple cycle that contains exactly one red edge. An (incomplete) correlation graph G contains an erroneous cycle if it contains a subgraph that is an erroneous cycle. A bad triangle is an erroneous cycle of length 3.*

Erroneous cycles are the canonical obstruction in CORRELATION CLUSTERING [13, 20], and bad triangles are the canonical obstruction in CLUSTER EDITING. Usually, these problems are formulated as edge editing problems, i.e., delete a minimum number of edges (blue or red) such that the resulting graph has no erroneous cycles/bad triangles. Previous work on CLUSTER EDITING with (inclusive or exclusive) vertex splitting has allowed both edge edits and vertex splits as editing operations [3, 2, 1, 5]. However, we note that permissive vertex splitting is flexible enough to capture all editing operations. First, note that in the setting with blue and red edges, each edge-editing operation can be seen as changing the color of an edge. Now, consider any solution σ in which the color of an edge uv is changed. Then, we construct a new sequence of the same length where this edge edit is replaced by a vertex split. We choose one endpoint (without loss of generality v) and split it into v_1 and v_2 . The neighborhood of v_1 is exactly the neighborhood of the initial vertex v except that the edge towards u has the other color. If the edge uv was initially red, then the vertex v_2 has all vertices in the graph as red neighbors. If the edge uv was blue, then we add blue edges between v_2 and all vertices that end up in the same (blue) connected component as (one descendant of) u in $G_{|\sigma}$. The result of the edge edit is now modeled exactly by v_1 and the operation is safe because v_2 cannot participate in any erroneous cycle as it is a twin of (one descendant of) u . Moving forward, we assume that all editing operations are vertex splits, and we say that a sequence σ of vertex splits *clusters* an (incomplete) correlation graph G if $G_{|\sigma}$ has no erroneous cycles. We now state the problems that we study:

CORRELATION CLUSTERING WITH PERMISSIVE VERTEX SPLITTING (CCPVS)

Input: An (incomplete) correlation graph G and a non-negative integer k .

Problem: Does there exist a sequence σ of at most k vertex splits which clusters G ?

CLUSTER EDITING WITH PERMISSIVE VERTEX SPLITTING (CEPVS) is the same problem restricted to correlation graphs. We conclude this section with our main structural insight, which is that clustering an (incomplete) correlation graph G via a sequence of vertex splits is equivalent to performing a very natural notion of overlapping clustering on the vertices of G .

► **Definition 3.** *A covering of an (incomplete) correlation graph $G = (V, E)$ is a set family $\mathcal{F} \subseteq 2^V$ such that $\bigcup \mathcal{F} = V$. The cost of the covering \mathcal{F} is*

$$\text{cost}_G(\mathcal{F}) = \sum_{v \in V} (\#\mathcal{F}(v) - 1),$$

where $\#\mathcal{F}(v) := |\{X \mid v \in X \in \mathcal{F}\}|$ counts the number of sets in \mathcal{F} which contain v .

8:6 Correlation Clustering with Vertex Splitting

► **Definition 4.** An overlapping clustering of an (incomplete) correlation graph G is a covering \mathcal{F} with the following two properties:

- for every blue edge $uv \in B$, there exists at least one cluster $X \in \mathcal{F}$ with $\{u, v\} \subseteq X$, and
- for every red edge $uv \in R$, there exists two distinct clusters $X, Y \in \mathcal{F}$ with $u \in X$ and $v \in Y$.

For a specific edge uv , we say that a clustering covers the edge if it is blue and the first condition holds and we say that it resolves the edge if it is red and the second condition holds.

► **Lemma 5.** An (incomplete) correlation graph G can be clustered with k vertex splits if and only if G has an overlapping clustering of cost k .

Proof. For the first direction, let σ be a sequence of k vertex splits clustering $G = (V, B, R)$, i.e., σ produces a graph $G_{|\sigma} = (V_{|\sigma}, E_{|\sigma})$ with no erroneous cycles. We will construct an overlapping clustering \mathcal{F} of cost at most k . We note that it is easy to extend any such overlapping clustering to one of cost exactly k . We begin by choosing an arbitrary vertex $v \in V_{|\sigma}$. We denote by v^* the ancestor of v in V . Let $C_v \subseteq V_{|\sigma}$ be the vertices of the connected component of v in the subgraph of $G_{|\sigma}$ induced by all blue edges, and C_{v^*} be the set of corresponding ancestor vertices in V . We add C_{v^*} to \mathcal{F} and remove C_v from $G_{|\sigma}$. We repeat this process exhaustively. The resulting \mathcal{F} is a covering of G , as each vertex in V has at least one descendant in $V_{|\sigma}$. Moreover, our construction guarantees that each vertex in $V_{|\sigma}$ is considered exactly once. Consequently, for each vertex $v \in V$ we have that $\#\mathcal{F}(v)$ is no greater than the number of descendants of v in $V_{|\sigma}$. Thus, \mathcal{F} has cost at most k . Each blue edge is covered by construction.

For the final step, we show how to augment \mathcal{F} such that all red edges are resolved while maintaining that $\text{cost}_G(\mathcal{F}) \leq k$. We begin by identifying some red edge uv which is not resolved by \mathcal{F} . This implies that each of u and v are contained in exactly one cluster $X \in \mathcal{F}$. The red edge uv implies that there is some red edge u_1v_1 in $G_{|\sigma}$, where u_1 is a descendant of u and v_1 is a descendant of v . Moreover, the construction of \mathcal{F} guarantees that there is some blue path between v_1 and a descendant of u , but this latter descendant cannot be u_1 or else we have identified an erroneous cycle in $G_{|\sigma}$. Thus, u has multiple descendants in $G_{|\sigma}$ and is therefore a split vertex. Since u is a split vertex but is only contained in one cluster X in \mathcal{F} , we can add the cluster $\{u\}$ to \mathcal{F} , thereby resolving uv , while maintaining that $\text{cost}_G(\mathcal{F}) \leq k$. We repeat this process until all red edges are resolved.

