

# Clustering with Local Restrictions

Daniel Lokshantov\* and Dániel Marx\*\*

**Abstract.** We study a family of graph clustering problems where each cluster has to satisfy a certain local requirement. Formally, let  $\mu$  be a function on the subsets of vertices of a graph  $G$ . In the  $(\mu, p, q)$ -PARTITION problem, the task is to find a partition of the vertices where each cluster  $C$  satisfies the requirements that (1) at most  $q$  edges leave  $C$  and (2)  $\mu(C) \leq p$ . Our first result shows that if  $\mu$  is an *arbitrary* polynomial-time computable monotone function, then  $(\mu, p, q)$ -PARTITION can be solved in time  $n^{O(q)}$ , i.e., it is polynomial-time solvable *for every fixed*  $q$ . We study in detail three concrete functions  $\mu$  (number of nonedges in the cluster, maximum degree of nonedges in the cluster, number of vertices in the cluster), which correspond to natural clustering problems. For these functions, we show that  $(\mu, p, q)$ -PARTITION can be solved in time  $2^{O(p)} \cdot n^{O(1)}$  and in randomized time  $2^{O(q)} \cdot n^{O(1)}$ , i.e., the problem is fixed-parameter tractable parameterized by  $p$  or by  $q$ .

## 1 Introduction

Partitioning objects into clusters or similarity classes is an important task in various applications such as data mining, facility location, interpreting experimental data, VLSI design, and many more. The partition has to satisfy certain constraints: typically, we want to ensure that objects in a cluster are “close” or “similar” to each other and/or objects in different clusters are “far” or “dissimilar.” Additionally, we may want to partition the data into a certain prescribed number  $k$  of clusters, or we may have upper/lower bounds on the size of the clusters. Different objectives and different distance/similarity measures give rise to specific combinatorial problems.

Correlation clustering [19, 1, 3, 20] deals with a specific form of similarity measure: for each pair of objects, we know that either they are similar or dissimilar. This means that the similarity information can be expressed as an undirected graph, where the vertices represent the objects and similar objects are adjacent. In the ideal situation every connected component of the graph is a clique, in which case the components form a clustering that completely agrees with the similarity information. However, due to inconsistencies in the data or experimental errors, such a perfect partitioning might not always be possible. The goal in correlation clustering is to partition the vertices into an arbitrary number of clusters in a way that agrees with the similarity information as much as possible: we want to minimize the number of pairs for which the clustering disagrees with the input data (i.e., similar pairs that are put into different clusters, or dissimilar pairs that are clustered together).

In many cases, such as in variants of the correlation clustering problem defined in the previous paragraph, the objective is to minimize the total error of the solution. Thus

---

\* University of Bergen, Bergen, Norway. daniello@ii.uib.no

\*\* Humboldt-Universität zu Berlin, Berlin, Germany. dmarx@cs.bme.hu

the goal is to find a solution that is good in a global sense, but this does not rule out the possibility that the solution contains clusters that are very bad. In this paper, the opposite approach is taken: we want to find a partition where each cluster is “good” in a certain local sense. This means that the partition has to satisfy a set of local constraints on each cluster, but we do not try to optimize the total fitness of clusters.

The setting in this paper is the following. We want to partition the graph into an arbitrary number of clusters such that (1) at most  $q$  edges leave each cluster, and (2) each cluster induces a graph that is “cluster-like.” Defining what we mean by the abstract notion of cluster-like gives rise to a family of concrete problems. Formally, let  $\mu$  be a function that assigns a nonnegative integer to each subset of vertices in the graph and let us require  $\mu(X) \leq p$  for every cluster  $X$  of the partition. There are many reasonable choices for the measure  $\mu$  that correspond to natural problems. In particular, in this paper we will obtain concrete results for the following three measures:

1.  $\text{nonedge}(X)$  is the number of nonedges induced by  $X$ ,
2.  $\text{nondeg}(X)$  is the maximum degree of the *complement* of the graph induced by  $X$  (i.e., each vertex of  $X$  is adjacent to all but at most  $\text{nondeg}(X)$  other vertices in  $X$ ), and
3.  $\text{size}(X) = |X|$  is the number of vertices of  $X$ .

The first two functions express that each cluster should induce a graph that is close to being a clique. The third function only requires that each cluster is small. For a given function  $\mu$  and integers  $p$  and  $q$ , we denote by  $(\mu, p, q)$ -PARTITION the problem of partitioning the vertices into clusters such that at most  $q$  edges leave each cluster and  $\mu(X) \leq p$  for every cluster.

Our first result is very simple yet powerful. Let  $\mu$  be a function satisfying the mild technical conditions that it is polynomial-time computable and monotone (i.e., if  $X \subseteq Y$ , then  $\mu(X) \leq \mu(Y)$ ). Observe that for example all three functions defined above satisfy these conditions. Our first result shows that for *every function*  $\mu$  satisfying these conditions and *every fixed integer*  $q$ , the problem  $(\mu, p, q)$ -PARTITION can be solved in polynomial time (the value  $p$  is considered to be part of the input). For example, it can be decided in polynomial time if there is a clustering where at most 13 edges leave each cluster and each cluster induces at most 27 nonedges (or even the more general question, where the maximum number  $p$  of nonedges is given in the input). This might be surprising: we believe that most people would guess that this problem is NP-hard. The algorithm is based on a simple application of uncrossing of posimodular functions and on the fact that for fixed  $q$  we can enumerate every (connected) cluster with at most  $q$  outgoing edges. The crucial observation is that if every vertex can be *covered* by a good cluster, then the vertices can be *partitioned* into good clusters. Thus the problem boils down to checking for each vertex  $v$  if it is contained in a suitable cluster.

While the algorithm is simple in hindsight, considerable efforts have been spent on solving some very particular special cases. For example, Heggernes et al. [13] gave a polynomial-time algorithm for  $(\text{nonedge}, 1, 3)$ -PARTITION and Langston and Plaut [15] argued that the very deep results of Robertson and Seymour on graph minors and immersions imply that  $(\text{size}, p, q)$ -PARTITION is polynomial-time solvable for every fixed  $p$  and  $q$ . These results follow as straightforward corollaries from our first result.

Although this simple algorithm is polynomial for every fixed  $q$ , the running time is about  $n^{O(q)}$ , thus it is not efficient even for small values of  $q$ . To improve the running time, we look at the problem from the viewpoint of parameterized complexity. We show that for several natural measures  $\mu$ , including the three defined above, the clustering problem can be solved in randomized time  $2^{O(q)} \cdot n^{O(1)}$ , that is, the problem is fixed-parameter tractable (FPT) parameterized by the bound  $q$  on the number of edges leaving a cluster. Moreover, the bound  $p$  can be assumed to be part of the input. Thus this algorithm can be efficient for small values of  $q$  (say,  $O(\log n)$ ) even if  $p$  is large. The algorithm has constant probability of error, but it can be derandomized at the cost of worse dependence on  $q$  in the running time. The problem (size,  $p, q$ )-PARTITION appears in the open problem list of the 1999 monograph of Downey and Fellows [8] under the name “Minimum Degree Partition,” where it is suggested that the problem is probably W[1]-hard parameterized by  $q$ . Our result answers this question by showing that the problem is FPT, contrary to the expectation of Downey and Fellows.

A crucial ingredient of our parameterized algorithm is the notion of *important separators*, which has been used (implicitly or explicitly) to obtain fixed-parameter tractability results for various cut or separator related problems. In particular, we use the “randomized selection of important sets” argument that was introduced very recently in [17] to prove the fixed-parameter tractability of (edge and vertex) multicut. With these tools at hand, we can reduce  $(\mu, p, q)$ -PARTITION to a special case that we call the “Satellite Problem.” We show that if the Satellite Problem is fixed-parameter tractable parameterized by  $q$  for a particular function  $\mu$ , then  $(\mu, p, q)$ -PARTITION is also fixed-parameter tractable parameterized by  $q$ . It seems that for many reasonable functions  $\mu$ , the Satellite Problem can be solved by dynamic programming techniques. In particular, this is true for the three functions defined above, and this results in randomized algorithms with running time  $2^{O(q)} \cdot n^{O(1)}$ . Note that the reduction to the SATELLITE PROBLEM works for every monotone  $\mu$ , and we need arguments specific to a particular  $\mu$  only in the algorithms for SATELLITE PROBLEM.

We also investigate  $(\mu, p, q)$ -PARTITION parameterized by  $p$  and show that for  $\mu = \text{size}$ ,  $\text{nonedge}$ , and  $\text{nondeg}$ , the problem is FPT parameterized by  $p$ : it can be solved in time  $2^{O(p)} \cdot n^{O(1)}$  (this time the value  $q$  is part of the input). For these results, we use a combination of color coding and dynamic programming.

Previous work on fixed-parameter tractability of clustering problems focused mostly on parameterization by the total error. In problems such as CLUSTER EDITING, the task is to modify the graph with at most  $k$  allowed editing operations into disjoint union of cliques [14, 11, 10]. Generalizations of the problem have been considered in [23, 9, 12], where the graph has to be modified in such a way that every component is “clique-like” (defined by measures similar to the ones in the current paper). It is not possible to directly compare these results with our results as we explore a different objective: instead of bounding the total number of operations required to turn the graph into clusters, we have a bound on the number of operations that can affect each cluster. However, in general, FPT results are more interesting for parameters that are typically smaller. Intuitively, the editing operations affecting a cluster is much smaller than the total number of operations, thus FPT results parameterized by local bounds on the clusters seems to be more interesting than results parameterized by the total number of operations.

## 2 Clustering and uncrossing

Given an undirected graph  $G$ , we denote by  $\Delta(X)$  the set of edges between  $X$  and  $V(G) \setminus X$ , and define  $d(X) = |\Delta(X)|$ . We will use two well-known and easily checkable properties of the function  $d$ : for  $X, Y \subseteq V(G)$ ,  $d$  satisfies the *submodular* and *posimodular* inequalities

$$d(X) + d(Y) \geq d(X \cap Y) + d(Y \cup X) \text{ and } d(X) + d(Y) \geq d(X \setminus Y) + d(Y \setminus X).$$

Let  $\mu : 2^{V(G)} \rightarrow \mathbb{Z}^+$  be a function assigning nonnegative integers to sets of vertices of  $G$ . Let  $p$  and  $q$  be two integers. We say that a set  $C \subseteq V(G)$  is a  $(\mu, p, q)$ -cluster if  $\mu(C) \leq p$  and  $d(C) \leq q$ . A  $(\mu, p, q)$ -partition of  $G$  is a partition of  $V(G)$  into  $(\mu, p, q)$ -clusters. The main problem considered in this paper is finding such a partition. A necessary condition for the existence of  $(\mu, p, q)$ -partition is that for every vertex  $v \in V(G)$  there is a  $(\mu, p, q)$ -cluster that contains  $v$ . Therefore, we are also interested in the problem of finding a cluster that contains a particular vertex.

