Editing Graphs into Few Cliques: Complexity, Approximation, and Kernelization Schemes

Falk Hüffner*, Christian Komusiewicz, and André Nichterlein

Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Germany
{falk.hueffner,christian.komusiewicz,andre.nichterlein}@tu-berlin.de

Abstract. Given an undirected graph G and a positive integer k, the NP-hard Sparse Split Graph Editing problem asks to transform G into a graph that consists of a clique plus isolated vertices by performing at most k edge insertions and deletions; similarly, the P_3-Bag Editing problem asks to transform G into a graph which is the union of two possibly overlapping cliques. We give a simple linear-time 3-approximation algorithm for Sparse Split Graph Editing, an improvement over a more involved known factor-3.525 approximation. Further, we show that P_3-Bag Editing is NP-complete. Finally, we present a kernelization scheme for both problems and additionally for the 2-Cluster Editing problem. This scheme produces for each fixed ε in polynomial time a kernel of order εk. This is, to the best of our knowledge, the first example of a kernelization scheme that converges to a known lower bound.

1 Introduction

The study of graph modification problems is a classic topic in theoretical computer science. The typical task in this context is, given a graph class Π and a graph G, to modify G by a minimum number of operations such that the resulting graph is contained in Π. By a general result, graph modification is NP-hard if the operation is vertex deletion and Π is hereditary [19]. In contrast, for *edge* modification problems where one may insert or delete edges, no such general hardness result is possible. One nontrivially tractable example is the case when Π is the class of split graphs, that is, graphs whose vertex set can be partitioned into a clique and an independent set (edges between the independent set and the clique are allowed). The problem of modifying a graph into a split graph by a minimum number of edge modifications (insertions or deletions) is polynomial-time solvable [15]. This result relies on the fact that a split graph can be recognized by its degree sequence. In contrast, Natanzon et al. [20] showed that the problem becomes NP-hard when allowing either only edge deletions or only edge insertions.

Damaschke and Mogren [6, 7] considered several graph modification problems for very restricted graph classes where, informally, the number of different neighborhoods is constant. In this paper, we study two problems of this kind.

* Supported by DFG project ALEPH (HU 2139/1).

First, we consider a very restricted subclass of split graphs where no edges between the independent set and the clique are allowed. More specifically, we call a graph G a *sparse split graph* if G consists of a clique and isolated vertices. The corresponding graph modification problem is defined as follows.

Sparse Split Graph Editing
Input: A graph $G = (V, E)$ and an integer $k \in \mathbb{N}$.
Question: Can G be transformed into a sparse split graph by at most k edge insertions and deletions?

Sparse Split Graph Editing was studied by Damaschke and Mogren [6] under the names $K_1[0]$-Bag Editing and Clique Editing. For example, it was shown that Sparse Split Graph Editing can be solved in $2^{O(\sqrt{k}\log k)} \cdot n^{O(1)}$ time whereas the NP-hardness of Sparse Split Graph Editing was initially left open [6]; it was later shown to be NP-hard by Kovác et al. [18]. We also consider the following further problem, as introduced by Damaschke and Mogren [6].

P_3-Bag Editing
Input: A graph $G = (V, E)$ and an integer $k \in \mathbb{N}$.
Question: Can G be transformed into two possibly overlapping cliques by at most k edge insertions and deletions?

We call such graphs P_3-*bag graphs*. The term refers to the fact that in such a graph merging all vertices with the same closed neighborhood results in a P_3 or an induced subgraph of a P_3. An equivalent definition is as follows: the graph class is the set of all graphs with edge clique cover number at most two.

Further related work. To obtain a sparse split graph by a minimum number of edge insertions is trivially solvable in polynomial time. If one allows only edge deletions, the problem is NP-hard [7]. Sparse Split Graph Editing has applications in the identification of core–periphery structures in social networks [4]. Other models considered in this context include Split Editing and Dense Split Graph Editing which asks to transform the input graph into a dense split graph, that is, a graph which consists of a clique and an independent set and in which all edges are present between the clique and the independent set [4].

Many graph classes defined by existence of a certain vertex partitioning can be captured with the notion of a *pattern* [16]. A pattern for a partition into d parts is a symmetric $d \times d$ matrix M with entries from $\{0, 1, *\}$. Then, an M-*partition* of a graph $G = (V, E)$ is a partition $V_1, \ldots, V_d$ of V such that two distinct vertices in (possibly equal) parts V_i and V_j are adjacent if $M(i, j) = 1$ and nonadjacent if $M(i, j) = 0$ (the entry $M(i, j) = *$ signifies no restriction). Thus, sparse split graphs are the graphs with a $\left(\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right)$-partition and P_3-bag graphs are the graphs with a $\left(\begin{smallmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{smallmatrix}\right)$-partition. Expressed with these definitions, Damaschke and Mogren [7] consider editing problems for the case where the diagonal is 1 and off-diagonal elements are 0 or 1.

2-Cluster Editing (also known as 2-Correlation Clustering on complete graphs) is to find a minimum number of edge modifications to convert a

graph into two disjoint cliques (that is, into a graph with a $\left(\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right)$-partition). It is NP-hard [21], but has a kernel with at most $4k+2$ vertices [14]. It can be solved in subexponential time $2^{O(\sqrt{k})}+n^{O(1)}$ [11]; a subexponential running time follows also from the more general result of Damaschke and Mogren [7]. Wu and Chen [22] give a different subexponential algorithm.