For the other direction, let \mathcal{F} be an overlapping clustering of G with cost k . For each vertex v that is contained in more than one cluster set in \mathcal{F} , we split v a total of $\#\mathcal{F}(v) - 1$ times. We assign each descendant to one set $X \in \mathcal{F}$ with $v \in X$ and we create blue edges towards all other vertices that are contained in X (or to the specific descendant of a vertex in X that was also assigned to X). All other edges incident to the descendant of v are red. We first show that this construction indeed corresponds to a series of vertex splits. For each blue edge uv , we have that there is some cluster set $X \in \mathcal{F}$ with $u, v \in X$. Hence, if u and/or v are split, then the blue edge uv corresponds to the blue edge between the two copies of u and v that are assigned to X . For each red edge uv , we have that there are some cluster sets $X \neq Y \in \mathcal{F}$ with $u \in X$ and $v \in Y$. Hence, if u and/or v are split, then the red edge uv corresponds to the red edge between (the descendant of) u assigned to X and (the descendant of) v that is assigned to Y . Moreover, we did exactly $\text{cost}(\mathcal{F})$ splits.

It remains to show that the sequence of splits results in a graph that contains no erroneous cycles. Suppose that an erroneous cycle $(u = v_0, v_1, \dots, v_p = w, u)$ with red edge uw remains. Note that each vertex is assigned to exactly one cluster set in \mathcal{F} as each unsplit vertex is contained in exactly one set in \mathcal{F} and each descendant of a split vertex is assigned to a

cluster set by construction. We will show that there is no blue edge between vertices that are assigned to different clusters and no red edge between vertices that are assigned to the same cluster set. This finishes the proof as u and w are then assigned to different cluster sets as they share a red edge, but w_i and w_{i-1} are assigned the same cluster set for each $i \in [p]$, a contradiction. First, assume that there is a blue edge xy where x and y are assigned to different cluster sets. If x and y are both unsplit vertices, then the blue edge between them is not covered by \mathcal{F} , a contradiction. Hence, at least one of the two vertices is the descendant of a split vertex and by construction, all edges to vertices that are assigned to different cluster sets are red. Now assume that there is a red edge xy where x and y are assigned to the same cluster set $X \in \mathcal{F}$. Again, if x and y are both unsplit vertices, then they are only contained in X in \mathcal{F} and hence the red edge between them is not resolved by \mathcal{F} , a contradiction. So at least one of the two vertices is the descendant of a split vertex and by construction, all edges to vertices that are assigned to X are blue, a final contradiction. This concludes the proof. ◀

3 Incomplete Information

We first consider the more general problem, CORRELATION CLUSTERING WITH PERMISSIVE VERTEX SPLITTING, which allows for incomplete information. Without vertex splits, it has long been known that CORRELATION CLUSTERING is in fact equivalent to MULTICUT [20], which is the problem of deleting a minimum number of edges from a graph $G = (V, E)$ such that every *terminal pair* of distinct vertices in a set $S \subseteq \binom{V}{2}$ is separated in the resulting graph. We define MULTICUT WITH VERTEX SPLITTING (MCVS) and show that it is equivalent to CCPVS. We believe that this result is of independent interest, but it will also prove immediately useful as it facilitates the main results of this section. Specifically, CCPVS and MCVS are both **para-NP-hard** when parameterized by the number of vertex splits, and for any $\varepsilon > 0$ it is NP-hard to approximate either problem within a $n^{1-\varepsilon}$ factor.

First we must define our new MULTICUT variant. In this context we use standard graph terminology, i.e., we discuss simple, unweighted, and undirected graphs with a single edge relation E . Note that this is equivalent to a correlation graph where edges in E are blue and all other vertex pairs are red, so permissive vertex splits are still well-defined. However, in the MULTICUT context we can safely assume that all vertex splits are *exclusive*, i.e., whenever splitting a vertex v into descendants v_1 and v_2 we have that $N(v_1) \cup N(v_2) = N(v)$ and $N(v_1) \cap N(v_2) = \emptyset$. The reason is that in MULTICUT it is never advantageous to assign more edges than required. Note that in the classic version of MULTICUT, it does not make sense to have an edge between two vertices of a terminal pair. We decided to keep this restriction as it streamlines some of the following arguments. A related technical detail to discuss is what happens to a terminal pair when one of its two vertices is split. We work with the variant where the terminal pair is simply removed in this case. Note that this is equivalent to the variant where we can choose either of the descendants to replace the original vertex in the terminal pair, since, as previously mentioned, we may always assume that any two descendants of the same vertex end up in different connected components.

MULTICUT WITH VERTEX SPLITTING (MCVS)

Input: A graph $G = (V, E)$, an integer k , and a set $S \subseteq \binom{V}{2}$ of *terminal pairs* with $S \cap E = \emptyset$.

Problem: Does there exist a sequence σ of at most k (exclusive) vertex splits such that each pair in S is separated in $G|_\sigma$?

8:8 Correlation Clustering with Vertex Splitting

We now show that CCPVS and MCVS are equivalent problems. Let $(G = (V, B, R), k)$ be an instance of CCPVS. We construct an equivalent instance $(H = (V', E'), S, k)$ of MCVS as follows. For each vertex $v \in V$ we create a vertex v' in V' . Additionally, for each blue edge $uw \in B$ we add the edge $u'w'$ to E' . Finally, for each red edge $uw \in R$ we add the terminal pair (u', w') to S . This completes the construction of H .

► **Theorem 6.** *For any integer $k \geq 0$, (G, k) is a yes-instance of CORRELATION CLUSTERING WITH PERMISSIVE VERTEX SPLITTING if and only if (H, S, k) is a yes-instance of MULTICUT WITH VERTEX SPLITTING.*

Proof. For the first direction, let $\sigma = (\sigma_1, \sigma_2, \dots)$ be a sequence of vertex splits clustering G . We will construct a sequence σ' of the same length which separates each pair in S by considering each σ_i in order. If σ_i splits vertex $v \in V$ into v_1 and v_2 then σ'_i splits v' into v'_1 and v'_2 . By construction, each neighbor u' of v' corresponds to a blue neighbor u of v . If u is a blue neighbor of v_1 , then we create the edge v'_1u' . Otherwise, we create the edge v'_2u' . This completes the construction of σ' . Now, we assume toward a contradiction that some terminal pair (v', u') is connected in $H_{|\sigma'}$. Then there is some path $(v' = w'_0, w_1, \dots, w'_p = u')$ in $H_{|\sigma'}$. Note that our construction ensures that this path contains at least two edges, and that there is a corresponding blue path $(v = w_0, w_1, \dots, w_p = u)$ in $G_{|\sigma}$. Moreover, because (v', u') is a terminal pair in $H_{|\sigma'}$, vu is a red edge in $G_{|\sigma}$. Thus, we have identified an erroneous cycle $(v = w_0, w_1, \dots, w_p = u, v)$ in $G_{|\sigma}$, contradicting that σ clusters G .