$(\mu, p, q)$ -PARTITION  
 Input: A graph  $G$ , integers  $p, q$ .  
 Find: A  $(\mu, p, q)$ -partition of  $G$ .

$(\mu, p, q)$ -CLUSTER  
 Input: Graph  $G$ , integers  $p, q$ , vertex  $v$ .  
 Find: A  $(\mu, p, q)$ -cluster  $C$  containing  $v$ .

The main observation of this section is that if  $\mu$  is *monotone* (i.e.,  $\mu(X) \leq \mu(Y)$  for every  $X \subseteq Y$ ), then this is actually a sufficient condition. Therefore, in these cases, it is sufficient to solve  $(\mu, p, q)$ -CLUSTER.

**Lemma 1.** *Let  $G$  be a graph, let  $p, q \geq 0$  be two integers, and let  $\mu : 2^{V(G)} \rightarrow \mathbb{Z}^+$  be a monotone function. If every  $v \in V(G)$  is contained in some  $(\mu, p, q)$ -cluster, then  $G$  has a  $(\mu, p, q)$ -partition. Furthermore, given a set of  $(\mu, p, q)$ -clusters  $C_1, \dots, C_n$  whose union is  $V(G)$ , a  $(\mu, p, q)$ -partition can be found in polynomial time.*

*Proof.* Let us consider a collection  $C_1, \dots, C_n$  of  $(\mu, p, q)$ -clusters whose union is  $V(G)$ . If the sets are pairwise disjoint, then they form a partition of  $V(G)$  and we are done. If  $C_i \subseteq C_j$ , then the union remains  $V(G)$  even after throwing away  $C_i$ . Thus we can assume that no set is contained in another. Suppose that  $C_i$  and  $C_j$  intersect. Now either  $d(C_i) \geq d(C_i \setminus C_j)$  or  $d(C_j) \geq d(C_j \setminus C_i)$  must be true: it is not possible that both  $d(C_i) < d(C_i \setminus C_j)$  and  $d(C_j) < d(C_j \setminus C_i)$  hold, as this would violate the posimodularity of  $d$ . Suppose that  $d(C_j) \geq d(C_j \setminus C_i)$ . Now the set  $C_j \setminus C_i$  is also a  $(\mu, p, q)$ -cluster: we have  $d(C_j \setminus C_i) \leq d(C_j) \leq q$  by assumption and  $\mu(C_j \setminus C_i) \leq \mu(C_j) \leq p$  from the monotonicity of  $\mu$ . Thus we can replace  $C_i$  by  $C_i \setminus C_j$  in the collection: it will remain true that the union of the clusters is  $V(G)$ . Similarly, if  $d(C_i) \geq d(C_i \setminus C_j)$ , then we can replace  $C_j$  by  $C_j \setminus C_i$ .

Repeating these steps (throwing away subsets and resolving intersections), we eventually arrive to a pairwise disjoint collection of  $(\mu, p, q)$ -clusters. Each step decreases the total size of clusters (note that when we replace  $C_i$  by  $C_i \setminus C_j$ , then  $C_i \cap C_j \neq \emptyset$ ). Therefore, this process terminates after a polynomial number of steps.  $\square$

In light of Lemma 1, it is sufficient to find a  $(\mu, p, q)$ -cluster  $C_v$  for each vertex  $v \in V(G)$ . If there is a vertex  $v$  for which there is no such cluster  $C_v$ , then obviously there

is no  $(\mu, p, q)$ -partition; if we have such a  $C_v$  for every vertex  $v$ , then Lemma 1 gives us a  $(\mu, p, q)$ -partition in polynomial time. For fixed  $q$ ,  $(\mu, p, q)$ -CLUSTER can be solved by brute force if  $\mu$  is polynomial-time computable: enumerate every set  $F$  of at most  $q$  edges and check if the component of  $G \setminus F$  containing  $v$  is a  $(\mu, p, q)$ -cluster. If  $C_v$  is a  $(\mu, p, q)$ -cluster containing  $v$ , then we find it when  $F = \Delta(C_v)$  is reached.

**Theorem 2.** *Let  $\mu$  be a polynomial-time computable monotone function. Then for every fixed  $q$ , there is an  $n^{O(q)}$  time algorithm for  $(\mu, p, q)$ -PARTITION.*

As we have seen, an algorithm for  $(\mu, p, q)$ -CLUSTER gives us an algorithm for  $(\mu, p, q)$ -PARTITION. In the rest of the paper, we devise more efficient algorithms for  $(\mu, p, q)$ -CLUSTER than the  $n^{O(q)}$  time brute force method described above.

### 3 Parameterization by $q$

The main result of this section is that  $(\mu, p, q)$ -PARTITION is (randomized) fixed-parameter tractable parameterized by  $q$  for the three functions nonedge, nondeg, and size.

**Theorem 3.** *There is a randomized algorithm for (size,  $p, q$ )-PARTITION, (nonedge,  $p, q$ )-PARTITION and (nondeg,  $p, q$ )-PARTITION using  $2^{O(q)}|V(G)|^{O(1)}$  time. If the input instance is a yes-instance the algorithm incorrectly returns no with probability less than  $\frac{1}{2}$ . On no-instances the algorithm always correctly answers no.*

By Lemma 1, all we need to show is that  $(\mu, p, q)$ -CLUSTER is fixed-parameter tractable parameterized by  $q$ . We introduce a somewhat technical variant of this question, the SATELLITE PROBLEM, and show that for every monotone function  $\mu$ , if SATELLITE PROBLEM is FPT, then  $(\mu, p, q)$ -CLUSTER is FPT as well. Thus we need arguments specific to a particular  $\mu$  only in solving the SATELLITE PROBLEM.

#### SATELLITE PROBLEM

Input: A graph  $G$ , integers  $p, q$ , a vertex  $v \in V(G)$ , a partition  $V_0, V_1, \dots, V_n$  of  $V(G)$  such that  $v \in V_0$  and there is no edge between  $V_i$  and  $V_j$  for any  $1 \leq i < j \leq n$ .

Find: A  $(\mu, p, q)$ -cluster  $C$  with  $V_0 \subseteq C$  and for every  $1 \leq i \leq n$ , either  $C \cap V_i = \emptyset$  or  $V_i \subseteq C$ .

That is, for every  $V_i$ , we have to decide whether to include or exclude it from the solution  $C$  (see Fig. 1). If we exclude  $V_i$  from  $C$ , then  $d(C)$  increases by the number of edges between  $V_0$  and  $V_i$ . If we include  $V_i$  into  $C$ , then  $\mu(C)$  increases accordingly. Thus we need to solve the knapsack-like problem of including sufficiently many  $V_i$  such that  $d(C) \leq q$ , but not including too many to ensure  $\mu(C) \leq p$ . As we shall see in Section 3.3, in many cases this problem can be solved by dynamic programming (and some additional arguments). The important fact that we use is that there are no edges between  $V_i$  and  $V_j$ , thus for most reasonable functions  $\mu$ , the way  $\mu(C)$  increases by including  $V_i$  is fairly independent from whether  $V_j$  is included in  $C$  or not.

The reduction to SATELLITE PROBLEM uses the concept of important separators (Section 3.1). The reduction itself is given in Section 3.2. In Section 3.3, we show how the Satellite Problem can be solved for the three functions nonedge, nondeg, size.

### 3.1 Important separators

The notion of *important separators* was introduced in [16] to prove the fixed-parameter tractability of multiway cut problems. This notion turned out to be useful in other applications as well [5, 6, 22]. The basic idea is that in many problems where terminals need to be separated in some way, it is sufficient to consider separators that are “as far as possible” from one of the terminals.

Since there are some small differences between edge and vertex separators, and some of the results appear only implicitly in previous papers, we make the paper self-contained by restating all the definitions and by reproving all the required results in this section. Let  $s, t$  be two vertices of a graph  $G$ . An  $s - t$  separator is a set  $S \subseteq E(G)$  of edges separating  $s$  and  $t$ , i.e., there is no  $s - t$  path in  $G \setminus S$ . An  $s - t$  separator is *inclusionwise minimal* if there is an  $s - t$  path in  $G \setminus S'$  for every  $S' \subset S$ .

**Definition 4.** Let  $s, t \in V(G)$  be vertices,  $S \subseteq E(G)$  be an  $s - t$  separator, and let  $K$  be the component of  $G \setminus S$  containing  $s$ . We say that  $S$  is an *important  $s - t$  separator* if it is inclusionwise minimal and there is no  $s - t$  separator  $S'$  with  $|S'| \leq |S|$  such that  $K \subset K'$  for the component  $K'$  of  $G \setminus S'$  containing  $s$ .

The main observation that we use is the following bound:

**Lemma 5.** Let  $s, t \in V(G)$ . If  $\mathcal{S}$  is the set of all important  $s - t$  separators, then  $\sum_{S \in \mathcal{S}} 4^{-|S|} \leq 1$ . Thus  $\mathcal{S}$  contains at most  $4^k$  separators of size at most  $k$ .

### 3.2 Reduction to the Satellite Problem

In this section we show how to reduce  $(\mu, p, q)$ -CLUSTER to the SATELLITE PROBLEM.

**Lemma 6.** If SATELLITE PROBLEM can be solved in time  $f(q) \cdot n^{O(1)}$  for some monotone  $\mu$ , then there is a randomized  $2^{O(q)} \cdot f(q) \cdot n^{O(1)}$  algorithm with constant error probability that finds a  $(\mu, p, q)$ -cluster containing  $v$  (if one exists).

The crucial definition of this section is the following:

**Definition 7.** We say that a set  $X \subseteq V(G)$ ,  $v \notin X$  is *important* if

1.  $d(X) \leq q$ ,
2.  $G[X]$  is connected,
3. there is no  $Y \supset X$ ,  $v \notin Y$  such that  $d(Y) \leq d(X)$  and  $G[Y]$  is connected.