Our results. First, we complement and improve on results for Sparse Split Graph Editing and P_3-Bag Editing. In particular, we show a factor-3 approximation for Sparse Split Graph Editing in Section 2, and prove NP-hardness of P_3-Bag Editing in Section 3. The former result improves a factor-3.524 approximation from Kovác et al. [18] and the latter answers an open question of Damaschke and Mogren [6].

Second, we provide kernelization schemes for Sparse Split Graph Editing, P_3-Bag Editing, and 2-Cluster Editing in Section 4. Analogous to a polynomial-time approximation scheme (PTAS), a kernelization scheme provides increasingly good bounds on the kernel size, at the cost of an increasing running time bound. Only few kernelization schemes are known (e. g. [1, 3, 10]), and they provide kernel size bounds of the form $(1+\varepsilon)k$, where the limit bound k is not known to be sharp (unlike for a PTAS). Abu-Khzam and Fernau [1] ask whether there are kernelization schemes that converge to a provable lower bound. We answer this question positively by providing, for the three above-mentioned problems, such schemes where the size bound converges, in fact, to 0. We formalize this by introducing the notion of strict kernelization schemes.

Definition 1. *A* strict kernelization scheme *is an algorithm $\mathcal{A}$ which takes as input an instance (I,k) of a parameterized problem and a constant $\varepsilon>0$ and produces in $(|I|+k)^{f(1/\varepsilon)}$ time an instance (I',k') such that $(I,k)\in L \iff (I',k')\in L$, $|I'|\leq\varepsilon\cdot g(k)$, and $k'\leq k$ for some functions f and g.*

Note that, by first kernelizing with $\varepsilon=1$ and then in a second step kernelizing the resulting instance with the intended value of ε, the running time of a strict kernelization scheme can always be improved to $g(k)^{f(1/\varepsilon)}+|I|^{O(1)}$.

Preliminaries. For a graph $G=(V,E)$ we set $n:=|V|$ and $m:=|E|$. The *open neighborhood* of a vertex u is $N_G(u):=\{v\mid\{u,v\}\in E\}$. The *closed neighborhood* of a vertex u is $N_G[u]:=\{u\}\cup N_G(u)$. For a vertex subset $V'\subseteq V$, the *subgraph induced by V'* is denoted by $G[V']$. For two disjoint vertex subsets $V_1,V_2\subseteq V$, the set of edges with one endpoint in V_1 and one endpoint in V_2 is denoted by $E_G(V_1,V_2)$. We omit the subscript if the graph G is clear from the context. A clique on $k\in\mathbb{N}$ vertices is denoted by K_k, and a complete bipartite graph with $k_1\in\mathbb{N}$ vertices in one part and $k_2\in\mathbb{N}$ vertices in the other part is denoted by K_{k_1,k_2}. The "$\triangle$" operator denotes the symmetric difference with $A\triangle B:=(A\cup B)\setminus(A\cap B)$.

For the relevant notions of parameterized complexity, such as kernelization, we refer to the monograph by Downey and Fellows [8]. Due to space constraints, several proofs are deferred to a full version.

2 Sparse Split Graph Editing

We first make several simple observations on the structure of sparse split graphs and on dense split graphs. These observations can be useful in applications of Sparse Split Graph Editing.

Characterizations. The class of sparse split graphs is hereditary, that is, it is closed under vertex deletions. Hence, sparse split graphs can be characterized by a set of forbidden induced subgraphs. In general, such characterizations can be useful for example for obtaining recognition algorithms for a graph class Π or for obtaining fixed-parameter algorithms for hard graph modifications problems for Π [5]. For sparse split graphs, the following simple characterization is known.

Theorem 1 ([23, Theorem 5.2.7]). *A graph G is a sparse split graph if and only if it does not contain a $2K_2$ or a P_3 as an induced subgraph.*

Like split graphs, sparse split graphs can be characterized by their degree sequence, that is, the list of degrees of their vertices sorted in descending order.

Theorem 2. *A graph is a sparse split graph if and only if its degree sequence is $\underbrace{c, c, \ldots, c}_{c+1}, \underbrace{0, 0, \ldots, 0}_{n-c-1}$ for some $c \geq 1$.*

Sparse split graphs are closely related to dense split graphs.

Lemma 1. *A graph G is a dense split graph if and only if its complement is a sparse split graph.*

By building the complement of the forbidden induced subgraphs for sparse split graphs, we can thus obtain the following forbidden subgraph characterization for dense split graphs.

Corollary 1. *A graph G is a dense split graph if and only if it does not contain a C_4 or a $K_2 + K_1$ as an induced subgraph.*

Similarly, we obtain the following corollary to Theorem 2.

Corollary 2. *A graph is a dense split graph if and only if its degree sequence is $\underbrace{n-1, n-1, \ldots, n-1}_{c}, \underbrace{c, c, \ldots, c}_{n-c}$ for some $c \geq 1$.*

Approximation. Kovác et al. [18] present an approximation algorithm for Sparse Split Graph Editing and prove an approximation factor of 3.524; they conjecture that the algorithm is a 3.383-approximation. We give a simpler 3-approximation, inspired by the polynomial-time algorithm for Split Graph Editing [15] that is based on a characterization by the degree sequence. This algorithm sorts the vertices by degree, and chooses the vertices up to a certain point in the sequence for the independent set and the remaining ones for the clique of the resulting split graph. Since sparse split graphs have a similar characterization by degree sequence (Theorem 2), we use the same algorithm to get

an approximation for Sparse Split Graph Editing. For separating the clique from the independent set, however, we do not calculate the threshold but try all of them. More precisely, for each $0 \leq x \leq n$, choose the x vertices with the highest degree as clique (resolving ties arbitrarily), and retain the best of these $n+1$ solutions. Here, "choosing as clique" means to add all missing edges within the vertex set and delete all other edges, yielding a sparse split graph.