For the other direction, let $\sigma' = (\sigma'_1, \sigma'_2, \dots)$ be a sequence of vertex splits such that no terminal pair is connected in $H_{|\sigma'}$. As before, we will construct a solution σ of the same length by considering each σ'_i in order. If σ'_i splits v' into v'_1 and v'_2 , then we split the corresponding vertex v into v_1 and v_2 as follows. If v' is a terminal with partner u' , then our construction guarantees that vu is a red edge in G . We create the red edge v_1u if v'_1 (or one of its descendants) is in a different component from u (or one of its descendants) in $H_{|\sigma'}$. Otherwise, we create the red edge v_2u . We mark the relevant pair of descendants so that, when performing subsequent splits, the red edge is always assigned such that its endpoints in G_σ correspond to vertices in different connected components of $H_{|\sigma'}$. Next, for each $u' \in N(v')$, we create the blue edge v_1u if σ'_i assigns u' to $N(v'_1)$. Otherwise, we create the blue edge v_2u . We now assume toward a contradiction that there is an erroneous cycle $(v = w_0, w_1, \dots, w_p = u, v)$ in $G_{|\sigma}$, with vu being the red edge. The blue path $(v = w_0, w_1, \dots, w_p = u)$ guarantees that there is a path $(v' = w'_0, w'_1, \dots, w'_p = u')$ from v' to u' in $H_{|\sigma'}$, and this together with the red edge vu implies that (v', u') is a terminal pair. This contradicts that no terminal pair is connected in $H_{|\sigma'}$. ◀

To reduce MCVS to CCPVS, we simply reverse the previous reduction of CCPVS to MCVS. Formally, let $(G = (V, E), S, k)$ be an instance of MCVS. We create an instance $(H = (V', B, R), k)$ of CCPVS as follows. For each vertex $v \in V$ we add vertex v' to V' , for each edge $uw \in E$ we add the blue edge $u'w'$ to B , and finally for each terminal pair $(u, v) \in S$, we add the red edge $u'v'$ to R . Note that B and R are disjoint, as by definition no terminal pair in S is also an edge in E .

► **Theorem 7.** *For any integer $k \geq 0$, (G, S, k) is a yes-instance of MULTICUT WITH VERTEX SPLITTING if and only if (H, k) is a yes-instance of CORRELATION CLUSTERING WITH PERMISSIVE VERTEX SPLITTING.*

Proof. Note that applying the reduction behind Theorem 6 to H results in the instance (G, S, k) . Thus, Theorem 6 already shows that the two instances are equivalent. ◀

Theorems 6 and 7, together with the observation that both reductions exactly preserve the number of vertices, allow us to state the following strong notion of equivalence between CCPVS and MCVS.

► **Corollary 8.** *For any function f , CCPVS admits a kernel of size $f(k)$ if and only if MCVS does. Furthermore, the minimization variant of CCPVS admits a polynomial-time $f(n)$ -approximation algorithm if and only if the minimization variant of MCVS does.*

Now that we have established the equivalence of MCVS and CCPVS, we are ready to show the hardness of both problems.

► **Theorem 9.** *MCVS is NP-hard even if $k = 2$. Additionally, for any $\varepsilon > 0$ it is NP-hard to approximate MCVS to within a factor of $n^{1-\varepsilon}$.*

Proof. Let $G = (V, E)$ be the input graph for k -COLORABILITY with $k \geq 3$. We will construct an equivalent input instance $(H, S, k - 1)$ for MCVS. We construct the graph H and terminal set S from G as follows. We add V to H and for each edge $uv \in E$, we add (u, v) to S . We then add a new vertex a to H , and create edges from a to all other vertices. This completes the construction.

We first argue that we may assume that any solution of $(H, S, k - 1)$ *only* splits a . To see this, let σ be a sequence of vertex splits of length at most $k - 1$ such that all terminal pairs are disconnected in $H_{|\sigma}$. Suppose that some vertex $v \neq a$ is split. Before this split, v only has one neighbor a^* , which is either equal to a or a descendant of a . We simply replace the split of v with a split of a^* into a_1^* and a_2^* such that $N(a_1^*) = \{v\}$ and $N(a_2^*) = N(a^*) \setminus \{v\}$. In the resulting graph, v is disconnected from all other vertices in V , and so it is disconnected from all of its terminal partners. We proceed with the assumption that in any solution a is the only split vertex.

Assume that $(H, S, k - 1)$ is a yes-instance. We show that then G is k -colorable. By the above, $H_{|\sigma}$ contains k descendants a_1, \dots, a_k of a and these vertices naturally partition the set V into k sets $C_i = N(a_i)$ for $i \in [k]$. No terminal-pair can appear with both endpoints in one of these sets so the same holds for $E \subseteq S$. Hence, C_1, \dots, C_k is a valid k -coloring of G .

In the other direction, assume that G has a k -coloring with the color partition C_1, \dots, C_k . Then we can split $a \in H$ a total of $k - 1$ times into descendants a_1, \dots, a_k such that $N(a_i) = C_i$. Since $\{a_1, \dots, a_k\}$ is independent it is easy to verify that these $k - 1$ splits separate every terminal pair in S .

We conclude that MCVS is already NP-hard with parameter $k = 2$ as the above provides a reduction from 3-COLORABILITY. The approximation hardness follows directly from the facts that, given any constant $\varepsilon > 0$, computing an $n^{1-\varepsilon}$ -approximation for CHROMATIC NUMBER is NP-hard [29], and that our constructed instance of MCVS has only $n + 1$ vertices. ◀

Taken together with Corollary 8, Theorem 9 gives us the same result for CCPVS.

► **Corollary 10.** *CCPVS is NP-hard even if $k = 2$. Additionally, for any $\varepsilon > 0$ it is NP-hard to approximate CCPVS to within a factor of $n^{1-\varepsilon}$.*

4 Complete Information

We now restrict our study to correlation graphs, i.e., we study CLUSTER EDITING WITH PERMISSIVE VERTEX SPLITTING. Our main results are NP-hardness (Section 4.1), a polynomial kernel (Section 4.2), and a polynomial-time 7-approximation (Section 4.3). We begin by introducing a new structure and subsequent lemmas which will be helpful in attaining the latter two results.

