Designing FPT algorithms for cut problems using randomized contractions

Chitnis, Rajesh; Cygan, Marek; Hajiaghayi, Mohammadtaghi; Pilipczuk, Marcin; Pilipczuk, Michalø

DOI:
10.1137/15M1032077

SIAM J. COMPUT.
Vol. 45, No. 4, pp. 1171–1229

DESIGNING FPT ALGORITHMS FOR CUT PROBLEMS USING RANDOMIZED CONTRACTIONS*

RAJESH CHITNIS†, MAREK CYGAN‡, MOHAMMADTAGHI HAJIAGHAYI†, MARCIN PILIPCZUK‡, AND MICHAŁ PILIPCZUK‡

Abstract. We introduce a new technique for designing fixed-parameter algorithms for cut problems, called *randomized contractions*. We apply our framework to obtain the first fixed-parameter algorithms (FPT algorithms) with exponential speed up for the STEINER CUT and NODE MULTIWAY CUT-UNCUT problems. We prove that the parameterized version of the UNIQUE LABEL COVER problem, which is the base of the UNIQUE GAMES CONJECTURE, can be solved in $2^{O(k^2 \log |\Sigma|)} n^4 \log n$ deterministic time (even in the stronger, vertex-deletion variant), where k is the number of unsatisfied edges and $|\Sigma|$ is the size of the alphabet. As a consequence, we show that one can in polynomial time solve instances of UNIQUE GAMES where the number of edges allowed not to be satisfied is upper bounded by $O(\sqrt{\log n})$ to optimality, which improves over the trivial $O(1)$ upper bound. We prove that the STEINER CUT problem can be solved in $2^{O(k^2 \log k)} n^4 \log n$ deterministic time and $\tilde{O}(2^{O(k^2 \log k)} n^2)$ randomized time, where k is the size of the cutset. This result improves the double exponential running time of the recent work of Kawarabayashi and Thorup presented at FOCS'11. We show how to combine considering "cut" and "uncut" constraints at the same time. More precisely, we define a robust problem, NODE MULTIWAY CUT-UNCUT, that can serve as an abstraction of introducing uncut constraints and show that it admits an algorithm running in $2^{O(k^2 \log k)} n^4 \log n$ deterministic time, where k is the size of the cutset. To the best of our knowledge, the only known way of tackling uncut constraints was via the approach of Marx, O'Sullivan, and Razgon [*ACM Trans. Algorithms*, 9 (2013), 30], which yields algorithms with double exponential running time. An interesting aspect of our algorithms is that they can handle positive real weights.

Key words. fixed-parameter tractability, randomized contractions, graph separations problems, unique label cover

AMS subject classifications. 68Q25, 68W40

DOI. 10.1137/15M1032077

1. Introduction.

Graph cut problems is a class of problems where, given a graph, one is asked to find a *cutset* of minimum size whose removal makes the graph satisfy a global separation property. The motivation for studying graph cut problems stems from the fundamental minimum cut problem, where the goal is to separate two terminals from each other by removing the least possible number of vertices or edges, depending on the variant. Even though the minimum cut problem can be solved in

*Received by the editors July 22, 2015; accepted for publication (in revised form) April 19, 2016; published electronically July 6, 2016. A preliminary version of this work was presented at FOCS 2012.

http://www.siam.org/journals/sicomp/45-4/M103207.html

†Department of Computer Science, University of Maryland at College Park, College Park, MD 20742 (rchitnis@cs.umd.edu, hajiagha@cs.umd.edu). These authors were supported in part by NSF CAREER award 1053605, ONR YIP award N000141110662, DARPA/AFRL award FA8650-11-1-7162, and a University of Maryland Research and Scholarship Award (RASA).