Theorem 3. Sparse Split Graph Editing *can be approximated in linear time within a factor of* 3.

Proof. A linear running time can be achieved by processing the vertices in order of decreasing degree, where the clique would be formed by all vertices processed so far. We maintain m_c, the number of edges within the clique; updating m_c can be done in $O(m)$ time total. The number of modifications for clique size c can then be calculated as $(\binom{c}{2} - m_c) + (m - m_c)$.

We now analyze the approximation factor. The analysis is based on the proof of Hammer and Simeone [15] showing that Split Graph Editing is polynomial time solvable. Let C_{opt} be the clique of an optimal solution S_{opt} of cost k_{opt} and C the clique of the solution S calculated by the approximation algorithm for $c = |C_{\text{opt}}|$, with cost k. For $S_1, S_2 \subseteq V$, we denote by $E(S_1)$ the edges that have both endpoints in $S_1 \subseteq V$ and by $E(S_1, S_2)$ the edges with one endpoint in S_1 and the other endpoint in S_2. With this, the cost k_{opt} can be decomposed:

$$k_{\text{opt}} = \underbrace{\frac{c(c-1)}{2} - |E(C_{\text{opt}})|}_{\text{edges added in } C_{\text{opt}}} + \underbrace{|E(V \setminus C_{\text{opt}})| + |E(C_{\text{opt}}, V \setminus C_{\text{opt}})|}_{\text{deleted edges with endpoint(s) in } V \setminus C_{\text{opt}}}. \tag{1}$$

Observe that for any set $S \subseteq V$ it holds that:

$$\sum_{v \in S} \deg(v) = 2|E(S)| + |E(S, V \setminus S)|, \tag{2}$$

where $\deg(v)$ denotes the degree of v. Rearranging Equality (2) to have $|E(S)|$ on the left-hand side and inserting the right-hand side in Equality (1) for $|E(C_{\text{opt}})|$ and $|E(V \setminus C_{\text{opt}})|$ yields:

$$k_{\text{opt}} = \frac{1}{2}\left(c(c-1) - \sum_{v \in C_{\text{opt}}} \deg(v) + \sum_{v \in V \setminus C_{\text{opt}}} \deg(v)\right) + |E(C_{\text{opt}}, V \setminus C_{\text{opt}})|. \tag{3}$$

Let $d_1 \geq d_2 \geq \ldots \geq d_n$ be the degrees of the vertices in descending order. It follows that:

$$\sum_{v \in C_{\text{opt}}} \deg(v) \leq \sum_{i=1}^{c} d_i \qquad \text{and} \qquad \sum_{v \in V \setminus C_{\text{opt}}} \deg(v) \geq \sum_{i=c+1}^{n} d_i. \tag{4}$$

Inserting this into Equality (3) yields:

$$k_{\text{opt}} \geq \frac{1}{2}\left(c(c-1) - \sum_{i=1}^{c} d_i + \sum_{i=c+1}^{n} d_i\right) + |E(C_{\text{opt}}, V \setminus C_{\text{opt}})| \tag{5}$$

Observe that if C_{opt} contains the vertices with the highest degree in G, then Inequality (5) becomes an equality. Furthermore, our approximation algorithm for $x = c$ actually contains the c vertices with highest degree in C. Thus, using the same analysis as above for k and C instead of k_{opt} and C_{opt}, we obtain

$$k = \frac{1}{2}\left(c(c-1) - \sum_{i=1}^{c} d_i + \sum_{i=c+1}^{n} d_i\right) + |E(C, V \setminus C)|. \tag{6}$$

It remains to bound the size of $E(C, V \setminus C)$. To this end, observe that

$$\begin{aligned} |E(C, V \setminus C)| &\leq \sum_{v \in V \setminus C} \deg(v) = \sum_{i=c+1}^{n} d_i \overset{(4)}{\leq} \sum_{v \in V \setminus C_{\text{opt}}} \deg(v) \\ &\overset{(2)}{\leq} 2|E(V \setminus C_{\text{opt}})| + 2|E(V \setminus C_{\text{opt}}, C_{\text{opt}})| \overset{(1)}{\leq} 2k_{\text{opt}} \end{aligned}$$

Putting this together yields $k \leq 3k_{\text{opt}}$. □

Using a computer program, we determined the worst-case approximation factor (i. e., with unlucky tie resolving) for all graphs up to 11 vertices. The worst case is a factor of 2.5, and only one graph with this factor was found (up to adding singletons): a disjoint union of a triangle and a P_3.

3 P_3-Bag Editing

We now turn to P_3-Bag Editing. Recall that a P_3-bag graph is a graph that consists of exactly two possibly overlapping cliques.

Characterizations. We first give a forbidden subgraph characterization of P_3-bag graphs. Note that P_3-bag graphs cannot be characterized by their degree sequence: Two disjoint triangles and a cycle on six vertices have both the degree sequence $2, 2, 2, 2, 2, 2$. However, only the former is a P_3-bag graph.