8:10 Correlation Clustering with Vertex Splitting

► **Definition 11.** A bad star S in a correlation graph G is a set $\{v_0, v_1, \dots, v_{|S|-1}\}$ of vertices where all edges in $\{\{v_0, v_i\} \mid i \in [|S|-1]\}$ are blue and all edges in $\{\{v_i, v_j\} \mid i \neq j \in [|S|-1]\}$ are red. The vertex v_0 is called the center and all other vertices are called leaves. The weight of a bad star $\text{weight}(S)$ is the number of leaves in the star minus one. A bad star forest is a collection T of vertex-disjoint bad stars. We write $\text{weight}(T) := \sum_{S \in T} \text{weight}(S)$ to denote the sum of weights of its members. A correlation graph G contains a bad star forest if it contains a subgraph which is a bad star forest.

The first lemma states a useful lower bound in terms of bad stars.

► **Lemma 12.** If G contains a bad star forest of weight k then we need at least k vertex splits to cluster G .

Proof. We begin by showing that if $G = (V, B, R)$ contains a (not necessarily induced) subgraph $H = (V_H, B_H, R_H)$ and at least k vertex splits are needed to separate each pair in S , then at least k vertex splits are needed to cluster G . Suppose otherwise. Then, using Lemma 5, there is some overlapping clustering \mathcal{F} of G with cost less than k . We will construct an overlapping clustering \mathcal{F}_H of H with cost less than k . We begin by setting $\mathcal{F}_H = \{X \cap V_H \mid X \in \mathcal{F}\}$. It is clear that this is a covering of H , that every blue edge is covered, and that $\#\mathcal{F}_H(v) \leq \#\mathcal{F}(v)$ for every vertex $v \in V_H$. We now ensure that each red edge is resolved. Let uv be a red edge which is not resolved, so each of u and v belong to only a single cluster $X \in \mathcal{F}_H$. In this case we claim that \mathcal{F} contains two distinct clusters $Y \neq Z$ such that $Y \cap V_H = Z \cap V_H = X$. Otherwise, either \mathcal{F} does not resolve uv or one of u or v is contained in multiple clusters of \mathcal{F}_H , both contradictions. Thus, we can safely add the cluster $\{u\}$ (chosen without loss of generality) to \mathcal{F}_H , thereby resolving uv while maintaining that $\#\mathcal{F}_H(u) \leq \#\mathcal{F}(u)$. We repeat this process iteratively until all red edges are resolved. In doing so, we produce an overlapping clustering \mathcal{F}_H of H with $\text{cost}_H(\mathcal{F}_H) \leq \text{cost}_G(\mathcal{F}) < k$, a contradiction.

It remains to show that a bad star forest of weight k requires at least k vertex splits to cluster. We begin by showing that a bad star S of weight k requires at least k vertex splits. Let x be the center vertex of S and let \mathcal{F} be an overlapping clustering of S . Let the clusters in \mathcal{F} which contain x be C_1, C_2, \dots, C_p . Moreover for each $1 \leq i \leq p$, let $\hat{C}_i = C_i \setminus \{x\}$. Note that we may assume each \hat{C}_i is nonempty, as the cluster $\{x\}$ covers no blue edges and resolves no red edges in S , and can therefore be safely removed from \mathcal{F} . Observe also that each leaf v of S must be contained in some set \hat{C}_i since the edge xv is blue. Now suppose that some leaf v is contained in two sets $\hat{C}_i \neq \hat{C}_j$. We remove v from the cluster C_j (chosen arbitrarily) and add the cluster $\{v\}$ to \mathcal{F} . The blue edge xv is still covered by C_i , the cluster $\{v\}$ ensures that all red edges incident to v are still resolved, and we have not increased the cost of the clustering. Thus, we may safely assume that the sets $\hat{C}_1, \hat{C}_2, \dots, \hat{C}_p$ are a partition of the leaves of S . Consider one such set \hat{C}_i . These leaves induce a red clique and none of these red edges is resolved by C_i , so we have that at least $|\hat{C}_i| - 1$ of these leaves are contained in multiple clusters in \mathcal{F} . Since we also know that $\#\mathcal{F}(x) = p$, we conclude

$$\begin{aligned} \text{cost}_S(\mathcal{F}) &\geq (|\hat{C}_1| - 1) + (|\hat{C}_2| - 1) + \dots + (|\hat{C}_p| - 1) + \#\mathcal{F}(x) - 1 \\ &= |\hat{C}_1| + |\hat{C}_2| + \dots + |\hat{C}_p| - p + p - 1 = |S \setminus \{x\}| - 1 = \text{weight}(S) = k \end{aligned}$$

Finally, let T be a bad star forest made up of t bad stars S_1, S_2, \dots, S_t . Let k be the weight of T and suppose toward a contradiction that T admits an overlapping clustering \mathcal{F} of cost less than k . Then, we repeat the technique from earlier in this proof to construct overlapping clusterings $\mathcal{F}_{S_1}, \mathcal{F}_{S_2}, \dots, \mathcal{F}_{S_t}$ of the bad stars. Because the bad stars are vertex-disjoint, we

have that $\text{cost}_{S_1}(\mathcal{F}_{S_1}) + \text{cost}_{S_2}(\mathcal{F}_{S_2}) + \dots + \text{cost}_{S_i}(\mathcal{F}_{S_i}) \leq \text{cost}_T(\mathcal{F}) < k$. This implies that there is some S_i such that $\text{cost}_{S_i}(\mathcal{F}_{S_i})$ is less than the weight of S_i , but we have already proven that this is impossible. ◀

The second lemma states that every optimal solution contains a cluster that contains all vertices of a sufficiently large blue clique.