It is easy to see that  $X$  is an important set if and only if  $\Delta(X)$  is an important  $u - v$  separator for every  $u \in X$ . Thus we can use Lemma 5 to enumerate every important set. Lemma 8 establishes the connection between important sets and finding  $(\mu, p, q)$ -clusters: we can assume that the components of  $G \setminus C$  for the solution  $C$  are important sets. In Lemma 9, we show that by randomly choosing important sets, with some probability we can obtain an instance of the Satellite Problem where  $V_1, \dots, V_n$  contain all the components of  $G \setminus C$ . This gives us the reduction stated in Lemma 6 above.

**Lemma 8.** Let  $C$  be a inclusionwise minimal  $(\mu, p, q)$ -cluster containing  $v$ . Then every component of  $G \setminus C$  is an important set.

*Proof.* Let  $X$  be a component of  $G \setminus C$ . It is clear that  $X$  satisfies the first two properties of Definition 7 (note that  $\Delta(X) \subseteq \Delta(C)$ ). Thus let us suppose that there is a  $Y \supset X$ ,  $v \notin Y$  such that  $d(Y) \leq d(X)$  and  $G[Y]$  is connected. Let  $C' := C \setminus Y$ . Note that  $C'$  is a proper subset of  $C$ : every neighbor of  $X$  is in  $C$ , thus a connected superset of  $X$  has to contain at least one vertex of  $C$ . It is easy to see that  $C'$  is a  $(\mu, p, q)$ -cluster: we have  $\Delta(C') = (\Delta(C) \setminus \Delta(X)) \cup \Delta(Y)$  and therefore  $d(C') \leq d(C) - d(X) + d(Y) \leq d(C) \leq q$  and  $\mu(C') \leq \mu(C) \leq p$  (by the monotonicity of  $\mu$ ). This contradicts the minimality of  $C$ .  $\square$

**Lemma 9.** *Given a graph  $G$ , vertex  $v \in V(G)$ , integers  $p, q$ , and a monotone function  $\mu : 2^{V(G)} \rightarrow \mathbb{Z}^+$ , we can construct in time  $2^{O(q)} \cdot n^{O(1)}$  an instance  $I$  of the SATELLITE PROBLEM such that*

- *If some  $(\mu, p, q)$ -cluster contains  $v$ , then  $I$  is a yes-instance with probability  $2^{-O(q)}$ ,*
- *If there is no  $(\mu, p, q)$ -cluster containing  $v$ , then  $I$  is a no-instance.*

*Proof.* For every  $u \in V(G)$ ,  $u \neq v$ , let us use the algorithm of Lemma 5 to enumerate every important  $u - v$  separator of size at most  $q$ . For every such separator  $S$ , let us put the component  $K$  of  $G \setminus S$  containing  $u$  into the collection  $\mathcal{S}$ . Note that a component  $K$  can be obtained for more than one vertex  $u$ , but we put only one copy into  $\mathcal{S}$ .

Let  $\mathcal{S}'$  be a subset of  $\mathcal{S}$ , where each member  $K$  of  $\mathcal{S}$  is chosen with probability  $2^{-d(K)}$  independently at random. Let  $Z$  be the union of the sets in  $\mathcal{S}'$ , let  $V_1, \dots, V_n$  be the connected components of  $G[Z]$ , and let  $V_0 = V(G) \setminus Z$ . It is clear that  $V_0, V_1, \dots, V_n$  give an instance  $I$  of SATELLITE PROBLEM, and a solution for  $I$  gives a  $(\mu, p, q)$ -cluster containing  $v$ . Thus we only need to show that if there is a  $(\mu, p, q)$ -cluster  $C$  containing  $v$ , then  $I$  is a yes-instance with probability  $2^{-O(q)}$ .

Let  $C$  be an inclusionwise minimal  $(\mu, p, q)$ -cluster containing  $v$ . Let  $B$  be the vertices on the boundary of  $C$ , i.e., the vertices of  $C$  incident to  $\Delta(C)$ . Let  $K_1, \dots, K_t$  be the components of  $G \setminus C$ . Note that every edge of  $\Delta(C)$  enters some  $K_i$ , thus  $\sum_{i=1}^t d(K_i) = d(C) \leq q$ . By Lemma 8, every  $K_i$  is an important set, and hence it is in  $\mathcal{S}$ . Consider the following two events:

- (1) Every component  $K_i$  of  $G \setminus C$  is in  $\mathcal{S}'$  (and hence  $K_i \subseteq Z$ ).
- (2)  $Z \cap B = \emptyset$ .

The probability that (1) holds is  $\prod_{i=1}^t 4^{-d(K_i)} = 4^{-\sum_{i=1}^t d(K_i)} \geq 4^{-q}$ . Event (2) holds if for every  $b \in B$ , no set  $K \in \mathcal{S}$  with  $b \in K$  is selected into  $\mathcal{S}'$ . Recall that for every  $K \in \mathcal{S}$  with  $b \in K$ ,  $\Delta(K)$  is an important  $b - v$  separator. Thus by Lemma 5,  $\sum_{K \in \mathcal{S}, b \in K} 4^{-|d(K)|} \leq 1$ . The probability that  $Z \cap B = \emptyset$  can be bounded by

$$\begin{aligned} \prod_{K \in \mathcal{S}, K \cap B \neq \emptyset} (1 - 4^{-d(K)}) &\geq \prod_{b \in B} \prod_{K \in \mathcal{S}, b \in K} (1 - 4^{-d(K)}) \geq \prod_{b \in B} \prod_{K \in \mathcal{S}, b \in K} \exp\left(\frac{-4^{-d(K)}}{(1 - 4^{-d(K)})}\right) \\ &\geq \prod_{b \in B} \prod_{K \in \mathcal{S}, b \in K} \exp\left(-\frac{4}{3} \cdot 4^{-d(K)}\right) = \prod_{b \in B} \exp\left(-\frac{4}{3} \cdot \sum_{K \in \mathcal{S}, b \in K} 4^{-d(K)}\right) \geq (e^{-\frac{4}{3}})^{|B|} \geq e^{-4q/3}. \end{aligned}$$

In the first inequality, we use that every term is less than 1 and every term on the right hand side appears at least once on the left hand side; in the second inequality, we use

that  $1 + x \geq \exp(x/(1 + x))$  for every  $x > -1$ . Events (1) and (2) are independent: (1) is a statement about the selection of subsets of  $\mathcal{S}$  that are disjoint from  $B$ , while (2) involves only sets intersecting  $B$ . Thus by probability  $2^{-O(q)}$ , both (1) and (2) hold.

Suppose that both (1) and (2) hold, we show that instance  $I$  of the SATELLITE PROBLEM is a yes-instance. In this case, every component  $K_i$  of  $G \setminus C$  is a component  $V_j$  of  $G[Z]$ :  $K_i \subseteq Z$  by (1) and every neighbor of  $K_i$  is outside  $Z$ . Thus  $C$  is a solution of  $I$ , as it can be obtained as the union of  $V_0$  and some components of  $G[Z]$ .  $\square$

To derandomize the proof of Lemma 9 and obtain a deterministic version of Lemma 6, we use the standard technique of splitters (see details in the appendix). Note that the dependence on  $q$  becomes worse in the deterministic version ( $2^{O(q^2)}$  instead of  $2^{O(q)}$ ). We leave it as a question for future work to improve the derandomization to match the bound of the randomized algorithm.

**Lemma 10.** *If SATELLITE PROBLEM can be solved in time  $f(q) \cdot n^{O(1)}$  for some monotone  $\mu$ , then there is a  $2^{O(q^2)} \cdot f(q) \cdot n^{O(1)}$  algorithm for  $(\mu, p, q)$ -CLUSTER.*

### 3.3 Solving the Satellite Problem

In this section, we give efficient algorithms for solving the SATELLITE PROBLEM when the function  $\mu$  is size, nonedge and nondeg. We describe the three algorithms by increasing difficulty. In the case when  $\mu$  is size, solving the SATELLITE PROBLEM turns out to be equivalent to the classical KNAPSACK problem with polynomial bounds on the values and weights of the items.

Recall that the input to the SATELLITE PROBLEM is a graph  $G$ , integers  $p, q$ , a vertex  $v \in V(G)$ , a partition  $V_0, V_1, \dots, V_n$  of  $V(G)$  such that  $v \in V_0$  and there is no edge between  $V_i$  and  $V_j$  for any  $1 \leq i < j \leq n$ . The task is to find a vertex set  $C$ , such that  $C = V_0 \cup \bigcup_{i \in S} V_i$  for a subset  $S$  of  $\{1, \dots, n\}$  and  $C$  satisfies  $d(C) \leq q$  and  $\mu(C) \leq p$ . For a subset  $S$  of  $\{1, \dots, n\}$  we define  $C(S) = V_0 \cup \bigcup_{i \in S} V_i$ .

**Lemma 11.** *The SATELLITE PROBLEM for size can be solved in  $O(q|V(G)| \log |V(G)|)$  time.*

*Proof.* Notice that  $d(C) = d(V_0) - \sum_{i \in S} d(V_i)$ . Hence, we can reformulate the SATELLITE PROBLEM with  $\mu = \text{size}$  as finding a subset  $S$  of  $\{1, \dots, n\}$  such that  $\sum_{i \in S} d(V_i) \geq d(V_0) - q$  and  $\sum_{i \in S} |V_i| \leq p - |V_0|$ . Thus, we can associate with every  $i$  an item with value  $d(V_i)$  and weight  $|V_i|$ . The objective is to find a set of items with total value at least  $d(V_0) - q$  and total weight at most  $p - |V_0|$ . This problem is known as KNAPSACK and can be solved in  $O(nv \log w)$  time by a classical dynamic programming [4, 7] algorithm, where  $n$  is the number of items,  $v$  is the value we seek to attain and  $w$  is the weight limit. Since the value is bounded from above by  $q$  and the weight by  $|V(G)|$ , the statement of the lemma follows.  $\square$

The case that  $\mu = \text{nonedge}$  is slightly more complicated, however we can still solve it using a dynamic programming algorithm (see the appendix).

**Lemma 12.** *The SATELLITE PROBLEM for nonedge can be solved in  $O(pn|E(G)||V(G)|)$  time.*

For the version of SATELLITE PROBLEM when  $\mu = \text{nondeg}$  we do not have a polynomial time algorithm. Instead, we give a  $2^q |V(G)|^{O(1)}$  time randomized algorithm.

**Lemma 13.** *There is a randomized algorithm which given an instance of the nondeg-SATELLITE PROBLEM runs in  $2^q |V(G)|^{O(1)}$  time, correctly answers no on all no-instances and answers yes on yes-instances with probability at least  $e^{-2q}$ .*

Repeating the algorithm  $O(e^{2q})$  times will decrease the probability of false negatives from  $1 - e^{-2q}$  to  $\frac{1}{2}$ . Lemmata 9, 11, 12 and 13 give Theorem 3.