‡Institute of Informatics, University of Warsaw, Poland (cygan@mimuw.edu.pl, malcin@mimuw.edu.pl, michal.pilipczuk@mimuw.edu.pl). The second author was supported in part by NSF CAREER award 1053605, ONR YIP award N000141110662, DARPA/AFRL award FA8650-11-1-7162, a University of Maryland Research and Scholarship Award (RASA), and NCN grant N206567140, Foundation for Polish Science. The fourth author was partially supported by NCN grant N206567140 and Foundation for Polish Science. The research of the fifth author, leading to these results, was done when the author was affiliated with the University of Bergen, Norway, and received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013)/ERC grant agreement 267959.

polynomial time, many of its natural generalizations become NP-hard. Moreover, many problems, whose classical definitions do not resemble cut formulations, after choosing an appropriate combinatorial viewpoint show deep links with finding minimum separators; the most important examples are Feedback Vertex Set and Odd Cycle Transversal.

Therefore, circumventing NP-hardness of fundamental graph cut problems, like Multiway Cut (given a graph with a set of terminals, separate the terminals from each other using minimum size cutset) or Multicut (given a graph with a set of terminal pairs, separate terminals in the pairs using minimum size cutset), became an important algorithmic challenge. It is then no surprise that graph cut problems were studied intensively from the point of view of approximation; cf. [1, 4, 9, 22, 23, 29, 27, 36, 48, 52, 56].

In this paper we address a different paradigm of tackling NP-hard problems, that is, *fixed-parameter tractability* (FPT). Recall that in the parameterized complexity setting an instance of the problem comes with an additional integer k, called the *parameter*, which intuitively measures the hardness of the instance. The goal is to devise an algorithm solving the problem with running time of form $f(k)n^c$, where f is some computable function and c is a fixed constant. In other words, for every fixed parameter the algorithm has to work in polynomial time, where the degree of the polynomial is independent of the parameter. Algorithms with such a running time guarantee are called *fixed-parameter* algorithms, and if a problem admits one, then we say that it is *fixed-parameter tractable*. For a more detailed introduction to fixed-parameter tractability we address the interested reader to the recent monographs [15, 21].

Graph separation problems in the context of parameterized complexity were probably first considered in the seminal work of Marx [44]. Marx established fixed-parameterized complexity of Multiway Cut parameterized by the size of the cutset and Multicut parameterized by the size of the cutset plus the number of terminal pairs. The main tool introduced by Marx is the notion of an *important separator*, which later turned out to be the core ingredient of parameterized algorithms for, e.g., Directed Feedback Vertex Set [11] or Almost 2-SAT [54]. In the last decade, the graph separation problems have become one of the most intensively studied subareas of parameterized complexity, leading to the development of various interesting techniques, such as shadow removal [46] and its generalizations to directed graphs [14], treewidth reduction [45], and branching guided by an LP or (k-)submodular CSP relaxation [18, 57].

We introduce a new technique, called *randomized contractions*, of constructing fixed-parameter algorithms for graph cut problems. In this introduction, we first give an overview of this technique and our results, and then provide a discussion and comparison with other known techniques.

1.1. Our techniques. On a high level, the technique of *randomized contractions* is based on a WIN/WIN approach, introduced by Kawarabayashi and Thorup [37] and also used by Grohe and Marx in their algorithm to test the topological minor relation [31]. The WIN/WIN approach can be described as follows: either we find a well-balanced separation of small order, whose one side can be simplified by a recursive call, or the graph admits a highly connected structure, which can be used to identify the solution. The main novelty of this paper is the way these steps are executed: we show that a well-balanced separation can be easily and efficiently found using the color coding technique introduced by Alon, Yuster, and Zwick [2], and the color coding technique also greatly helps in exhibiting the solution in the presence of the

high-connected structure.

Recall that the main idea of the color coding technique, originally introduced to solve some special cases of the Subgraph Isomorphism problem, is to color the graph at random and ensure that with high probability the solution gets sufficiently highlighted to be recognizable quickly. It has now become a classical tool in the parameterized complexity toolbox. At the heart of our results lies an observation that it also can be used to highlight either a well-balanced separation or a structure of the solution in a highly connected graph. Our usage of the color coding technique, especially in the search for a well-balanced separation, resembles the algorithm of Karger [35] that finds a minimum cut in a graph in near-linear time by contracting random edges; this inspiration gave the name to our technique.