► **Lemma 13.** *If a correlation graph G contains a blue clique C of size at least $k + 1$, then any overlapping clustering \mathcal{F} of cost at most k contains a set $X \in \mathcal{F}$ with $C \subseteq X$.*

Proof. Let C be a blue clique in G of size at least $k + 1$ and let \mathcal{F} be any overlapping clustering of cost at most k . Assume towards a contradiction that \mathcal{F} does not contain a set X with $C \subseteq X$. Since \mathcal{F} has cost at most k , there exists a vertex $v \in C$ that is contained in exactly one set $Y \in \mathcal{F}$. Moreover, since Y does not contain all vertices of C by assumption, there exists a vertex $u \in C \setminus Y$. Since C is a blue clique, the edge uv is blue and is covered by some set $Z \in \mathcal{F}$. Observe that $Y \neq Z$ as $u \in Z$ and $u \notin Y$. Since Z covers the edge uv , it holds that $v \in Z$, a contradiction to the assumption that v is only contained in Y . ◀

4.1 NP-hardness

We now show that CLUSTER EDITING WITH PERMISSIVE VERTEX SPLITTING is NP-hard.

► **Proposition 14.** *Deciding whether a given correlation graph admits an overlapping clustering of cost at most k is NP-hard.*

Proof. We reduce from VERTEX COVER. Let $(G = (V, E), k')$ be an instance of VERTEX COVER. We construct a correlation graph H as follows. Let U be a set of $k' + 1$ vertices (not contained in V). The vertex set of H is $U \cup V$. For each edge $e = uv \in E$, we add a red edge uv to H . All other edges (including all edges incident to a vertex in U) are blue. Finally, we set $k = k'$.

We next show that the reduction is correct. First assume that there is an overlapping clustering \mathcal{F} of H of cost at most k . By Lemma 13, for each vertex $v \in V$, there exists a set $X_v \in \mathcal{F}$ with $U \cup \{v\} \subseteq X_v$. Note that $U \subset X_u \cap X_v$ for any pair $u, v \in V$ and therefore $X_u = X_v$ as otherwise the cost of \mathcal{F} is at least $|U| > k$. Hence, there exists a set $X \in \mathcal{F}$ with $U \cup V \subseteq X$. Since all blue edges are covered by X , we next focus on resolving all red edges. Note that since X contains all vertices in H and the cost of \mathcal{F} is at most k , all remaining sets in $\mathcal{F}' = \mathcal{F} \setminus X$ contain at most k vertices combined. If for some red edge uv none of the two vertices u or v is contained in a set in \mathcal{F}' , then this red edge is not resolved by \mathcal{F} . Thus for each red edge, at least one of the two endpoints is contained in a set in \mathcal{F}' . Note that this immediately implies that G contains a vertex cover of size at most k (all vertices that are contained in a set in \mathcal{F}').

For the other direction, assume that G contains a vertex cover S of size at most k . We construct an overlapping clustering \mathcal{F} of H of cost at most k as follows. The family \mathcal{F} contains one set $X = U \cup V$ and for each vertex $v \in S$, it contains a set $X_v = \{v\}$. Note that the cost of \mathcal{F} is at most k and all blue edges in H are covered by X . Moreover, each red edge uw in H is resolved as by construction it holds that \mathcal{F} contains the set $X_u = \{u\}$ or $X_w = \{w\}$. Without loss of generality, let \mathcal{F} contain X_u . Then, the red edge uw is resolved as w is contained in X and u is contained in $X_u \neq X$. This concludes the proof. ◀

4.2 Polynomial Kernel

We next show that CLUSTER EDITING WITH PERMISSIVE VERTEX SPLITTING parameterized by k admits a polynomial kernel. Note that this is in stark contrast to the para-NP-hardness of CORRELATION CLUSTERING WITH PERMISSIVE VERTEX SPLITTING parameterized by k .

► **Theorem 15.** CLUSTER EDITING WITH PERMISSIVE VERTEX SPLITTING parameterized by the number of vertex splits admits a kernel with $O(k^3)$ vertices.

Proof. Let $(G = (V, B, R), k)$ be the input instance of CEPVS. We begin by computing an inclusion-maximal bad star forest T in G . If $\text{weight}(T) \geq k$, then we conclude, according to Lemma 12, that (G, k) is a no-instance and output an appropriate trivial kernel.

Otherwise let S be the vertices of T and note that $|S| \leq 3 \text{weight}(T) \leq 3k$. Since T is inclusion-maximal, we know that $G \setminus S$ cannot contain any bad stars and in particular no bad triangles. We conclude that $G \setminus S$ is therefore a cluster graph. Let C_1, C_2, \dots, C_p be these clusters. We next exhaustively apply the following simple reduction rule.

► **Reduction rule 1.** If G contains a blue clique C such that all edges with one endpoint in C are red, then remove C from G .

Next, we bound the number p of cliques in $G \setminus S$ as follows. Assume that $G \setminus S$ contains at least $4k + 1$ cliques. Note that by application of Reduction Rule 1, all clusters in $G \setminus S$ have at least one blue edge towards S . Pick for each clique C in $G \setminus S$ one such blue edge towards S and let v_C be the endpoint in C of this edge. Note that these chosen edges form a collection of vertex-disjoint stars with all centers in S (but not necessarily all vertices in S being centers). Moreover, since the vertices v_C and $v_{C'}$ belong to different cliques for each pair $C \neq C'$ of cliques in $G \setminus S$, the edge between the two is red. Hence, all stars with at least two leaves in $V \setminus S$ are bad stars. Let $S' \subseteq S$ be the set of vertices in S that are not the center of such bad stars, that is, vertices in S for which we chose at most one incident blue edge as a representative for a clique. Let $S^* = S \setminus S'$. Since we chose at most one blue edge sv_C for each vertex $s \in S'$, the number of chosen blue edges included in bad stars is at least

$$(4k + 1) - |S'| \geq (4k + 1) - |S| + |S^*| \geq (4k + 1) - 3k + |S^*| = k + 1 + |S^*|.$$

Hence, the weight of the constructed collection of bad stars is at least $k + 1$ and by Lemma 12, we conclude that (G, k) is a no-instance. Thus, if $G \setminus S$ contains at least $4k + 1$ cliques after applying Reduction Rule 1 exhaustively, we can return a trivial no-instance. Otherwise, the number p of cliques is bounded by $4k$.

We are now left with the task of bounding the size of each individual cluster C_i to arrive at a polynomial kernel. To that end, we apply the following marking and deletion procedure to each cluster: For a fixed cluster C_i , begin with an initially empty set M_i . For each vertex $v \in S$, arbitrarily mark $k + 1$ red and $k + 1$ blue neighbors of v in C_i by adding them to M_i (or all red/blue neighbors if there are at most k). Note that we mark at most $|M_i| \leq |S|(2k + 2) \leq 6k^2 + 6k$ vertices this way.