## 4 Parameterization by $p$

### 4.1 Algorithms

In this section, we give algorithms with running time  $2^{O(p)} |V(G)|^{O(1)}$ :

**Theorem 14.** *There is a  $8e^{p+o(p)} |V(G)|^{O(1)}$  time algorithm for (size,  $p, q$ )-PARTITION and a  $8e^{3p+o(p)} |V(G)|^{O(1)}$  time algorithm for (nonedge,  $p, q$ )-PARTITION and (nondeg,  $p, q$ )-PARTITION.*

Because of Lemma 1, it is sufficient to solve the corresponding  $(\mu, p, q)$ -CLUSTER problem within the same time bound. The setting is as follows. We are given a graph  $G$ , integers  $p$  and  $q$  and a vertex  $v$  in  $G$ . The objective is to find a set  $C$  not containing  $v$  such that  $d(C \cup \{v\}) \leq q$  and, depending on which problem we are solving, either  $|C \cup \{v\}| = \text{size}(C \cup \{v\}) \leq p$ ,  $\text{nonedge}(C \cup \{v\}) \leq p$  or  $\text{nondeg}(C \cup \{v\}) \leq p$ .

For a set  $S$  and vertex  $v$ , define  $\Delta(S, v)$  to be the set of edges with one endpoint in  $S$  and one in  $\{v\}$ . Define  $\bar{\Delta}(S, v)$  to be  $\Delta(S) \setminus \Delta(S, v)$ , and let  $d(S, v) = |\Delta(S, v)|$  and  $\bar{d}(S, v) = |\bar{\Delta}(S, v)|$ . We will say that a set  $C$  is  $v$ -minimal if  $v \notin C$  and  $d(C' \cup \{v\}) > d(C \cup \{v\})$  for every  $C' \subset C$ . As size, nonedge and nondeg are monotone we can focus on  $v$ -minimal sets  $C$ . The following fact uses that there are no parallel edges:

**Observation 15.** Let  $C$  be a  $v$ -minimal set. Then  $\bar{d}(C, v) < d(C, v) \leq |C|$

In particular, if  $v \notin C$  and  $\bar{d}(C, v) \geq d(C, v)$ , then  $d(v) \leq d(C \cup \{v\})$ , contradicting that  $C$  is minimal. Since  $\bar{d}(C, v) < |C|$ , it follows that  $C$  must contain a vertex  $u$  such that  $N[u] \subseteq C \cup \{v\}$ . Now we show that there are not too many  $v$ -minimal sets  $C$  of size at most  $p$  such that  $G[C]$  is connected.

**Lemma 16.** *For any graph  $G$ , vertex  $v$  and integer  $p$ , there are at most  $4^p |V(G)|$   $v$ -minimal sets  $C$  such that  $|C| \leq p$  and  $G[C]$  is connected. Furthermore, all such sets can be listed in  $O(4^p |V(G)|)$  time.*

*Proof.* By Observation 15, any  $v$ -minimal set  $C$  of size at most  $p$  satisfies  $\bar{d}(C, v) < p$ . Let  $S$  be a set such that  $|S| \leq p$  and  $G[S]$  is connected. Let  $F$  be a subset of  $N(S) \setminus \{v\}$  of size at most  $p - 1$ . We prove by downward induction on  $|S|$  and  $|F|$  that there are at most  $2^{2p - |S| - |F| - 1}$   $v$ -minimal sets such that  $|C| \leq p$ ,  $G[C]$  is connected,  $S \subseteq C$ , and  $F \cap C = \emptyset$ . If  $|S| = p$  then the only possibility for  $C$  is  $S$ , while  $2^{2p - |S| - |F| - 1} \geq 1$ . Similarly, consider the case that  $|F| = p - 1$ . Now, every vertex of  $F$  has at least one

edge into  $C$  and hence  $\bar{d}(C, v) = p - 1$ . Hence  $N(C) = F \cup \{v\}$  and the only possibility for  $C$  is the connected component of  $G \setminus (F \cup \{v\})$  that contains  $S$ . Hence there is one possibility for  $C$  and  $2^{2p - |S| - |F| - 1} \geq 1$ .

For the inductive step, consider a set  $S$  such that  $|S| \leq p$  and  $G[S]$  is connected and a subset  $F$  of  $N(S) \setminus \{v\}$  of size at most  $p - 1$ . We want to bound the number of  $v$ -minimal sets such that  $|C| \leq p$  and  $G[C]$  is connected,  $S \subseteq C$  and  $F \cap C = \emptyset$ . If  $N(S) \setminus (F \cup \{v\})$  is empty, then there is only one choice for  $C$ , namely  $S$ , and  $2^{2p - |S| - |F| - 1} \geq 1$ . Otherwise, consider a vertex  $u \in N(S) \setminus (F \cup \{v\})$ . By the induction hypothesis, the number of  $v$ -minimal sets such that  $|C| \leq p$  and  $G[C]$  is connected,  $S \cup \{u\} \subseteq C$  and  $F \cap C = \emptyset$  is at most  $2^{2p - |S| - |F| - 2}$ . Similarly, the number of  $v$ -minimal sets such that  $|C| \leq p$  and  $G[C]$  is connected,  $S \subseteq C$  and  $(F \cup \{u\}) \cap C = \emptyset$  is at most  $2^{2p - |S| - |F| - 2}$ . Since either  $u \in C$  or  $u \notin C$ , the two cases cover all possibilities for  $C$  and hence there are at most  $2 \cdot 2^{2p - |S| - |F| - 2} = 2^{2p - |S| - |F| - 1}$  possibilities for  $C$ .

For a fixed  $S$  and  $F$ , the above proof can be translated into a procedure which lists all  $v$ -minimal sets such that  $|C| \leq p$  and  $G[C]$  is connected,  $S \subseteq C$  and  $F \cap C = \emptyset$ . We run the procedure for  $S = \{u\}$  and  $F = \emptyset$  for every possible choice of  $u$ . Hence, there are at most  $4^p |V(G)|$   $v$ -minimal sets  $C$  such that  $|C| \leq p$  and  $G[C]$  is connected, and the sets can be efficiently listed. This concludes the proof.  $\square$

The following observation is handy for using Lemma 16.

**Observation 17.** Let  $C$  be a  $v$ -minimal set of  $G$  and  $G[S]$  be a connected component of  $G[C]$ . Then  $S$  is a  $v$ -minimal set.

In particular, if  $S$  is not a  $v$ -minimal set, then it contains a  $v$ -minimal set  $S' \subset S$  and it is easy to see that  $d(\{v\} \cup (C \setminus S) \cup S') \leq d(\{v\} \cup C)$ , contradicting the minimality of  $C$ . Observation 17 tells us that any  $v$ -minimal set is the union of connected  $v$ -minimal sets. This makes it possible to use Lemma 16. We are now ready to give an algorithm for (size,  $p, q$ )-CLUSTER, the easiest of the three clustering problems. Our algorithm is based on a combination of *color coding* [2] with a dynamic programming algorithm which uses the observations made in this section.

**Proposition 18 ([21]).** For every  $n, k$  there is a family of functions  $\mathcal{F}$  of size  $O(e^k \cdot k^{O(\log k)} \cdot \log n)$  such that every function  $f \in \mathcal{F}$  is a function from  $\{1, \dots, n\}$  to  $\{1, \dots, k\}$  and for every subset  $S$  of  $\{1, \dots, n\}$  there is a function  $f \in \mathcal{F}$  that is bijective when restricted to  $S$ . Furthermore, given  $n$  and  $k$ ,  $\mathcal{F}$  can be computed in time  $O(e^k \cdot k^{O(\log k)} \cdot \log n)$ .

**Lemma 19.** (size,  $p, q$ )-CLUSTER can be solved in time  $2^{O(p)} |V(G)|^{O(1)}$ .

*Proof.* We are given as input a graph  $G$  together with a vertex  $v$  and integers  $p$  and  $q$ . The task is to find a vertex set  $C$  of size at most  $p - 1$  such that  $d(\{v\} \cup C) \leq q$ . It is sufficient to search for a  $v$ -minimal set  $C$  satisfying these properties. By Observation 17,  $C$  can be decomposed into  $C = S_1 \cup S_2 \dots \cup S_t$  such that  $S_i$  is a connected  $v$ -minimal set for every  $i$ ,  $S_i \cap S_j = \emptyset$  for every  $i \neq j$  and no edge of  $G$  has one endpoint in  $S_i$  and the other in  $S_j$  for every  $i \neq j$ . The algorithm of Lemma 16 can be used to list all connected  $v$ -minimal sets  $S_1 \dots S_n$ ; we have  $n \leq 4^p |V(G)|$ . For a subset  $Z$  of

$\{1, \dots, n\}$ , define  $C(Z) = \{v\} \cup \bigcup_{i \in Z} S_i$ . Let  $Z \subseteq \{1, \dots, n\}$  be such that for every  $i, j \in Z$  with  $i \neq j$ , we have  $S_i \cap S_j = \emptyset$ . We have that  $|C(Z)| = 1 + \sum_{i \in Z} |S_i|$  and

$$d(C(Z)) \leq d(v) + \sum_{i \in Z} (\bar{d}(S_i, v) - d(S_i, v)).$$

If there is no edge with one endpoint in  $S_i$  and the other in  $S_j$  for some  $i \neq j$ ,  $i, j \in Z$ , then the inequality above holds with equality. Our algorithm will select a  $Z$  such that  $C = \bigcup_{i \in Z} S_i$ . To ensure that the algorithm picks  $Z$  such that the sets  $S_i$  and  $S_j$  will be disjoint for every pair of distinct integers  $i, j \in Z$  we will use color coding. In particular, we construct a family  $\mathcal{F}$  of functions from  $V(G) \setminus \{v\}$  to  $\{1, \dots, p-1\}$  as described in Proposition 18. The family  $\mathcal{F}$  has size  $O(e^p \cdot p^{O(\log p)} \cdot \log |V(G)|)$ .

For each function  $f \in \mathcal{F}$  we will think of the function as a coloring of  $V(G) \setminus \{v\}$  with colors from  $\{1, \dots, p-1\}$ . We will only look for a  $v$ -minimal set  $C$  whose vertices have different colors. This will not only ensure that any two sets  $S_i$  and  $S_j$  that we pick will be disjoint, it also automatically ensures that the size of the set  $C$  we return is at most  $p-1$ . If the input instance was a yes-instance then a solution set  $C$  exists, and the construction of  $\mathcal{F}$  ensures that there will be a function  $f \in \mathcal{F}$  which colors all vertices in  $C$  with different colors.