Theorem 4. *A graph G is a P_3-bag graph if and only if it does not contain a $3K_1$, P_4, or C_4 as an induced subgraph.*

Proof. It is easy to see that P_4 and C_4 are not P_3-bag graphs. From a more general result on forbidden subgraphs for graphs with certain M-partitions [9, Corollary 3.3], it follows that a minimal forbidden subgraph for P_3-bag graphs can have at most four vertices, and there can be at most two minimal forbidden subgraphs with four vertices. Finally, it is easy to verify that all graphs with three or fewer vertices except for $3K_1$ are P_3-bag graphs. □

For P_3-Bag Editing, we can also consider the complement problem. The following characterizations follow from our characterizations of P_3-bag graphs.

Lemma 2. *For a graph G, the following are equivalent.*

1. *G is a complement of a P_3-bag graph.*
2. *G consists of a complete bipartite graph (*biclique*) plus isolated vertices.*
3. *G does not contain a K_3, P_4, or $2K_2$ as an induced subgraph.*

Thus, while Sparse Split Graph Editing is the problem of editing a graph into a clique plus isolated vertices, P_3-Bag Editing is the problem of editing the complement of a graph into a biclique plus isolated vertices. Note that the problem of editing a graph into a biclique (without isolated vertices) is the complement problem of 2-Cluster Editing.

NP-hardness. We now show that P_3-Bag Editing is NP-complete. This demonstrates the value of the subexponential fixed-parameter algorithm that solves P_3-Bag Editing in $O(2^{\sqrt{k}\log k})$ time [7].

Theorem 5. *P_3-Bag Editing is NP-complete.*

Proof (sketch). Containment in NP is obvious. To prove NP-hardness, we provide a polynomial-time reduction from the Bisection problem, which was shown to be NP-hard by Garey et al. [12].

Bisection
Input: A graph $G = (V, E)$ and an integer $k \in \mathbb{N}$.
Question: Does G have a bisection with cut size at most k, that is, a partition of V into two sets V_1 and V_2 such that $|V_1| = |V_2|$ and $|E(V_1, V_2)| \leq k$?

Given a Bisection instance $(G = (V, E), k)$ with $m > k$ we construct an equivalent P_3-Bag Editing instance $(G' = (V', E'), k')$ as follows. First, copy G into G'. Next, add for each vertex $v \in V$ a clique with n^2 vertices to G' and make all vertices in this clique adjacent to v in G'. Denote the vertices in this clique by $C(v)$ (with $v \notin C(v)$). We call these cliques *pendant cliques* to distinguish them from the at most two maximal cliques in the P_3-bag graph. We first explain the intuition behind the construction. The pendant cliques are pairwise non-adjacent. This forces a balanced "distribution" of the pendant cliques to the two maximal cliques of the P_3-bag graph as any non-balanced distribution exceeds the budget (which we define below). Then the balanced distribution of the pendant cliques forces the original vertex set V to be also split into two equal size sets. Hence, choosing the budget k' appropriately ensures a cut size of at most k between these two sets. To define k', we use $t := n/2$ to denote the size of the two parts in a bisection of G and set

$$k' := \underbrace{n^4 \cdot 2\binom{t}{2}}_{\substack{\text{edges added}\\ \text{between cliques}}} + \underbrace{n^2 \cdot n(t-1)}_{\substack{\text{edges added}\\ \text{between cliques and}\\ \text{original vertices}}} + \underbrace{k}_{\substack{\text{edges removed}\\ \text{in cut of}\\ \text{bisection}}} + \underbrace{2\binom{t}{2} - (m-k)}_{\substack{\text{edges added between}\\ \text{original vertices inside the}\\ \text{two parts of the partition}}}.$$

It now holds that (G, k) is a yes-instance of Bisection $\iff$ (G', k') is a yes-instance of P_3-Bag Editing; we omit the proof. □

In the proof above, the intersection of the maximal cliques in the optimal solution for the constructed instance is empty. Thus, the reduction also provides an alternative NP-hardness proof for 2-Cluster Editing.

4 Kernelization Schemes

We now give strict kernelization schemes (see Definition 1 in Section 1) for 2-Cluster Editing, Sparse Split Graph Editing, and P_3-Bag Editing. Since complementing the graph does not affect k, they also apply to Biclique Editing, Dense Split Graph Editing, and Biclique+Singletons Editing.

The idea of all three schemes is to apply data reduction that ensures that the number of edge modifications incident on each vertex is at least some constant c. Then, if we can solve the instance with k modifications, the number of vertices remaining is at most $2k/c$, and by setting $c := 2/\varepsilon$, we can achieve any kernel of order εk. The critical property that allows the data reduction is that from knowing the neighborhood of just one vertex in an optimal solution, we can easily construct a complete optimal solution graph. Here, for simplicity, we use *solution* to refer either to the set of editing operations or to the graph from the target class of the editing problem that is obtained by applying the editing operations.

We formulate the data reduction for any graph modification problem that is "neighborhood-reconstructible" and "allows isolation", and prove that our problems have these properties. For convenience, instead of P_3-Bag Editing we consider the complement problem Biclique+Singletons Editing, that is, the problem of editing into a biclique plus isolated vertices.

Definition 2. *A graph modification problem is* neighborhood-reconstructible *in $p(n)$ time for some polynomial p when given the nonempty neighborhood of a vertex in a solution G', one can in $p(n)$ time either find a solution with at most k edge modifications or determine that the solution G' incurs more than k edge modifications. This method is called* neighborhood reconstruction.

Observe that we demand the reconstructibility only for nonempty neighborhoods. This is done to cope with vertices that can become singletons in the solution.