► **Reduction rule 2.** For any cluster C_i , delete all (unmarked) vertices in $C_i \setminus M_i$ from G .

Let \hat{G} be the graph obtained after applying the reduction rule to some cluster C_i . We now need to show that this reduction rule is safe and sound. Let $R_i := C_i \setminus M_i$ be the vertices removed by the reduction rule.

First note that if \mathcal{F} is an overlapping clustering of G , then $\mathcal{F} \setminus R_i$ (interpreted as a multiset⁴, that is, the same cluster might appear multiple times in it) is trivially an overlapping clustering of \hat{G} and $\text{cost}_{\hat{G}}(\mathcal{F} \setminus R_i) \leq \text{cost}_G(\mathcal{F})$. Thus, the reduction rule is safe.

To prove soundness, let $\hat{\mathcal{F}}$ be an overlapping clustering of \hat{G} with $\text{cost}_{\hat{G}}(\hat{\mathcal{F}}) \leq k$. Let $u \in R_i$ be one of the removed vertices. We argue that we can include u in the clustering without increasing the cost. Note that since we removed a vertex, the size of C_i was initially at least $2k + 2$ and hence, by Lemma 13, we have that $\hat{\mathcal{F}}$ contains a set \hat{C} with $C_i \subseteq \hat{C}$. We add u to this cluster and now argue that u does not have to be included in any further clusters if (G, k) is a yes-instance. To that end, we show that every edge incident to u is already covered/resolved by this new clustering.

Let uv be any blue edge incident to u . Note that if $v \in C_i$ then uv is covered by \hat{C} , so we may assume that $v \in S$. Then v has at least $k + 2$ blue neighbors in C_i as otherwise we would have marked u . Let N be a set of $k + 1$ neighbors of v in C_i that were marked. By Lemma 13, there exists a cluster set $X \in \hat{\mathcal{F}}$ with $N \cup \{v\} \subseteq X$. Hence, $X = \hat{C}$ as otherwise the cost of $\hat{\mathcal{F}}$ would be at least $k + 1$ as each vertex in N would appear in at least two sets. Thus, uv is covered by $\hat{C} \cup \{u\}$ in the constructed overlapping clustering.

Now let uv be any red edge incident to u . Again, we claim that because u was unmarked, v must have at least $k + 2$ red neighbors in C_i . Either $v \in S$, in which case the argument is the same as before, or $v \in V \setminus (S \cup C_i)$. In this case, all of C_i is contained in v 's red neighborhood, and we have already observed that C_i has at least $2k + 2$ vertices. Let N be a set of $k + 1$ red neighbors of v in C_i that were marked. Note that v is contained in a set $X \neq \hat{C} \in \hat{\mathcal{F}}$ as otherwise each vertex in N would be contained in at least two sets and $\text{cost}(\hat{\mathcal{F}}) \geq k + 1$. Hence, uv is resolved as $u \in \hat{C} \cup \{u\}$ and $v \in X$.

We conclude that the resulting clustering covers all blue edges incident to u and resolves all red edges incident to u at the same cost as the clustering $\hat{\mathcal{F}}$. By repeating the procedure for the remaining vertices of R_i we conclude that there exists a clustering \mathcal{F} which clusters G and $\text{cost}_G(\mathcal{F}) = \text{cost}_{\hat{G}}(\hat{\mathcal{F}})$. Repeating this argument for every cluster demonstrates that Rule 2 is indeed sound.

Finally, note that after application of Rule 1 and Rule 2 to a yes-instance, we have $p \leq 4k$ clusters of size at most $|S|2(k + 1) \leq 6k^2 + 6k$ each and therefore the total number of vertices in the end is at most $|S| + 4k(6k^2 + 6k) = 24k^3 + 24k^2 + 3k \in O(k^3)$. This concludes the proof. \blacktriangleleft

4.3 Constant-Factor Approximation

We conclude this section with a constant-factor approximation for CLUSTER EDITING WITH PERMISSIVE VERTEX SPLITTING. Again, this is in stark contrast to CORRELATION CLUSTERING WITH PERMISSIVE VERTEX SPLITTING.

► **Theorem 16.** CLUSTER EDITING WITH PERMISSIVE VERTEX SPLITTING *admits a 7-approximation in polynomial time.*

Proof. Let $G = (V, E)$ be a correlation graph. We again begin by computing an inclusion-maximal bad star forest T in G . By Lemma 12, the weight of T is at most opt (the minimum cost of an overlapping clustering of G). Since the number of vertices in a bad star is at most thrice its weight (a bad triangle has weight one and contains three vertices), the set S of

⁴ Technically an overlapping clustering cannot be a multiset. We refer the reader to the proof of Lemma 12, in which we formally show how to adapt this construction into an overlapping clustering with cost bounded by $\text{cost}_G(\mathcal{F})$.

8:14 Correlation Clustering with Vertex Splitting

vertices in T is at most 3 opt . Since T is inclusion-maximal, the graph induced by $V \setminus S$ does not contain any bad star, that is, the blue edges form a cluster graph. Let \mathcal{C} be the set of (blue) cliques in this graph. If $|\mathcal{C}| \leq 1$, then we find a simple 3-approximation by putting each vertex $v \in S$ into its own cluster set X_v and adding one cluster set $X_S = V$. Note that the cost of this overlapping clustering is $|S| \leq 3 \text{ opt}$. Hence, we assume for the remainder of the proof that $|\mathcal{C}| \geq 2$.

Next, we restrict our search to a solution that contains one cluster set X_S with $S \subseteq X_S$ and for each vertex $v \in S$ one cluster set $X_v = \{v\}$. Note that we can add these sets to any solution (if they are not already present within the solution) to get a new solution whose cost is at most $2|S| \leq 6 \text{ opt}$ larger than the original. We call overlapping clusterings that satisfy the above *simple solutions* and we denote the minimum cost of a simple solution by opt' .