When considering a particular coloring  $f$ , we discard all sets from  $S_1, \dots, S_n$  which have two vertices of the same color, so from this point, without loss of generality, all sets in  $S_1, \dots, S_n$  have at most one vertex of each color. For a vertex set  $S$ , define  $\text{colors}(S)$  to be the set of colors occurring on vertices on  $G$ . For every  $0 \leq i \leq n$ ,  $0 \leq j \leq |E(G)|$  and  $R \subseteq \{1, \dots, p-1\}$ , we define  $T[i, j, S]$  to be true if there is a set  $Z \subseteq \{1, \dots, i\}$  such that all vertices of  $C(Z)$  have distinct colors,  $d(v) + \sum_{i \in Z} (\bar{d}(S_i, v) - d(S_i, v)) = j$  and  $\text{colors}(C(Z)) \subseteq R$ . Clearly, there is a  $v$ -minimal set  $C$  such that  $d(\{v\} \cup C) \leq j$  and all vertices of  $C$  have different color if and only if  $T[n, j, \{1, \dots, p-1\}]$  is true for some  $j \leq q$ . We can fill the table  $T$  using the following recurrence.

$$T[i, j, R] = \begin{cases} T[i-1, j, R] & \text{if } \text{colors}(S_i) \setminus R \neq \emptyset \\ T[i-1, j, R] \vee T[i-1, j + d(S_i, v) - \bar{d}(S_i, v), R \setminus \text{colors}(S_i)] & \text{otherwise} \end{cases} \quad (1)$$

Here we initialize  $T[0, d(v), \emptyset]$  to true. The table has size  $4^p |V(G)|^{O(1)} \cdot 2^p |V(G)|^{O(1)} = 8^p |V(G)|^{O(1)}$  and can be filled in time proportional to its size. Hence the total running time for the algorithm is  $(8e)^{p+o(p)} |V(G)|^{O(1)}$ .  $\square$

For  $(\text{size}, p, q)$ -CLUSTER the size of the set  $C$  we look for is already bounded by  $p$ . For  $(\text{nonedge}, p, q)$ -CLUSTER and  $(\text{nondeg}, p, q)$ -CLUSTER, we cannot make this assumption, thus further arguments and a more complicated dynamic programming are needed to obtain Theorem 14 (see details in the appendix).

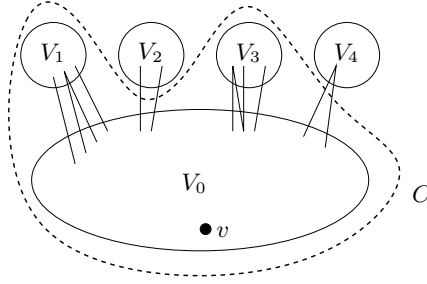
## 4.2 Hardness results

The algorithmic results in Section 3 still hold when parallel edges are allowed. Interestingly, the results in Section 4.1 do not: in particular, Observation 15 breaks down if there are parallel edges. The following hardness result shows that allowing parallel edges indeed make the problems more difficult:

**Theorem 20.**  $(\text{nonedge}, p, q)$ -PARTITION and  $(\text{nondeg}, p, q)$ -PARTITION are NP-complete for  $p = 0$  on graphs with parallel edges. The problem  $(\text{size}, p, q)$ -PARTITION is W[1]-hard parameterized by  $p$  on graphs with parallel edges (but in P for every fixed  $p$ ).

## References

1. N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: ranking and clustering. In *STOC 2005*, pages 684–693, New York, NY, USA, 2005. ACM.
2. N. Alon, R. Yuster, and U. Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.
3. N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56(1-3):89–113, 2004.
4. R. Bellman. Dynamic programming treatment of the travelling salesman problem. *J. ACM*, 9(1):61–63, 1962.
5. J. Chen, Y. Liu, and S. Lu. An improved parameterized algorithm for the minimum node multiway cut problem. In *WADS*, pages 495–506, 2007.
6. J. Chen, Y. Liu, S. Lu, B. O’Sullivan, and I. Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. *J. ACM*, 55(5), 2008.
7. T. Cormen, C. Leiserson, R. Rivest, and C. Stein. Introduction to algorithms, 2001.
8. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.
9. M. R. Fellows, J. Guo, C. Komusiewicz, R. Niedermeier, and J. Uhlmann. Graph-based data clustering with overlaps. In *COCOON*, pages 516–526, 2009.
10. M. R. Fellows, M. A. Langston, F. A. Rosamond, and P. Shaw. Efficient parameterized preprocessing for cluster editing. In *FCT*, pages 312–321, 2007.
11. J. Guo. A more effective linear kernelization for cluster editing. *Theor. Comput. Sci.*, 410(8-10):718–726, 2009.
12. J. Guo, I. A. Kanj, C. Komusiewicz, and J. Uhlmann. Editing graphs into disjoint unions of dense clusters. In *ISAAC*, pages 583–593, 2009.
13. P. Heggernes, D. Lokshtanov, J. Nederlof, C. Paul, and J. A. Telle. Generalized graph clustering: Recognizing  $(, )$ -cluster graphs. In *WG*, pages 171–183, 2010.
14. F. Hüffner, C. Komusiewicz, H. Moser, and R. Niedermeier. Fixed-parameter algorithms for cluster vertex deletion. *Theory Comput. Syst.*, 47(1):196–217, 2010.
15. M. A. Langston and B. C. Plaut. On algorithmic applications of the immersion order : An overview of ongoing work presented at the third slovenian international conference on graph theory. *Discrete Mathematics*, 182(1-3):191–196, 1998.
16. D. Marx. Parameterized graph separation problems. *Theoret. Comput. Sci.*, 351(3):394–406, 2006.
17. D. Marx and I. Razgon. Fixed-parameter tractability of multicut parameterized by the size of the cutset. To appear in *STOC 2011*.
18. L. Mathieson and S. Szeider. The parameterized complexity of regular subgraph problems and generalizations. In *CATS*, pages 79–86, 2008.
19. C. Mathieu, O. Sankur, and W. Schudy. Online correlation clustering. In *STACS*, pages 573–584, 2010.
20. C. Mathieu and W. Schudy. Correlation clustering with noisy input. In *SODA*, pages 712–728, 2010.
21. M. Naor, L. J. Schulman, and A. Srinivasan. Splitters and near-optimal derandomization. In *FOCS*, pages 182–191, 1995.
22. I. Razgon and B. O’Sullivan. Almost 2-sat is fixed-parameter tractable (extended abstract). In *ICALP 2008(1)*, pages 551–562, 2008.
23. R. van Bevern, H. Moser, and R. Niedermeier. Kernelization through tidying. In *LATIN*, pages 527–538, 2010.



**Fig. 1.** Instance of SATELLITE PROBLEM with a solution  $C$ .

## A Omitted proofs

### A.1 Proof of Lemma 5

*Proof (Proof of Lemma 5).* Let  $\lambda$  be the size of the smallest  $s - t$  separator. We prove by induction on the number of edges of  $G$  that  $\sum_{S \in \mathcal{S}} 4^{-|S|} \leq 2^{-\lambda}$ . If  $\lambda = 0$ , then there is a unique important  $s - t$  separator of size at most  $k$ : the empty set. Thus we can assume that  $\lambda > 0$ .

For an  $s - t$  separator  $S$ , let  $K_S$  be the component of  $G \setminus S$  containing  $s$  (i.e., if  $S$  is inclusionwise minimal, then  $S = \Delta(K)$ ). First we show the well-known fact that there is a unique  $s - t$  separator  $S^*$  of size  $\lambda$  such that  $K_{S^*}$  is inclusionwise maximal, i.e., there is no other  $s - t$  separator  $S$  of size  $\lambda$  with  $K_{S^*} \subset K_S$ . Suppose that there are two separators  $S'$  and  $S''$  with  $|S'| = |S''| = \lambda$  that are inclusionwise maximal in this sense. By the submodularity of  $d$ , we have

$$\underbrace{d(K_{S'})}_{=\lambda} + \underbrace{d(K_{S''})}_{=\lambda} \geq d(K_{S'} \cup K_{S''}) + \underbrace{d(K_{S'} \cap K_{S''})}_{\geq \lambda}.$$

The left hand side is exactly  $2\lambda$ , while the second term of the right hand side is at least  $\lambda$  (as  $\Delta(K_{S'} \cap K_{S''})$  is an  $(X, Y)$ -separator). Therefore,  $d(K_{S'} \cup K_{S''}) \leq \lambda$ . This means that  $d(K_{S'} \cup K_{S''})$  is also a minimum  $s - t$  cut, contradicting the maximality of  $S'$  and  $S''$ .

Next we show that for every important  $s - t$  separator  $S$ , we have  $K_{S^*} \subseteq K_S$ . Suppose this is not true for some  $S$ . We use submodularity again:

$$\underbrace{d(K_{S^*})}_{=\lambda} + d(K_S) \geq d(K_{S^*} \cup K_S) + \underbrace{d(K_{S^*} \cap K_S)}_{\geq \lambda}.$$

By definition,  $d(K_{S^*}) = \lambda$ , and  $\Delta(K_{S^*} \cap K_S)$  is an  $s - t$  separator, hence  $d(K_{S^*} \cap K_S) \geq \lambda$ . This means that  $d(K_{S^*} \cup K_S) \leq d(K_S)$ . However this contradicts the assumption that  $S$  is an important  $s - t$  separator:  $\Delta(K_{S^*} \cup K_S)$  is an  $s - t$  separator

not larger than  $S$ , but  $K_{S^*} \cup K_S$  is a proper superset of  $K_S$  (as  $K_{S^*}$  is not a subset of  $K_S$  by assumption).

We have shown that for every important separator  $S$ , the set  $K_S$  contains  $K_{S^*}$ . Let  $e \in S^*$  be an arbitrary edge of  $S^*$  (note that  $\lambda > 0$ , hence  $S^*$  is not empty) and let  $v$  be the endpoint of  $e$  not in  $K_{S^*}$ . An important  $s-t$  separator  $S$  either contains  $e$  or not. We will bound the contributions of these two types of separators to the sum.