Lemma 3. *2-Cluster Editing, Sparse Split Graph Editing, and Biclique+Singletons Editing are neighborhood-reconstructible in linear time.*

Proof. For 2-Cluster Editing, neighborhood reconstruction is possible even given an empty neighborhood of a vertex. Assume we know the neighborhood $N(u)$ of any vertex u in a solution. Then we can reconstruct the solution in linear time: one clique is $C_1 := N[u]$ and the other is $C_2 := V \setminus N[u]$. We can in linear time determine $m_{1,2}$, the number of edges between C_1 and C_2. Then the number of modifications k can be calculated as $\binom{|C_1|}{2} + \binom{|C_2|}{2} - m + 2m_{1,2}$.

For Sparse Split Graph Editing, let C and I be the clique and the isolated vertices of a solution, respectively. Only vertices in C have nonempty neighborhoods in a solution. Assume we know the neighborhood $N(u)$ of a

vertex $u \in C$ in a solution. Then $C = N[u]$ and $I = V \setminus N[u]$. We can count in linear time the number m_C of edges within C, and the number of edge modifications is $m + \binom{|C|}{2} - 2m_C$.

For Biclique+Singletons Editing, let B_1 and B_2 be the two parts of the biclique, and I the isolated vertices. Assume that for a vertex $u \in B_1 \cup B_2$ (without loss of generality $u \in B_1$), we know the neighborhood $N(u)$ of u in a solution. Then we have $B_2 = N(u)$ and $B_1 \cup I = V \setminus N(u)$. It remains to allocate the vertices in $B_1 \cup I$ to B_1 or I. Each decision for a vertex $v \in B_1 \cup I$ can be made independently: if there are at least $|B_2|/2$ edges from v to B_2, we place v in B_1, and otherwise we place it in I. Since each edge will be considered at most once, this can be done in $O(m)$ time. We can then count in linear time the number m_B of edges between B_1 and B_2, and the number of edge modifications is $m + |B_1||B_2| - 2m_B$. □

The first rule directly exploits neighborhood reconstructibility: Assume that there is a vertex with nonempty neighborhood in the solution and that the difference between the input neighborhood and solution neighborhood is small. Then, we can find an optimal solution by guessing this small difference and then using neighborhood reconstruction. When this fails for all vertices, we know that each vertex has many incident edge modifications or is isolated in the solution.

Rule 1. Consider a constant c and a graph modification problem that is neighborhood-reconstructible in $p(n)$ time. For each vertex u, try all ways of changing up to $c-1$ incidences with the other vertices, that is, consider the neighborhoods $\{N(u) \triangle T \mid T \subseteq V \setminus \{u\}, |T| \leq c-1\}$. If for some u and some T, neighborhood reconstruction finds a solution with at most k edge modifications, then replace the instance by a trivial "yes"-instance.

Lemma 4. *Rule 1 is sound and can be executed in $O(n^c \cdot p(n))$ time.*

Proof. It is clear that the rule is sound, that is, it produces a "yes"-instance if and only if the original instance is a "yes"-instance. The running time can be seen as follows. There are n vertices and $O(n^{c-1})$ vertex sets to try, and each choice can be checked in $p(n)$ time. □

Observation 1. Exhaustively applying Rule 1 yields an instance in which it holds for every solution with at most k edge modifications that each vertex is incident with at least c edge modifications or isolated in the solution.

For 2-Cluster Editing, at most two vertices have empty neighborhood in the solution. Hence, Rule 1 is already sufficient to bound the number of incident edge modifications for all except two vertices. This is not sufficient for Sparse Split Graph Editing and Biclique+Singletons Editing where a solution may contain many singletons. Here, we exploit another problem property.

Definition 3. *A graph modification problem* allows isolation *if the property of being a solution is hereditary, and adding an isolated vertex to a solution yields another solution.*

Thus, any solution can be transformed into a new one by picking an arbitrary vertex and removing all incident edges, hence the name. Observe that Sparse Split Graph Editing and Biclique+Singletons Editing allow isolation.

Rule 2. For a graph modification problem that allows isolation, assume it is known that in every solution, each vertex has at least c incident edge modifications or degree 0 in the solution (or both). If G contains a vertex u with $\deg(u) \leq c$, then remove u from G and reduce k by $\deg(u)$.

Lemma 5. *Rule 2 is sound and can be performed exhaustively in $O(nm)$ time.*

Observation 2. Exhaustively applying Rules 1 and 2 yields an instance in which it holds for any solution with at most k editing operation each vertex has at least c incident edge modifications.

Rule 3. For a graph modification problem, assume we can for some constant $c \geq 1$ in polynomial time reduce to an instance where the number of edge modifications incident on each vertex is at least c. If the graph contains more than $2k/c$ vertices, then return a trivial no-instance.

The above observation together with Rule 3 yields the problem kernel of oder $2k/c$ for graph modification problems that are neighborhood-reconstructible and allow isolation. For the running time bound of the kernelization, we make use of the fact that neighborhood reconstruction runs in linear time for all three problems.