We next show that there is always a simple solution of cost opt' that contains for each clique $C \in \mathcal{C}$ a cluster set X_C with $C \subseteq X_C$. Start with any simple solution \mathcal{F} of cost opt' and any clique $C \in \mathcal{C}$ and assume that \mathcal{F} does not contain a cluster set containing C . We prove that in this case each vertex in C is contained in at least two cluster sets in \mathcal{F} . Assume towards a contradiction that some vertex $v \in C$ is contained in exactly one cluster set Y (note that by definition of overlapping clusterings, each vertex is contained in at least one cluster set). Since we assumed that no cluster set completely contains C , there exists a vertex $u \in C \setminus Y$. However, since u and v are contained in the same clique C , the edge between them is blue and has to be covered by some cluster set $Z \in \mathcal{F}$ (and hence Z has to contain both u and v). Note that $Z \neq Y$ since $u \in Z$ but $u \notin Y$. This contradicts the assumption that v is only contained in cluster set Y .

We construct a new simple solution \mathcal{F}' of cost opt' by removing all vertices in C from all cluster sets in \mathcal{F} and adding them all to X_S . In addition, we add one new cluster set $X_C = C$. Note that the cost of \mathcal{F}' is at most the cost of \mathcal{F} as we removed each vertex in C from at least two cluster sets and added them to exactly two cluster sets. Moreover, the new solution is indeed an overlapping clustering as all blue edges incident to a vertex in C are covered by X_S as all blue neighbors are either in S or in C . Since no red neighbors of any vertex in C are contained in C , the new cluster set X_C ensures that all red edges incident to vertices in C are resolved. Repeating the above for all cliques in \mathcal{C} yields a simple solution of cost opt' that contains for each clique $C \in \mathcal{C}$ a cluster set X_C with $C \subseteq X_C$.

The next step is to show that there is always an optimal simple solution (a simple solution of cost opt') in which $X_C \neq X_{C'}$ for any pair $C \neq C' \in \mathcal{C}$. Start with any optimal simple solution \mathcal{F} that contains a cluster set $X_C \supseteq C$ for each clique $C \in \mathcal{C}$ and assume that $X_{C_1} = X_{C_2}$ for some cliques $C_1 \neq C_2$. Observe that all vertices from at least one of the two cliques are contained in at least two cluster sets each as if there are vertices $u \in C_1$ and $v \in C_2$ that are only contained in $X_{C_1} = X_{C_2}$, then the red edge between them is not resolved by \mathcal{F} . Without loss of generality, let all vertices of C_1 be contained in at least two cluster sets each. Then, we construct a new simple solution \mathcal{F}' of cost opt' by removing all vertices in C_1 from all cluster sets in \mathcal{F} and adding them all to X_S and adding one new cluster set $X_{C_1} = C_1$. The proof that this is correct is exactly the same as before. The cost of \mathcal{F}' is at most the cost of \mathcal{F} as we removed each vertex in C_1 from at least two cluster sets and added them to exactly two cluster sets. Moreover, the new solution is indeed an overlapping clustering as all blue edges incident to a vertex in C_1 are covered by X_S and the new cluster set X_{C_1} ensures that all red edges incident to vertices in C are resolved. Repeating the above for all cliques in \mathcal{C} yields an optimal simple solution that contains for each clique $C \in \mathcal{C}$ a cluster set $X_C \supseteq C$ such that $X_C \neq X_{C'}$ for all $C \neq C' \in \mathcal{C}$.

Next, we guess which clique $C^* \in \mathcal{C}$ satisfies $X_{C^*} = X_S$ in an optimal simple solution satisfying all of the above.⁵ By that, we mean that we try all possibilities of the following and return the best solution found. For each clique $C \in \mathcal{C} \setminus \{C^*\}$, we compute a minimum vertex cover K_C of the blue edges between C and S in $O(n^3)$ time using König's theorem (note that the considered graph is bipartite by construction). We add each vertex in $K_C \cap S$ to X_C and each vertex in $K_C \cap C$ to X_S .

We claim that $\sum_{C \in \mathcal{C} \setminus \{C^*\}} |K_C| \leq \text{opt}' - |S|$. By the above arguments, we can start with an overlapping clustering \mathcal{F} consisting of one cluster set $X_S = S \cup C^*$, one cluster set $X_C = C$ for each $C \in \mathcal{C} \setminus \{C^*\}$, and one cluster set $X_v = \{v\}$ for each $v \in S$. Note that all red edges are resolved and all blue edges except for those between S and $V \setminus (S \cup C^*)$ are covered. To cover a blue edge uv with $u \in S$ and $v \in C$ for some $C \in \mathcal{C} \setminus \{C^*\}$, there are three possibilities: We can add u to X_C , we can add v to X_S , or there exists a different cluster set $Y \in \mathcal{F}$ with $\{u, v\} \subseteq Y$. Note that in the third case, we can remove v from Y and add it to X_S and still get an optimal simple solution. Moreover, the optimal way to cover all blue edges between S and $V \setminus (S \cup C^*)$ using the first two possibilities corresponds exactly to $\sum_{C \in \mathcal{C} \setminus \{C^*\}} |K_C|$. Since now all red edges are resolved and all blue edges are covered and the cost of the constructed overlapping clustering is

$$|S| + \sum_{C \in \mathcal{C} \setminus \{C^*\}} |K_C| \leq \text{opt}' \leq 7 \text{opt},$$

we successfully computed a factor-7 approximation in polynomial time. ◀

We leave it as an open problem to improve the approximation factor. We conjecture that a refined concept of a simple solution might yield an approximation factor of 4.

5 Conclusion

We have introduced permissive vertex splitting, which generalizes the earlier exclusive and inclusive vertex splitting notions by allowing symmetry with respect to “positive” and “negative” pairwise similarity data. Our type of vertex splitting turns out to be quite satisfying, as it corresponds to a natural definition of overlapping clustering. Unfortunately, the general case of CORRELATION CLUSTERING WITH PERMISSIVE VERTEX SPLITTING is rather intractable, as it is para-NP-hard and admits no $n^{1-\varepsilon}$ approximation in polynomial time for any $\varepsilon > 0$ (unless $P = NP$). On the positive side, when restricted to datasets with complete data we obtain a kernel with $O(k^3)$ vertices and a polynomial time 7-approximation. Interesting questions remain, for example whether one can reduce our approximation factor to 4 (or even lower), whether a kernel with only linearly many vertices exists (as is the case when only inclusive splits and edge-edits are allowed [1]), or whether refined lower bounds in terms of running time or approximation factor can be found. Future work might also consider the parameterized complexity of CORRELATION CLUSTERING WITH PERMISSIVE VERTEX SPLITTING and CLUSTER EDITING WITH PERMISSIVE VERTEX SPLITTING with respect to structural parameters of the input, or with restrictions on the number of clusters which contain any given node.