Although the intuition behind color coding is of probabilistic nature, the algorithms obtained using this approach can be derandomized using the technique of *splitters* of Naor, Schulman, and Srinivasan [49]. In fact, we find it more convenient to present our algorithms already in the derandomized version, so in spite of the name of the technique there will be no randomization at all; instead we use the following abstraction.

Lemma 1. *Given a set U of size n, and integers $0 \leq a, b \leq n$, one can in time $2^{O(\min(a,b)\log(a+b))}n \log n$ construct a family $\mathcal{F}$ of at most $2^{O(\min(a,b)\log(a+b))} \log n$ subsets of U, such that the following holds: for any sets $A, B \subseteq U$, $A \cap B = \emptyset$, $|A| \leq a$, $|B| \leq b$, there exists a set $S \in \mathcal{F}$ with $A \subseteq S$ and $B \cap S = \emptyset$.*

Our approach is most natural for edge-deletion problems; however, we can also extend it to node-deletion variants. For the node-deletion problems, however, the situation is more complicated and we need to define two kinds of separations. Only when the graph does not have both kinds of separations do we get enough structure to solve the problem with other methods. Moreover, one needs to be much more careful in this final case, as we obtain much weaker structural properties of the graph.

1.2. Our results. We use the technique of randomized contractions to provide the first fixed-parameter algorithm solving an important problem in parameterized complexity, and moreover we show how our approach can be applied to reduce the time complexity of the best known algorithms from double exponential to single exponential for some problems already known to be FPT.

1.2.1. Unique Label Cover. In the Unique Label Cover problem we are given an undirected graph G, where each edge $uv = e \in E(G)$ is associated with a permutation $\psi_{e,u}$ of a constant size alphabet Σ. The goal is to construct a labeling $\Psi : V(G) \to \Sigma$ maximizing the number of satisfied edge constraints, that is, edges for which $(\Psi(u), \Psi(v)) \in \psi_{uv,u}$ holds. At first glance Unique Label Cover does not seem related to the previously mentioned cut problems; however, it is not hard to show that the node-deletion version of Unique Label Cover is a generalization of the Group Feedback Vertex Set problem [33], and hence of Odd Cycle Transversal, Feedback Vertex Set, as well as Multiway Cut.

The optimization version of Unique Label Cover is the subject of the very extensively studied Unique Games Conjecture, proposed by Khot [38] in 2002, which is used as a hardness assumption for showing several tight inapproximability results. The Unique Games Conjecture states that for every sufficiently small $\varepsilon, \delta > 0$, there exists an alphabet size $|\Sigma|(\varepsilon, \delta)$, such that given an instance $(G, \Sigma, (\psi_{e,v})_{e \in E(G), v \in e})$ it is NP-hard to distinguish between the cases $|OPT| \leq \delta|E(G)|$ and $|OPT| \geq (1 - \varepsilon)|E(G)|$. In 2010 Arora, Barak, and Steurer [3] pre-

sented a breakthrough subexponential time algorithm, which in $2^{O(|\Sigma|n^{\varepsilon})}$ running time satisfies $(1-\varepsilon)|E(G)|$ edge constraints, assuming the given instance satisfies $|OPT| \geq (1-\varepsilon^c)|E(G)|$. We refer the reader to a recent survey of Khot [39] for more detailed discussion on the Unique Games Conjecture.

Since all the edge constraints are permutations, fixing a label for one vertex gives only one possibility for each of its neighbors, assuming we want to satisfy all the edges. For this reason we can verify in polynomial time whether $OPT = |E(G)|$. In this paper we show that we can efficiently solve the Unique Label Cover problem, assuming almost all the edges are to be satisfied. In particular, we design a fixed parameter algorithm for Node Unique Label Cover, which is a generalization of Edge Unique Label Cover.

Node Unique Label Cover
Input: An undirected graph G, a finite alphabet Σ of size s, an integer k, and for each edge $e \in E(G)$ and each of its endpoints v a permutation $\psi_{e,v}$ of Σ, such that if $e = uv$, then $\psi_{e,u} = \psi_{e,v}^{-1}$.
Question: Does there exist a set $X \subseteq V(G)$ of size at most k and a function $\Psi : V(G) \setminus X \to \Sigma$ such that for any $uv \in E(G \setminus X)$ we have $(\Psi(u), \Psi(v)) \in \psi_{uv,u}$?