Theorem 6. *For any $c \geq 1$, 2-Cluster Editing, Sparse Split Graph Editing, and P_3-Bag Editing have a kernel with at most $2k/c$ vertices that can be computed in $O\big(nm + ck^2 \cdot \left(\frac{2k}{c-1}\right)^c\big)$ time.*

Proof. Let $\delta(u)$ denote the number of edge modifications incident on a vertex u. By Observation 2, we have $2k = \sum_{u \in V} \delta(u) \geq cn$, implying $n \leq 2k/c$. The straightforward running time of $O(n^c m)$ caused by Rule 1 can be improved by applying data reduction in rounds for $c' = 1$ to $c' = c$. The first round with $c' = 1$ takes $O(nm)$ time and produces an instance with at most $2k$ vertices. Before a round with $c' \geq 2$, there are at most $2k/(c'-1)$ vertices left, and the number of edges can be bounded by $O(k^2)$. Thus, the remaining rounds run in time

$$\sum_{c'=2}^{c} O\left(\left(\frac{2k}{c'-1}\right)^{c'} k^2\right) = O\left(ck^2 \cdot \left(\frac{2k}{c-1}\right)^{c}\right). \qquad \square$$

Note that for 2-Cluster Editing, already for $c = 1$, we obtain a kernel with $2k$ vertices, improving the $4k + 2$-vertex kernel derived from a more general result for d-Cluster Editing [14].

Subexponential-time algorithms. We can use our strict kernelization schemes to obtain subexponential-time algorithms.

Theorem 7. *If a graph problem can be solved in $2^{O(n)}$ time and it has a strict kernelization scheme that produces for any $c > 0$ in $n^{O(c)}$ time a kernel of at most $O(k/c)$ vertices, then it can be solved in $2^{O(\sqrt{k \log k})} + n^{O(1)}$ time.*

Proof. We kernelize for increasing c up to $c = \sqrt{k/\log k}$ and then solve the instance. This requires $k^{O(\sqrt{k/\log k})} + n^{O(1)} + 2^{O(\sqrt{k \log k})} = 2^{O(\sqrt{k \log k})} + n^{O(1)}$ time. □

For Sparse Split Graph Editing and P_3-Bag Editing, this running time is similar to the known subexponential-time algorithms [7], for 2-Cluster Editing this running time almost meets the best known running time [11]. We can also use Theorem 7 to rule out strict kernelization schemes for certain problems with known lower bounds on their running time.

Theorem 8. Cluster Editing *does not not have a kernelization scheme that produces for any $c > 0$ in $n^{O(c)}$ time a kernel of at most $O(k/c)$ vertices, unless the exponential time hypothesis (ETH) is false.*

Proof. Cluster Editing is easily solved in $2^{O(n)}$ time by standard dynamic programming over vertex subsets. Moreover, assuming ETH, there is no $2^{o(k)} \cdot n^{O(1)}$ time algorithm for Cluster Editing [17]. □

5 Outlook

Several open questions remain. For example, is Sparse Split Graph Editing APX-hard, or does it have a PTAS? Possibly a PTAS for the 2-Correlation Clustering problem [13] can be adapted. Furthermore, it seems worthwhile to explore the relation between kernelization schemes, subexponential-time solvability and polynomial-time approximation schemes more closely. For example, it would be interesting to investigate whether, similar to efficient polynomial-time approximation schemes (EPTAS) there are *efficient* strict kernelization schemes with running time $f(1/\varepsilon) \cdot (|I| + k)^{O(1)}$. Finally, it is open to find further applications of strict kernelization schemes. We would like to remark that the schemes also apply to the edge deletion variants of the considered problems. Preliminary considerations indicate that some of these problems admit efficient strict kernelization schemes.

Acknowledgment. We are grateful to Henning Fernau for fruitful discussions about the problems considered in this work.

References

[1] F. N. Abu-Khzam and H. Fernau. Kernels: Annotated, proper and induced. In *Proc. 2nd IWPEC*, volume 4169 of *LNCS*, pages 264–275. Springer, 2006.

[2] M. D. Barrus, M. Kumbhat, and S. G. Hartke. Graph classes characterized both by forbidden subgraphs and degree sequences. *J. Graph Theory*, 57(2): 131–148, 2008.

[3] S. Bessy, F. V. Fomin, S. Gaspers, C. Paul, A. Perez, S. Saurabh, and S. Thomassé. Kernels for feedback arc set in tournaments. *J. Comput. System Sci.*, 77(6):1071–1078, 2011.

[4] S. P. Borgatti and M. G. Everett. Models of core/periphery structures. *Soc. Networks*, 21(4):375–395, 1999.

[5] L. Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Inf. Process. Lett.*, 58(4):171–176, 1996.

[6] P. Damaschke and O. Mogren. Editing the simplest graphs. In *Proc. 8th WALCOM*, volume 8344 of *LNCS*, pages 249–260. Springer, 2014.

[7] P. Damaschke and O. Mogren. Editing simple graphs. *J. Graph Algorithms Appl.*, 18(4):557–576, 2014. doi: 10.7155/jgaa.00337.

[8] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity.* Texts in Computer Science. Springer, 2013.

[9] T. Feder and P. Hell. On realizations of point determining graphs, and obstructions to full homomorphisms. *Discrete Math.*, 308(9):1639–1652, 2008.

[10] H. Fernau. Parameterized algorithmics: A graph-theoretic approach. Habilitationsschrift, Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, 2005.

[11] F. V. Fomin, S. Kratsch, M. Pilipczuk, M. Pilipczuk, and Y. Villanger. Tight bounds for parameterized complexity of cluster editing with a small number of clusters. *J. Comput. System Sci.*, 80(7):1430–1447, 2014.

[12] M. R. Garey, D. S. Johnson, and L. J. Stockmeyer. Some simplified NP-complete graph problems. *Theor. Comput. Sci.*, 1(3):237–267, 1976.

[13] I. Giotis and V. Guruswami. Correlation clustering with a fixed number of clusters. *Theory of Computing*, 2(1):249–266, 2006.