Let  $S$  be an important  $s-t$  separator containing  $e$ . Then  $S \setminus e$  is an  $s-t$  separator in  $G \setminus e$ ; in fact, it is an important  $s-t$  separator of  $G \setminus e$ . Therefore, if  $\mathcal{S}'$  is the set of all important  $s-t$  separators in  $G \setminus e$ , then the set  $\mathcal{S}_1 = \{S' \cup e \mid S' \in \mathcal{S}'\}$  contains every important  $s-t$  separator of  $G$  containing  $e$ . Obviously, the size  $\lambda'$  of the minimum  $s-t$  separator in  $G \setminus e$  is at least  $\lambda - 1$ . The induction statement shows that  $\sum_{S' \in \mathcal{S}'} 4^{-|S'|} \leq 2^{-\lambda'} \leq 2^{-(\lambda-1)}$  and therefore  $\sum_{S \in \mathcal{S}_1} 4^{-|S|} = \sum_{S' \in \mathcal{S}'} 4^{-|S'|-1} \leq 2^{-(\lambda-1)}/4 = 2^{-\lambda}/2$ .

Let us consider now the important  $s-t$  separators not containing  $e$ . We have seen that  $K_{S^*} \subseteq K_S$  for every such  $s-t$  separator  $S$ . As  $e \notin S$ , even  $K_{S^*} \cup \{v\} \subseteq K_S$  is true. Let us obtain the graph  $G'$  from  $G$  by removing  $(K_{S^*} \cup \{v\}) \setminus \{s\}$  and making  $s$  adjacent to the neighborhood  $K_{S^*} \cup \{v\}$  in  $G$ . There is no  $s-t$  separator  $S$  of size  $\lambda$  in  $G'$ : such a set  $S$  would be an  $s-t$  separator of size  $\lambda$  in  $G$  as well, with  $K_{S^*} \cup \{v\} \subseteq K_S$ , contradicting the maximality of  $S^*$ . Thus the minimum size  $\lambda'$  of an  $s-t$  separator in  $G'$  is greater than  $\lambda$ . Let  $\mathcal{S}_2$  contain all the important  $s-t$  separators of  $G$  not containing  $e$ . We have seen that  $K_{S^*} \cup \{v\} \subseteq K_S$  for every separator  $S \in \mathcal{S}_2$ , thus such an  $S$  is an  $s-t$  separator of  $G'$  and in fact every such  $S$  is an important  $s-t$  separator in  $G'$  as well. Therefore, by the induction hypothesis,  $\sum_{S \in \mathcal{S}_2} 4^{-|S|} \leq 2^{-\lambda'} \leq 2^{-\lambda}/2$ .

Adding the bounds in the two cases, we get the required bound  $2^{-\lambda}$ .  $\square$

## A.2 Proof of Lemma 10

Recall that an  $(n, k, k^2)$ -splitter is a family of functions from  $[n]$  to  $[k^2]$  such that for any subset  $X \subseteq [n]$  with  $|X| = k$ , one of the functions in the family is injective on  $X$ . Naor, Schulman, and Srinivasan [21] gave an explicit construction of an  $(n, k, k^2)$ -splitter of size  $O(k^6 \log k \log n)$ .

*Proof (Proof of Lemma 10).* In the algorithm of Lemma 9, a random subset of a universe  $\mathcal{S}$  of size  $s = |\mathcal{S}| \leq 4^q \cdot |V(G)|$  is selected. If the  $(\mu, p, q)$ -CLUSTER problem has a solution  $C$ , then is a collection  $A \subseteq \mathcal{S}$  of  $a \leq q$  sets and a collection  $B \subseteq \mathcal{C}$  of  $b \leq q \cdot 4^q$  sets such that if every set in  $A$  is selected and no set in  $B$  is selected, then (1) and (2) hold. Instead of the selecting a random subset, we try every function  $f$  in an  $(s, a + b, (a + b)^2)$ -splitter family and every subset  $F \subseteq [(a + b)^2]$  of size  $a$  (there are  $\binom{(a+b)^2}{a+b} = 2^{O(q^2)}$  such sets  $F$ ). For a particular choice of  $f$  and  $F$ , we select those sets  $S \in \mathcal{S}$  into  $\mathcal{S}'$  for which  $f(S) \in F$ . The size of the splitter family is  $2^{O(q)} \log |V(G)|$  and the number of possibilities for  $F$  is  $2^{O(q^2)}$ . Therefore, we construct  $2^{O(q^2)} \cdot \log |V(G)|$  instances of the SATELLITE PROBLEM.

By the definition of the splitter, there will be a function  $f$  that is injective on  $A \cup B$ , and there is a subset  $F$  such that  $f(S) \in F$  for every set  $S$  in  $A$  and  $f(S) \notin F$  for every set  $S$  in  $B$ . For such an  $f$  and  $F$ , the selection will ensure that (1) and (2) hold. This

means that the constructed instance of the SATELLITE PROBLEM corresponding to  $f$  and  $F$  has a solution as well. Thus solving every constructed instance of the SATELLITE PROBLEM with the assumed  $f(q) \cdot n^{O(1)}$  time algorithm gives a  $2^{O(q^2)} \cdot f(q) \cdot n^{O(1)}$  algorithm for  $(\mu, p, q)$ -CLUSTER.  $\square$

### A.3 Proof of Lemma 12

*Proof (Proof of Lemma 12).* Consider the set  $C(S)$  for a subset  $S$  of  $\{1, \dots, i-1\}$ . We investigate what happens to  $\text{nonedge}(C(S))$  and  $d(C(S))$  when  $i$  is inserted into  $S$ . For  $\text{nonedge}$  we have the following equation.

$$\text{nonedge}(C(S \cup \{i\})) = \text{nonedge}(C(S)) + \text{nonedge}(V_i) + |C(S)| \cdot |V_i| - d(V_i) \quad (2)$$

Furthermore,  $d(C(S \cup \{i\})) = d(C(S)) - d(V_i)$ . Define  $T[i, j, k, \ell]$  to be true if there is a subset  $S$  of  $\{1, \dots, i\}$  such that  $|C| = j$ ,  $d(C(S)) = k$  and  $\text{nonedge}(C(S)) = \ell$ . If such a set  $S$  exists then either  $i \in S$  or  $i \notin S$ . Together with Equation 2 this yields the following recurrence for  $T[i, j, k, \ell]$ .

$$T[i, j, k, \ell] = T[i-1, j, k, \ell] \vee T[i-1, j-|V_i|, k+d(V_i), \ell-\text{nonedge}(V_i)-(j-|V_i|) \cdot |V_i|+d(V_i)] \quad (3)$$

The size of the table  $T$  is  $O(pn|E(G)||V(G)|)$  since  $i$  ranges from 1 to  $n$ ,  $j$  from 0 to  $|V(G)|$ ,  $k$  from 0 to  $|E(G)|$  and  $\ell$  from 0 to  $p$  as it makes no sense to add more sets to  $C$  after the threshold  $p$  of non-edges in  $C$  has been exceeded. We initialize the table to true in  $T[0, |V_0|, d(V_0), \text{nonedge}(V_0)]$  and false everywhere else. Then we compute the values of the table using Equation 3, treating every time we go out of bounds as a false entry. The algorithm returns true if there is an entry of  $T$  which is true for  $i = n$ ,  $k \leq q$  and  $\ell \leq p$ . The running time bound is immediate, while correctness follows from Equations 2 and 3.  $\square$

### A.4 Proof of Lemma 13

*Proof.* In Lemma 11, the set  $S$  described which sets  $V_i$  went into  $C$ . For this lemma, it is more convenient to let  $S$  describe the sets  $V_i$  which are *not* in  $C$ . Define  $U = \{1, \dots, n\}$ , the task is to find a subset  $S$  of  $U$  such that  $d(C(U \setminus S)) \leq q$  and  $\text{nondeg}(C(U \setminus S)) \leq p$ . We iterate over all possible values  $c \geq |V_0|$  of  $|C(S)|$ , and for each value of  $c$  we will only look for sets  $S$  such that  $|C(U \setminus S)| = c$ . This gives us the following advantage: for every vertex  $v \in V_i$  for  $i \geq 1$  if we choose to put  $V_i$  into  $C$  then  $v$  will have exactly  $c - d(v) - 1$  non-neighbors in  $C$ . Hence for any  $i$  such that  $V_i$  contains a vertex  $v$  with degree less than  $c - p - 1$  we know that  $i \in S$ . In other words, such a component  $V_i$  should not be in the solution  $C$ , hence we can remove  $V_i$  from the graph and decrease  $q$  by  $d(V_i)$  (as the edges  $\Delta(V_i) \subseteq \Delta(V_0)$  will leave  $C$  in any solution). Therefore, we can assume that every vertex  $v \notin V_0$  has degree at least  $c - p - 1$ .

From now on we only need to worry about  $d(C(U \setminus S)) \leq q$  and about the non-degrees of vertices in  $V_0$ . A vertex  $v \in V_0$  will have exactly  $c - 1 - d(v) + |\Delta(v) \cap \Delta(C(U \setminus S))|$  non-neighbors. In particular, we need to make sure that no vertex  $v \in V_0$  will have more than  $p + d(v) - c + 1$  neighbors outside of  $C(U \setminus S)$ . For every  $v \in V_0$

we define the capacity of  $v$  to be  $\text{cap}(v) = p + d(v) - c + 1$ . If any vertex has negative capacity, we discard the choice of  $c$ , as it is infeasible.

Every vertex  $v \in V_0$  gets  $\text{cap}(v)$  bins, and it distributes all the edges of  $\Delta(V_0) \cap \Delta(v)$  into the bins uniformly at random. Then every bin receives a color from  $\{1, \dots, q\}$  uniformly at random, and all edges in the bin receive the color of the bin. Since every edge of  $\Delta(V_0)$  is in exactly one bin, every edge of  $\Delta(V_0)$  gets one color.

We will look for a special kind a solution: a subset  $S \subseteq U$  such that  $|C(U \setminus S)| = c$  and every edge of  $\Delta(C(U \setminus S))$  has a different color. In particular, this implies  $d(C(U \setminus S)) \leq q$ . Since every vertex  $v \in V_0$  is incident to vertices of at most  $\text{cap}(v)$  different colors, this also ensures that for every  $v \in V_0$ , at most  $\text{cap}(v)$  edges incident to  $v$  leave  $C(S)$  and hence  $\text{nondeg}(C(S)) \leq p$ . Therefore, any set  $S$  satisfying these requirements indeed give a solution for the SATELLITE PROBLEM. On the other hand, if  $S$  is a subset of  $U$  such that  $|C(U \setminus S)| = c$ ,  $d(C(U \setminus S)) \leq q$  and  $\text{nondeg}(C(U \setminus S)) \leq p$ , then the probability that the edges of  $\Delta(C(U \setminus S))$  were colored with different colors can be bounded from below as follows.