Theorem 2. *There is an $O(2^{O(k^2 \log s)} n^4 \log n)$ time algorithm solving* Node Unique Label Cover.

To justify our parameterization, we would like to note that there is a long line of polynomial time approximation algorithms designed for instances of Unique Label Cover; currently the best is by Charikar, Makarychev, and Makarychev [8], working under the assumption $|OPT| \geq (1-\varepsilon)|E(G)|$, and where the alphabet is of constant size. Therefore, it is reasonable to assume that only a small number of constraints is not going to be satisfied. Our results imply that one can in polynomial time verify whether it is possible to satisfy $|E(G)| - O(\sqrt{\log n})$ constraints; consequently, we extend the range of instances that can be solved optimally in polynomial time.

Finally, we show that the dependence on the alphabet size in Theorem 2 is probably necessary, since the problem parameterized by the cutsize only is $W[1]$-hard. Hence, the existence of an algorithm parameterized by the cutsize only would cause $FPT = W[1]$, which is considered implausible. For a more detailed introduction to the hierarchy of parameterized problems and consequences of its collapse, we refer to the books of Downey and Fellows [20] or of Flum and Grohe [25]. We consider this result an interesting counterposition of the parameterized status of Group Feedback Vertex Set [17], which is FPT even when the group size is not a parameter.

Theorem 3. *The* Edge Unique Label Cover *problem, and consequently* Node Unique Label Cover, *is $W[1]$-hard when parameterized by k only.*

1.2.2. Steiner Cut. Next, we address a robust generalization of both k-Way Cut and Multiway Cut problems, namely, the Steiner Cut problem.

Steiner Cut
Input: A graph G, a set of terminals $T \subseteq V(G)$, and integers s and k.
Question: Does there exist a set X of at most k edges of G, such that in $G \setminus X$ at least s connected components contain at least one terminal?

Using our technique we present an FPT algorithm working in $O(2^{O(k^2 \log k)} n^4 \log n)$, where the polynomial factor can be improved to $\tilde{O}(n^2)$ at the cost of our algorithm being randomized. These results improve the double exponential time complexity of the recent algorithm of Kawarabayashi and Thorup [37].[1]

THEOREM 4. *There is a deterministic* $O(2^{O(k^2 \log k)} n^4 \log n)$ *and randomized* $\tilde{O}(2^{O(k^2 \log s)} n^2)$ *running time algorithm solving* STEINER CUT.

1.2.3. Connectivity constraints. We define the following problem as an abstraction of introducing "cut" and "uncut" constraints at the same time.

NODE MULTIWAY CUT-UNCUT (N-MWCU)
Input: A graph G together with a set of terminals $T \subseteq V(G)$, an equivalence relation $\mathcal{R}$ on the set T, and an integer k.
Question: Does there exist a set $X \subseteq V(G) \setminus T$ of at most k nonterminals such that for any $u, v \in T$, the vertices u and v belong to the same connected component of $G \setminus X$ iff $(u, v) \in \mathcal{R}$?

Fixed-parameter tractability of this problem can be derived from the framework of Marx, O'Sullivan, and Razgon [45], complemented with a reduction of the number of equivalence classes of $\mathcal{R}$ in the flavor of the reduction for MULTIWAY CUT of Razgon [53]. However, the dependence on k of the running time is double exponential. Using our framework we show the following.

THEOREM 5. *There is an* $O(2^{O(k^2 \log k)} n^4 \log n)$ *time algorithm solving N-MWCU.*

1.2.4. Weights. As mentioned in the abstract, our approach generalizes well to the weighted setting, which is not the case for many other techniques in parameterized complexity such as important separators. As the level of technical details in all our algorithms is high, we prove Theorems 2, 4, and 5 in the unweighted case (i.e., as they are stated in the introduction). Then, in section 8, we discuss extensions to the weighted setting.

1.2.5. Subsequent usages and extensions. We would like to mention here a few applications and extensions of our techniques, developed after the extended abstract of our work was published [12].