[14] J. Guo. A more effective linear kernelization for cluster editing. *Theor. Comput. Sci.*, 410(8-10):718–726, 2009.

[15] P. L. Hammer and B. Simeone. The splittance of a graph. *Combinatorica*, 1 (3):275–284, 1981.

[16] P. Hell. Graph partitions with prescribed patterns. *Eur. J. Combin.*, 35: 335–353, 2014.

[17] C. Komusiewicz and J. Uhlmann. Cluster editing with locally bounded modifications. *Discrete Appl. Math.*, 160(15):2259–2270, 2012.

[18] I. Kovác, I. Selecéniová, and M. Steinová. On the clique editing problem. In *Proc. 39th MFCS*, volume 8635 of *LNCS*, pages 469–480. Springer, 2014.

[19] J. M. Lewis and M. Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *J. Comput. System Sci.*, 20(2):219–230, 1980.

[20] A. Natanzon, R. Shamir, and R. Sharan. Complexity classification of some edge modification problems. *Discrete Appl. Math.*, 113:109–128, 2001.

[21] R. Shamir, R. Sharan, and D. Tsur. Cluster graph modification problems. *Discrete Appl. Math.*, 144(1–2):173–182, 2004.

[22] B. Y. Wu and L.-H. Chen. Parameterized algorithms for the 2-clustering problem with minimum sum and minimum sum of squares objective functions. *Algorithmica*, 2014. doi: 10.1007/s00453-014-9874-8. To appear.

[23] W. Xie. Obstructions to trigraph homomorphisms. Master's thesis, School of Computing Science, Simon Fraser University, British Columbia, Canada, 2006.

A Proofs

A.1 Full proof of Theorem 2

Proof. Clearly, any sparse split graph has such a degree sequence. For the converse, note that each such sequence defines a sparse split graph with $n-c-1$ singletons and a clique of $c+1$ vertices. Moreover, sparse split graphs are *unigraphs*, that is, each sparse split graph is the only realization of its degree sequence, up to isomorphism [2, Proposition 6]. □

A.2 Full proof of Lemma 1

Proof. Consider a sparse split graph $G = (V, E)$ and let A denote the clique in G. Let $\bar{G}$ be the complement graph of G. Then, A is an independent set in $\bar{G}$. Moreover, $V \setminus A$ is a clique in $\bar{G}$ since it is an independent set in G. Finally, in $\bar{G}$ each vertex in A is adjacent to each vertex in $V \setminus A$. Hence, $\bar{G}$ is a dense split graph. The converse can be shown in an analogous manner. □

A.3 Full proof of Theorem 5

Proof. The containment in NP is easy to see: just guess an edge modification set and check whether the resulting graph is indeed a P_3-bag graph. To prove the NP-hardness, we provide a polynomial-time reduction from the Bisection problem, which was shown to be NP-hard by Garey et al. [12].

Bisection
Input: A graph $G = (V, E)$ and an integer $k \in \mathbb{N}$.
Question: Does G have a bisection with cut size at most k, that is, a partition of V into two sets V_1 and V_2 such that $|V_1| = |V_2|$ and $|E(V_1, V_2)| \leq k$?

Given a Bisection instance $(G = (V, E), k)$ with $m > k$ we construct an equivalent P_3-Bag Editing instance $(G' = (V', E'), k')$ as follows. First, copy G into G'. Next, add for each vertex $v \in V$ a clique with n^2 vertices to G' and make in G' all vertices in this clique adjacent to v. Denote the vertices in this clique by $C(v)$. We call these cliques *pendant cliques* to distinguish them from the at most two maximal cliques in the P_3-bag graph. We first explain the intuition behind the construction. The pendant cliques are pairwise non-adjacent. This forces a balanced "distribution" of the pendant cliques to the two maximal cliques of the P_3-bag graph as any non-balanced distribution exceeds the budget (which we define below). Then the balanced distribution of the pendant cliques forces the original vertex set V to be also split into two equal size sets. Hence, choosing the budget k' appropriately ensures a cut size of at most k between these two sets.

To define k', we use $t := n/2$ to denote the size of the two parts in a bisection of G and set

$$k' := \underbrace{n^4 \cdot 2\binom{t}{2}}_{\substack{\text{edges added}\\ \text{between cliques}}} + \underbrace{n^2 \cdot n(t-1)}_{\substack{\text{edges added}\\ \text{between cliques and}\\ \text{original vertices}}} + \underbrace{k}_{\substack{\text{edges removed}\\ \text{in cut of}\\ \text{bisection}}} + \underbrace{2\binom{t}{2} - (m-k).}_{\substack{\text{edges added between}\\ \text{original vertices inside the}\\ \text{two parts of the partition}}}$$

We now prove that (G, k) is a yes-instance of Bisection $\iff$ (G', k') is a yes-instance of P_3-Bag Editing.

"$\Rightarrow$:" Let V_1 and V_2 denote a bisection with cut size at most k. For a vertex subset $X \subseteq V$, let $C(X) := \bigcup_{v \in X} C(v)$ denote the union of its pendant cliques. In order to transform G' into a P_3-bag graph, we add all missing edges within the sets $V_1 \cup C(V_1)$ and $V_2 \cup C(V_2)$ and remove all edges with endpoints in both these sets. Observe that the resulting graph is indeed a P_3-bag graph: each of the two sets $V_1 \cup C(V_1)$ and $V_2 \cup C(V_2)$ forms a clique and there are no edges between these sets. Furthermore, the choice of k' ensures that the budget constraint is satisfied: we have to add the edges between the cliques $C(v_i)$, $1 \leq i \leq n$, between the cliques $C(v_i)$ and the original vertices, between the original vertices, and we remove the edges in the cut of the bisection.