For every vertex  $v \in V_0$ , define  $d(S, v) = |\Delta(C(U \setminus S)) \cap \Delta(v)|$ . The probability that the edges of  $\Delta(C(U \setminus S)) \cap \Delta(v)$  were distributed in different bins of  $v$  is

$$\prod_{i=0}^{d(S,v)-1} (\text{cap}(v) - i) / \text{cap}(v)^{d(S,v)} \geq e^{-d(S,v)}.$$

Thus the probability that all the edges of  $\Delta(C(U \setminus S))$  were distributed in different bins is at least

$$\exp\left(\sum_{v \in V_0} -d(S, v)\right) \geq e^{-q}.$$

The probability that these  $q$  bins were colored with different colors is  $\frac{q!}{q^q} \geq e^{-q}$ . Thus, the probability that the edges of  $\Delta(C(U \setminus S))$  receive different colors is at least  $e^{-2q}$ .

To complete the proof we need an algorithm that decides whether there is a set  $S \subseteq U$  such that  $|C(U \setminus S)| = c$ , and every edge of  $\Delta(C(U \setminus S))$  has a different color. For every  $i$  such that  $\Delta(V_i)$  contains more than one edge of the same color, we know that  $V_i \subseteq C$  and hence  $i \notin S$ . Let  $F_c$  be the set of *forced* indices, that is,  $F_c$  contains all  $i \in U$  such that  $\Delta(V_i)$  contains more than one edge of the same color.

For a vertex set  $X$ , define  $\text{colors}(X)$  to be the set of colors on the edges of  $\Delta(X)$ . For every  $0 \leq i \leq n$ ,  $0 \leq j \leq |V(G)|$  and  $R \subseteq \{1, \dots, q\}$ , we define  $T[i, j, R]$  to be true if there is a subset  $S$  of  $\{1, \dots, i\}$  such that  $|C(U \setminus S)| = j$ , all edges of  $\Delta(C(U \setminus S))$  have different colors and  $\text{colors}(C(U \setminus S)) \subseteq R$ .

Suppose such a set  $S$  exists, we have that either  $i \in S$  or  $i \notin S$ . If  $i \notin S$ , then  $S$  is a subset of  $\{1, \dots, i-1\}$  and hence  $T[i-1, j, R]$  is true. Observe that if  $i \in F_c$  then  $i$  must be in  $S$ . If on the other hand  $i \in S$ , then let  $S' = S \setminus \{i\}$ . In this case, we have that  $|C(U \setminus S')| = j + |V_i|$ , all edges of  $\Delta(C(U \setminus S'))$  have different colors and  $\text{colors}(C(U \setminus S')) \subseteq R \setminus \text{colors}(V_i)$ . This yields the following recurrence for  $T[i, j, R]$ .

$$T[i, j, R] = \begin{cases} T[i-1, j, R] & \text{if } i \in F_c \text{ or } \text{colors}(V_i) \not\subseteq R, \\ T[i-1, j, R] \vee T[i-1, j + |V_i|, R \setminus \text{colors}(V_i)] & \text{if } i \notin F_c. \end{cases} \quad (4)$$

Using Equation 4 we can find  $C$  in  $2^q|V(G)|^{O(1)}$  time. We initialize the table to true in  $T[0, |C(U)|, R]$  for all  $R \subseteq \{1, \dots, q\}$ . Then we use Recurrence 4 to fill the table. The algorithm returns true if  $T[n', c, R]$  is true for any subset  $R$  of  $\{1, \dots, q\}$ . The running time of the algorithm is dominated by the size of the table, which is  $2^q|V(G)|n$ . Correctness of the algorithm follows from the probability computations and Equation 4. This completes the proof of the lemma.  $\square$

### A.5 Proofs omitted from Section 4.1

The next lemma gives us a way to handle all large  $v$ -minimal sets  $C$  for (nonedge,  $p, q$ )-CLUSTER and (nondeg,  $p, q$ )-CLUSTER.

**Lemma 21.** *For any graph  $G$ , vertex  $v$  and integer  $p$ , there are at most  $O(2^p|V(G)|)$   $v$ -minimal sets  $C$  such that  $\text{nondeg}(C \cup \{v\}) \leq p$  and  $|C| \geq 3p$ . These sets can be listed in  $2^p|V(G)|^{O(1)}$  time.*

*Proof.* By Observation 15, any  $v$ -minimal set  $C$  contains a vertex  $u$  such that  $N[u] \setminus \{v\} \subseteq C$ . Thus, we go over every possibility for  $u$ , and we will enumerate all  $v$ -minimal sets  $C$  of size at least  $3p$  such that  $\text{nondeg}(C \cup \{v\}) \leq p$  and  $N[u] \setminus \{v\} \subseteq C$ . Notice that  $|C \setminus N[u]| \leq p$  since every vertex  $w \in C \setminus N[u]$  is a non-neighbour of  $u$ . Thus, if  $|C| \geq 3p$  then  $|N[u] \setminus \{v\}| \geq 2p$ . Hence, every vertex in  $C \setminus N[u]$  has at least  $|N[u] \setminus \{v\}| - p \geq p$  edges to  $N[u] \setminus \{v\}$ . Let  $S$  be the set of vertices in  $V(G) \setminus (N[u] \cup \{v\})$  which have at most  $p$  non-neighbours in  $N[u] \setminus \{v\}$ . If  $|S| \geq p+3$ , then no minimal set  $C$  satisfying the constraints and the requirement  $N[u] \setminus \{v\} \subseteq C$  can exist. To see this, suppose for contradiction that such a set  $C$  exists, then  $S \setminus C \geq 3$ . Since each vertex in  $S \setminus C$  has at most  $p$  non-neighbours in  $N[u] \setminus \{v\}$  it follows that

$$\bar{d}(C, v) \geq 3(|N[u] \setminus \{v\}| - p) \geq 3(|C| - 2p) \geq |C|$$

contradicting Observation 15. The second inequality holds since  $|N[u] \setminus \{v\}| \geq |C| - p$  and the third since  $|C| \geq 3p$ . Finally, since  $|S| \leq p+3$  and  $C \setminus (N[u] \setminus \{v\}) \subseteq S$  there are at most  $2^{p+3}$  possibilities sets  $C$  for every choice of  $u$ . Thus there are at most  $O(2^p|V(G)|)$  such sets, and they can be enumerated in  $2^p|V(G)|^{O(1)}$  time.  $\square$

Since every set  $C$  such that  $\text{nonedge}(C) \leq p$  satisfies  $\text{nondeg}(C) \leq p$ , we can use Lemma 21 to find out whether there is a set  $C$  such that  $v \notin C$ ,  $d(C \cup \{v\}) \leq q$ ,  $\text{nonedge}(C \cup \{v\}) \leq p$  and  $|C| \geq 3p$  in time  $2^p|V(G)|^{O(1)}$ . Thus we can concentrate on sets  $C$  of size at most  $3p$  for (nonedge,  $p, q$ )-CLUSTER and for (nondeg,  $p, q$ )-CLUSTER. For finding appropriate sets  $C$  of size at most  $3p$  for the two problems, we can give algorithms that are almost identical to the algorithm described in Lemma 19. Since the two algorithms are so similar, we describe both in one go.

**Lemma 22.** *There is a  $2^{O(p)}|V(G)|^{O(1)}$  time algorithm for the (nonedge,  $p, q$ )-CLUSTER and the (nondeg,  $p, q$ )-CLUSTER problem.*

*Proof.* We are given as input a graph  $G$  together with a vertex  $v$  and integers  $p$  and  $q$ . The task is to find a vertex set  $C$  such that  $d(\{v\} \cup C) \leq q$  and  $\text{nonedge}(\{v\} \cup C) \leq p$ ,

or  $\text{nondeg}(\{v\} \cup C) \leq p$ . It is sufficient to search for a  $v$ -minimal set  $C$  satisfying these properties. Using Lemma 21 we can check whether such a set of size at least  $3p$  exists in time  $2^p |V(G)|^{O(1)}$ . From now on we only need to consider sets  $C$  of size at most  $3p$ .

By Observation 17,  $C$  can be decomposed into  $C = S_1 \cup S_2 \dots \cup S_t$  such that  $S_i$  is a connected  $v$ -minimal set for every  $i$ ,  $S_i \cap S_j = \emptyset$  for every  $i \neq j$ , and no edge of  $G$  has one endpoint in  $S_i$  and the other in  $S_j$  for every  $i \neq j$ . Using Lemma 16 we list all connected  $v$ -minimal sets  $S_1 \dots S_n$  of size at most  $3p$ , and by Lemma 16  $n \leq 4^{3p} |V(G)|$ . For a set  $S$  and vertex  $v$ , define  $\text{nondeg}_v(S)$  to be the maximum number of non-edges to vertices in  $S$  over all vertices in  $S \setminus v$ . For a subset  $Z$  of  $\{1, \dots, n\}$  define  $C(Z) = \{v\} \cup \bigcup_{i \in Z} S_i$ . Now, let  $Z \subseteq \{1, \dots, n\}$  such that for every  $i, j \in Z$ , such that  $i \neq j$ ,  $S_i \cap S_j = \emptyset$ . We have that  $|C(Z)| = 1 + \sum_{i \in Z} |S_i|$  and that

$$d(C(Z)) \leq d(v) + \sum_{i \in Z} (\bar{d}(S_i, v) - d(S_i, v)) \quad (5)$$

$$\text{nonedge}(C(Z)) \leq |C(Z)| + \sum_{i \in Z} \text{nonedge}(S_i, v) - d(S_i, v) + \sum_{i \in Z} \sum_{i < j \in Z} |S_i| |S_j| \quad (6)$$

$$\text{nondeg}(C(Z)) \leq \max \left\{ \sum_{i \in Z} (|S_i| - d(S_i, v)), \max_{i \in Z} (\text{nondeg}_v(S_i \cup \{v\}) + \sum_{j \in Z \setminus \{i\}} |S_j|) \right\}$$

If there is no edge with one endpoint in  $S_i$  and the other in  $S_j$  for any  $i \neq j$ ,  $i \in Z$ ,  $j \in Z$  then the inequalities hold with equality. Our algorithm will select a  $Z$  such that  $C = \bigcup_{i \in Z} S_i$ . To ensure that the algorithm picks  $Z$  such that the sets  $S_i$  and  $S_j$  will be disjoint for every pair of distinct integers  $i, j \in Z$  we will use color coding. In particular, we construct a family  $\mathcal{F}$  of functions from  $V(G) \setminus \{v\}$  to  $\{1, \dots, 3p\}$  as described in Proposition 18. The family  $\mathcal{F}$  has size  $O(e^{3p} \cdot p^{O(\log p)} \cdot \log |V(G)|)$ .