First, the technique turned out to be useful in a number of other problems. Lokshtanov during Dagstuhl Seminar 14071 (February 2014) noticed that our technique immediately gives fixed-parameter tractability the VECTOR CONNECTIVITY problem, parameterized by the cutset size only, solving an open problem posed by Milanič; we refer to the recent work of Kratsch and Sorge [41] for the problem definition and a discussion of recent developments. Bringmann et al. [7], in their study of different parameterizations of STEINER MULTICUT, noticed that one can use randomized contractions to obtain an FPT algorithm for one of their most natural parameterizations.

Finally, a subset of the current authors together with Lokshtanov and Saurabh [16] developed a way to replace the recursive scheme in our approach with a static tree decomposition, where every adhesion has bounded size and every bag has properties similar to those dubbed "highly connected" in the description above. This improvement has led to an FPT algorithm for MINIMUM BISECTION.

[1] In [37] the authors solve the k-WAY CUT problem; however, a straightforward generalization of their algorithm solves the STEINER CUT problem as well.

1.3. Discussion of related work.

Important separators. Perhaps the most fruitful consequence of the early work of Marx [44] was the introduction of the concept of an *important separator*. Important separators proved to be a robust tool that enable us to capture the bounded-in-parameter character of the family of reasonable cutsets. They also can be naturally extended to the directed setting. This basic technique has found numerous applications [10, 11, 14, 19, 32, 42, 44, 54].

The important separators technique is based on greedy arguments, which unfortunately makes this approach work only in restricted settings. Consider, for instance, the "uncut" constraint present in the N-MWCU problem, i.e., we look for a cutset that separates some pairs of terminals but is required *not* to separate some other pairs. Any greedy choice of the farthest possible cutset, which is precisely the idea behind the notion of an important separator, can spoil the delicate requirements of the existence of some paths.

Furthermore, the proof of the core property of important separators—the bound on their number expresses in the parameter only—relies on amortization by the increase of the cost of the separation, which makes the argument work only in the unweighted setting (or with small integer weights). It is unclear whether this notion can lead to parameterized algorithms in the setting with arbitrary (positive) real weights.

Shadow removal. The fixed-parameter tractability of Multicut parameterized by the cutsize only, after resisting attacks as a long-standing open problem, was finally resolved in 2011 by Marx and Razgon [46] and, independently, by Bousquet, Daligault, and Thomassé [6]. The most important contribution of the work of Marx and Razgon [46] was the introduction of the shadow removal technique, an intricate blend of the important separators with the color coding technique. In some problems (e.g., Multicut) one can argue that a greedy step, in the sense of important separators, is possible, but one cannot apply it directly, as one does not know one side of the separation. The color coding technique is used to highlight possible application places.

A subset of the current authors together with Marx [14] showed that after a delicate transfer of the shadow removal technique to directed graphs, it almost immediately yields an FPT algorithm for Multiway Cut in directed graphs. Further usages include [13, 40, 42].

On a high level, one could say that the shadow removal technique extends the applicability of important separators and is used to obtain additional properties of the cutset we are looking for. In some sense it is perpendicular to randomized contractions: On one hand, its applicability is limited due to the need of some greedy reasoning to apply important separators. On the other hand, shadow removal seems crucial for most of its applications, especially in directed graphs; in particular, we are unable to handle these applications using randomized contractions.

Treewidth reduction. The treewidth reduction technique, developed by Marx, O'Sullivan, and Razgon [45], is probably the closest, in terms of the scope of applicability, to randomized contractions. It essentially states that in an undirected graph G with two terminals s and t, all inclusionwise minimal cuts between s and t of size at most k live in a part of G of treewidth bounded exponentially in k. The result is robust in the sense that it allows one to include a bounded number of terminal pairs to separate.

This clean structural result allows one to bypass the limitations of important separators: similarly as randomized contractions, it does not require any greedy step (thus

handling, e.g., the "uncut" contraints) and it easily handles weighted variants. The most natural problems that can be handled using the treewidth reduction technique, like N-MWCU, usually can be also solved using randomized contractions.