"$\Leftarrow$:" Let S be a minimum-size P_3-bag edge modification set for G' with $|S| \leq k$ and let $G'' := G' \triangle S$ be the resulting P_3-bag graph. Denote by V'_1, V'_2, and $V'_{1,2}$ the vertices that are only in the first, only in the second, or in both maximal cliques of G''. Hence, the two cliques are $V'_1 \cup V'_{1,2}$ and $V'_2 \cup V'_{1,2}$. Note that $V'_1 \cup V'_2 \cup V'_{1,2} = V'$. Observe that the sets V'_1 and V'_2 are both non-empty, since there is insufficient budget to transform the whole graph G' into a clique.

We first show that $C(V) \cap V'_{1,2} = \emptyset$. Assume towards a contradiction that for some pendant clique $C(v_i)$ we have $C(v_i) \cap V'_{1,2} \neq \emptyset$. Since all vertices in the pendant clique $C(v_i)$ are twins, we can assume by a result of [6, Proposition 1] that $C(v_i) \subseteq V'_{1,2}$. Without loss of generality assume that $v_i \in V'_1 \cup V'_{1,2}$. As each vertex in the pendant clique $C(v_i)$ has in G' exactly one neighbor outside of $C(v_i)$, namely v_i, it follows that $E_{G''}(C(v_i), V'_2) \subseteq S$, that is all edges between $C(v_i)$ and V'_2 are contained in S. Hence, $S \setminus E_{G''}(C(v_i), V'_2)$ is a smaller P_3-bag edge modification set (where $C(v_i)$ is contained in the set V'_1). This contradicts the minimality of S.

Next, we show that each of the two sets V'_1 and V'_2 contain exactly $n/2 = t$ pendant cliques. Assume towards a contradiction, that V'_1 contains at least $t+1$ pendant cliques and let $C(v_i)$ be one of them. As the closed neighborhood of each vertex of $C(v_i)$ in G' is $C(v_i) \cup v_i$, it follows that $E_{G''}(C(v_i), V'_1 \setminus (C(v_i) \cup v_i)) \subseteq S$. By the same argument, however,

$$S' = (S \setminus E_{G''}(C(v_i), V'_1 \setminus (C(v_i) \cup v_i))) \cup E_{G''}(C(v_i), V'_2)$$

is also P_3-bag edge modification set. Since $|V'_2| \leq (t-1)n^2 + n < (t+1)n^2 \leq |V'_1|$, it follows that $|S'| < |S|$; a contradiction to the minimality of S.

By the above, it follows that V'_1 and V'_2 contain each exactly t cliques from $C(V)$. Observe that each vertex in a clique $C(v_i)$ has only one neigh-

bor outside of $C(v_i)$. Hence, S contains all $n^4 \cdot 2\binom{t}{2}$ edges to make the t cliques in each set V_1' and V_2' all pairwise adjacent. Furthermore, each vertex $v \in V$ is in G'' adjacent to at least t cliques and in G' to exactly one clique, namely $C(v)$. Hence, S contains at least $n^2 \cdot n(t-1)$ further edges to make the n vertices in V adjacent to at least $t-1$ cliques each. Thus, the remaining budget is

$$\begin{aligned} k' - n^4 \cdot 2\binom{t}{2} - n^2 \cdot n(t-1) &= k + 2\binom{t}{2} - (m-k) < k + 2\binom{t}{2} \\ &= k + \frac{n}{2}\left(\frac{n}{2} - 1\right) < \frac{n^2}{2} + \frac{n^2}{4} < n^2. \end{aligned}$$

Since $|C(v)| = n^2$ and $C(v) \subseteq N_{G'}(v)$, it follows that every vertex $v \in V$ is in the same set V_1' or V_2' as $C(v)$. Now let $x := |E_{G'}(V \cap V_1', V \cap V_2')|$ denote the number of edges between original vertices that are in different cliques of G''. Then, x edge deletions are performed between V_1' and V_2'. Moreover, the number of edge insertions between the vertices from $V \cap V_1'$ and between the vertices from $V \cap V_2'$ is $2\binom{t}{2} - (m-x)$. The remaining budget for these modifications is $k + 2\binom{t}{2} - (m-k)$ and thus $x \leq k$. Thus, $V \cap V_1'$ and $V \cap V_2'$ denote a bisection of V in G with cut size at most k and hence (G, k) is a yes-instance of Bisection. □

A.4 Full proof of Lemma 5

Proof. The running time bound is easy to see. For soundness, we need to show

$$(G, k) \text{ is a yes-instance} \iff (G - u, k - \deg(u)) \text{ is a yes-instance,}$$

where $G - u$ is the graph G without vertex u.

"⇒": Consider a solution with at most k edge modifications. Deleting the vertex u in this solution yields another graph with the desired property as the property is hereditary. By the condition of the rule u is incident with at least $\deg(u)$ edge modifications. Thus, removing all edge modifications incident with u yields a solution for $G - u$ and this solution has at most $k - \deg(u)$ edge modifications.

"⇐": If we have a solution for $(G - u, k - \deg(u))$, we can extend it to a solution for G by additionally deleting all edges incident on u; this solution for G has cost at most $k - \deg(u) + \deg(u) = k$, and thus (G, k) is a yes-instance. □