For each function  $f \in \mathcal{F}$  we will think of the function as a coloring of  $V(G) \setminus \{v\}$  with colors from  $\{1, \dots, 3p\}$ . We will only look for a  $v$ -minimal set  $C$  whose vertices have different colors. This will ensure that any two sets  $S_i$  and  $S_j$  that we pick will be disjoint, and controls the total size of the set picked. If the input instance was a yes-instance then a solution set  $C$  exists, and the construction of  $\mathcal{F}$  ensures that there will be a function  $f \in \mathcal{F}$  which colors all vertices in  $C$  with different colors. When considering a particular coloring  $f$  we discard all sets from  $S_1, \dots, S_n$  which have two vertices of the same color, so from this point, without loss of generality, all sets in  $S_1, \dots, S_n$  have at most one vertex of each color. For a vertex set  $S$ , define  $\text{colors}(S)$  to be the set of colors occurring on vertices on  $G$ .

To solve  $(\text{nonedge}, p, q)$ -CLUSTER we define a table  $T_1$ . For every  $0 \leq i \leq n$ ,  $0 \leq j \leq |E(G)|$ ,  $0 \leq k \leq p$  and  $R \subseteq \{1, \dots, 3p\}$  we define  $T_1[i, j, k, S]$  to be true if there is a set  $Z \subseteq \{1, \dots, i\}$  such that all vertices of  $C(Z)$  have distinct colors,  $d(v) + \sum_{i \in Z} d(S_i, v) - \bar{d}(S_i, v) \leq j$ ,

$$|R| + \sum_{i \in Z} (\text{nonedge}(S_i, v) - d(S_i, v)) + \sum_{i \in Z} \sum_{i < j \in Z} |S_i| |S_j| \leq k,$$

and  $\text{colors}(C(Z)) = R$ . For (nondeg,  $p, q$ )-CLUSTER we define a table  $T_2$  in a similar manner. That is, for every  $0 \leq i \leq n$ ,  $0 \leq j \leq |E(G)|$ ,  $0 \leq k \leq p$ ,  $0 \leq \ell \leq p$  and  $R \subseteq \{1, \dots, 3p\}$  we define  $T_2[i, j, k, \ell, S]$  to be true if there is a set  $Z \subseteq \{1, \dots, i\}$  such that all vertices of  $C(Z)$  have distinct colors,  $d(v) + \sum_{i \in Z} d(S_i, v) - \bar{d}(S_i, v) \leq j$ ,

$$\max_{i \in Z} \left( \text{nondeg}_v(S_i \cup \{v\}) + \sum_{j \in Z \setminus \{i\}} |S_j| \right) \leq k \quad \sum_{i \in Z} (|S_i| - d(S_i, v)) \leq \ell$$

and  $\text{colors}(C(Z)) = R$ .

There is a  $v$ -minimal set  $C$  such that  $d(\{v\} \cup C) \leq q$ ,  $\text{nonedge}(\{v\} \cup C) \leq p$  and all vertices of  $C$  have different color if and only if  $T_1[n, j, k, R]$  is true for some  $j \leq q$ ,  $k \leq p$  and  $R \subseteq \{1, \dots, 3p\}$ . This follows directly from the definition of  $T_1$  and the fact that Equation 6 holds with equality when there is no edge with one endpoint in  $S_i$  and the other in  $S_j$  for some  $i \neq j$ ,  $i \in Z$ ,  $j \in Z$ . By an identical argument, there is a  $v$ -minimal set  $C$  such that  $d(\{v\} \cup C) \leq q$ ,  $\text{nondeg}(\{v\} \cup C) \leq p$  and all vertices of  $C$  have different color if and only if  $T_2[n, j, k, \ell, R]$  is true for some  $j \leq q$ ,  $k \leq p$ ,  $\ell \leq p$  and  $R \subseteq \{1, \dots, 3p\}$ . We can fill the tables  $T_1$  and  $T_2$  using the following recurrences.

$$T_1[i, j, k, R] = \begin{cases} T_1[i-1, j, k, R] & \text{if } \text{colors}(S_i) \not\subseteq R \\ T_1[i-1, j, k, R] \vee T_1[i-1, j', k', R'] & \text{otherwise} \end{cases}$$

$$T_2[i, j, k, \ell, R] = \begin{cases} T_2[i-1, j, k, \ell, R] & \text{if } \text{colors}(S_i) \not\subseteq R \\ T_2[i-1, j, k, \ell, R] & \text{if } |R| - |S_i| + \text{nondeg}_v(S_i \cup \{v\}) > k \\ T_2[i-1, j, k, \ell, R] \vee T_2[i-1, j', k', \ell', R'] & \text{otherwise} \end{cases}$$

where  $j' = j - d(S_i, v) + \bar{d}(S_i, v)$ ,  $k' = k - \text{nonedge}(S_i) - |S_i| + d(S_i, v) - |C_i|(|R| - |C_i|)$ ,  $k'_2 = k - |S_i|$ , and  $\ell' = \ell - |S_i| + d(S_i, v)$  and  $R' = R \setminus \text{colors}(S_i)$

The recurrences above are correct for the following reason. Let  $Z$  be a subset of  $Z \subseteq \{1, \dots, i-1\}$  such that all vertices of  $C(Z)$  have distinct colors,  $d(v) + \sum_{i \in Z} d(S_i, v) - \bar{d}(S_i, v) = j$ ,  $|R| + \sum_{i \in Z} \text{nonedge}(S_i, v) - d(S_i, v) + \sum_{i \in Z} \sum_{i < j \in Z} |S_i||S_j| = k$ , and  $\text{colors}(C(Z)) = R$ . Since every vertex in  $C(Z)$  has a different color, it follows that  $|C(Z)| = |R|$ . Inserting  $i$  into  $Z$  is only possible if  $\text{colors}(C(Z)) = R$ . In that case, when we insert  $i$  into  $Z$ ,  $d(v) + \sum_{p \in Z} d(S_p, v) - \bar{d}(S_p, v)$  increases by  $d(S_i, v) - \bar{d}(S_i, v)$ . The sum  $|R| + \sum_{i \in Z} \text{nonedge}(S_i, v) - d(S_i, v) + \sum_{i \in Z} \sum_{i < j \in Z} |S_i||S_j|$  increases by  $\text{nonedge}(S_i) + |S_i| - d(S_i, v) + |C_i|(|R| - |C_i|)$ . The expression  $\max_{p \in Z} (\text{nondeg}_v(S_p \cup \{v\}) + \sum_{j \in Z \setminus \{p\}} |S_j|)$  by  $|S_i|$  or to  $\text{nondeg}_v(S_i \cup \{v\}) + \sum_{j \in Z} |S_j|$ , whichever is the largest. The expression  $\sum_{p \in Z} (|S_p| - d(S_p, v))$  increases by  $|S_i| - d(S_i, v)$ . Finally, the set of colors used, will now be  $R \cup \text{colors}(S_i)$ .

We initialize the tables  $T_1$  and  $T_2$  as follows.  $T_1[0, j, k, \emptyset]$  is set to true for every  $j \geq d(v)$ ,  $k \geq 0$ .  $T_2[0, j, k, \ell, \emptyset]$  is set to true for every  $j \geq d(v)$ ,  $k \geq 0$ ,  $\ell \geq 0$ . Then we fill the tables using the recurrences above. The tables have size  $4^{3p}|V(G)|^{O(1)} \cdot 2^{3p}|V(G)|^{O(1)} = 8^{3p}|V(G)|^{O(1)}$  and can be filled in time proportional to their size. Hence the total running time for the algorithms is  $(8e)^{3p+o(p)}|V(G)|^{O(1)}$ .  $\square$

## A.6 Proofs omitted from Section 4.2

To prove the statements in Theorem 20, we reduce from the  $k$ -CLIQUE problem in  $d$ -regular graphs. Here we are given a graph  $G$  in which all vertices have degree  $d$ , and an integer  $k$ . The task is to find a clique  $C$  in  $G$  of size at least  $k$ . This problem is NP-complete and W[1]-complete when parameterized by  $k$  [18]. Given a  $d$ -regular graph  $G$  with  $n$  vertices and  $m$  edges and an integer  $k$ , we construct a weighted graph  $G'$  by adding a vertex  $v$  incident to all vertices of  $G$  with  $m + 1$  parallel edges to each vertex of  $G$ .

**Lemma 23.** *There is a clique  $C$  in  $G'$  such that  $v \in C$  and  $d(C) \leq (n - k)(m + 1) + k(d - k)$  if and only if  $G$  contains a clique of size  $k$ .*

*Proof.* In the forward direction, suppose there is a  $C$  in  $G'$  such that  $v \in C$  and  $d(C) \leq (n - k)(m + 1) + k(d - k)$ . Then  $C' = C \setminus \{v\}$  is a clique in  $G$ . It suffices to argue that  $|C'| \geq k$ . Suppose not. Then the number of edges leaving  $C$  is at least  $(n - k + 1)(m + 1) > (n - k)(m + 1) + k(d - k)$ , a contradiction.

In the backward direction, suppose  $G$  contains a clique of size  $k$ . Then  $C' = C \cup \{v\}$  is a clique in  $G'$  and the number of edges leaving  $C'$  is exactly  $(n - k)(m + 1) + k(d - k)$ .  $\square$

Notice that  $G'$  can be partitioned into cliques with at most  $q = (n - k)(m + 1) + k(d - k)$  edges leaving each clique if and only if there is a clique  $C$  in  $G'$  such that  $v \in C$  and  $d(C) \leq q$ . If such a clique  $C'$  exists then there exists such a clique of size  $k + 1$ . Hence, by Lemma 23, if (nonedge,  $0, q$ )-PARTITION or (nondeg,  $0, q$ )-PARTITION can be solved in polynomial time on graphs with parallel edges, then CLIQUE in  $d$ -regular graphs can. Similarly, if (size,  $p, q$ )-PARTITION can be solved in  $f(p)n^c$  time on graphs with parallel edges then CLIQUE in  $d$ -regular graphs can be solved in  $f(k + 1)n^c$  time. This proves Theorem 20.