Finally, we would like to amplify the call of Abu-Khzam et al. [1] to extend the study of vertex splitting (exclusive, inclusive, or permissive) to other classes of target graphs, many of which (e.g., bicluster graphs, s -cliques, s -clubs, s -plexes, k -cores, and γ -quasi-cliques) have been proposed as alternatives to cliques in clustering applications.

⁵ We can assume that one such clique always exists, but even if this was not the case, the following proof still works if the guess $\{C^*\} = \emptyset$ yields an optimal simple solution.

References

- 1 Faisal N. Abu-Khzam, Emmanuel Arrighi, Matthias Bentert, Pål Grønås Drange, Judith Egan, Serge Gaspers, Alexis Shaw, Peter Shaw, Blair D. Sullivan, and Petra Wolf. Cluster editing with vertex splitting. *arXiv preprint*, 2023. [arXiv:1901.00156](https://arxiv.org/abs/1901.00156).
- 2 Faisal N. Abu-Khzam, Joseph R. Barr, Amin Fakhhereldine, and Peter Shaw. A greedy heuristic for cluster editing with vertex splitting. In *Proceedings of the 4th International Conference on Artificial Intelligence for Industries (AI4I)*, pages 38–41. IEEE, 2021.
- 3 Faisal N. Abu-Khzam, Judith Egan, Serge Gaspers, Alexis Shaw, and Peter Shaw. Cluster editing with vertex splitting. In *Proceedings of the 5th International Symposium on Combinatorial Optimization (ISCO)*, pages 1–13. Springer, 2018.
- 4 Sanjeev Arora, Rong Ge, Sushant Sachdeva, and Grant Schoenebeck. Finding overlapping communities in social networks: Toward a rigorous approach. In *Proceedings of the 13th ACM Conference on Electronic Commerce (EC)*, pages 37–54. Association for Computing Machinery, 2012.
- 5 Emmanuel Arrighi, Matthias Bentert, Pål Grønås Drange, Blair D. Sullivan, and Petra Wolf. Cluster editing with overlapping communities. In *Proceedings of the 18th International Symposium on Parameterized and Exact Computation (IPEC)*, pages 2:1–2:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023.
- 6 Gard Askeland. Overlapping community detection using cluster editing with vertex splitting. Master’s thesis, University of Bergen, Bergen, Norway, 2022.
- 7 Sanghamitra Bandyopadhyay, Garisha Chowdhary, and Debarka Sengupta. FOCS: Fast overlapped community search. *IEEE Transactions on Knowledge and Data Engineering*, 27(11):2974–2985, 2015.
- 8 Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine Learning*, 56(1-3):89–113, 2004.
- 9 Jeffrey Baumes, Mark Goldberg, and Malik Magdon-Ismael. Efficient identification of overlapping communities. In *Proceedings of the 2005 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 27–36. Springer, 2005.
- 10 Francesco Bonchi, Aristides Gionis, and Antti Ukkonen. Overlapping correlation clustering. *Knowledge and Information Systems*, 35(1):1–32, 2013.
- 11 Sebastian Böcker. A golden ratio parameterized algorithm for cluster editing. *Journal of Discrete Algorithms*, 16:79–89, 2012.
- 12 Leizhen Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58(4):171–176, 1996.
- 13 Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. Clustering with qualitative information. *Journal of Computer and System Sciences*, 71(3):360–383, 2005.
- 14 Moses Charikar and Anthony Wirth. Maximizing quadratic programs: Extending Grothendieck’s inequality. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 54–60. IEEE, 2004.
- 15 Shuchi Chawla, Robert Krauthgamer, Ravi Kumar, Yuval Rabani, and D. Sivakumar. On the hardness of approximating multicut and sparsest-cut. *Computational Complexity*, 15(2):94–114, 2006.
- 16 Jianer Chen and Jie Meng. A 2k kernel for the cluster editing problem. *Journal of Computer and System Sciences*, 78(1):211–220, 2012.
- 17 Vincent Cohen-Addad, Euiwoong Lee, Shi Li, and Alantha Newman. Handling correlated rounding error via preclustering: A 1.73-approximation for correlation clustering. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1082–1104. IEEE, 2023.
- 18 Alex Crane, Brian Lavalley, Blair D. Sullivan, and Nate Veldt. Overlapping and robust edge-colored clustering in hypergraphs. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 143–151, 2024.

- 19 George B. Davis and Kathleen M. Carley. Clearing the FOG: Fuzzy, overlapping groups for social networks. *Social Networks*, 30(3):201–212, 2008.
- 20 Erik D. Demaine, Dotan Emanuel, Amos Fiat, and Nicole Immorlica. Correlation clustering in general weighted graphs. *Theoretical Computer Science*, 361(2-3):172–187, 2006.
- 21 Reinhard Diestel. *Graph Theory*. Springer, 2012.
- 22 Nan Du, Bai Wang, Bin Wu, and Yi Wang. Overlapping community detection in bipartite networks. In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, pages 176–179, 2008.
- 23 Esther Galbrun, Aristides Gionis, and Nikolaj Tatti. Overlapping community detection in labeled graphs. *Data Mining and Knowledge Discovery*, 28(5-6):1586–1610, 2014.
- 24 Reynaldo Gil-García and Aurora Pons-Porrata. Dynamic hierarchical algorithms for document clustering. *Pattern Recognition Letters*, 31(6):469–477, 2010.
- 25 Mark Goldberg, Stephen Kelley, Malik Magdon-Ismael, Konstantin Mertsalov, and Al Wallace. Finding overlapping communities in social networks. In *Proceedings of the 2nd IEEE International Conference on Social Computing (SC)*, pages 104–113, 2010.
- 26 Steve Gregory. An algorithm to find overlapping community structure in networks. In *Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 91–102. Springer, 2007.
- 27 Qinna Wang and Eric Fleury. Uncovering overlapping community structure. In *Proceedings of the 2nd International Workshop on Complex Networks (COMPLEX NETWORKS)*, pages 176–186. Springer, 2010.
- 28 Xufei Wang, Lei Tang, Huiji Gao, and Huan Liu. Discovering overlapping groups in social media. In *Proceedings of the 2010 IEEE International Conference on Data Mining (ICDM)*, pages 569–578, 2010.
- 29 David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(1):103–128, 2007.