However, we note that there are two shortfalls of treewidth reduction, as compared to randomized contractions. First, it inherently leads to double-exponential dependency on the parameter: the bound on the treewidth of the "small cut part" of G is necessarily exponential, and on top of that one uses a dynamic programming algorithm whose running time almost always depends at least exponentially on this treewidth. Second, it requires one to specify a bounded number of terminals to start with; hence it is unclear how to use it, e.g., for the Unique Label Cover problem.

It should be noted that algorithms using the treewidth reduction technique, despite their double-exponential dependency on the parameter, are usually conceptually much simpler and cleaner than their counterparts obtained using randomized contractions. This is particularly visible in the case of Steiner Multicut [7].

Branching guided by LP relaxations. A subset of the current authors, together with Wojtaszczyk [18], showed that one can use very strong structural properties of the LP relaxations of Vertex Cover and Multiway Cut to develop efficient branching algorithms for these problems, parameterized by the gap above the optimum value of the LP relaxation. Narayanaswamy et al. [50] observed that in the case of Vertex Cover, one can apply known reduction rules to improve running time even further. In this manner, quite unexpectedly, they obtained an improvement upon the classic $O(3^k nm)$-time algorithm for Odd Cycle Transversal [55]. The currently fastest algorithm in this line is due to Lokshtanov et al. [43].

Wahlström [57] observed that instead of the very inflexible LP relaxations, one could use (k-)submodular relaxation to a valued CSP problem, obtaining surprisingly efficient algorithms for a number of problems, including the $|\Sigma|^{2k} n^{O(1)}$-time algorithm Unique Label Cover. Subsequently, the dependency on the input size has been improved to linear [34]. We note that these works [34, 57] are subsequent to our work.

The above line of research gave a number of surprisingly efficient algorithms: the running time is usually single-exponential in the cutsize, and the techniques of [34] usually give good dependency on the input size. On the other hand, to apply them one needs to find a relaxation with strong properties (a k-submodular one in most cases), which is unknown, e.g., for Multicut or N-MWCU.

Limitations of randomized contractions. This discussion exhibits three limitations of the randomized contractions technique.

First, we do not know how to apply the randomized contractions technique to the Multicut problem without any bound on the number of terminals; recall that the algorithm of Marx and Razgon [46] makes use of important separators and shadow removal. This is mostly due to the fact that our technique, in the recursive step, needs a bound on the number of possible behaviors on a small cutset, similarly as is needed to develop a dynamic programming algorithm on graphs of bounded treewidth, or to apply the protrusion machinery [5, 26]. Note that Multicut, in the edge-deletion setting, is NP-hard on trees [28].

Second, our technique is inherently tailored to undirected graphs, whereas both important separators and shadow removal are well-understood on directed graphs as well. It is an interesting question whether one can obtain a convenient structural description of bounded size cuts in directed graphs, in the spirit of the treewidth reduction technique for undirected graphs [45]. A very recent work of one of the authors and Wahlström [51] showed that one cannot hope for bounded treewidth of

the underlying undirected graph, but *directed* treewidth can be bounded. However, the latter is probably insufficient for many algorithmic applications, as [51] proved also $W[1]$-hardness of Multicut in directed graphs with only four terminal pairs.

Third, in the case of Unique Label Cover our technique gives suboptimal running time, in terms of both the dependency on the parameter and the input size [34]. In our approach, we require that q, the minimum size of a side in a well-balanced separation, is greater than the number of possible behaviors of the solution on a cutset (which is usually exponential in the size of the cutset), and subsequent applications of the color coding technique introduce term $q^k = 2^{\Omega(k^2)}$ to the running time bound. Furthermore, the recursion scheme, together with multiple needs of finding small cuts, blows up the polynomial factor to quartic. We conjecture that in the other studied problems, as well as in the case of Minumum Bisection [16], it is possible to decrease both factors of the running time bound significantly, but possibly using very different techniques.

1.4. Organization of the paper. We start with an informal illustration of our technique in section 2, using the example of the edge-deletion version of the Unique Label Cover problem. We follow the illustration with some formal generic definitions and preliminary results in section 3. In sections 4, 5, and 6 we consider Node Unique Label Cover, Steiner Cut, and N-MWCU, respectively, proving Theorems 2, 4, and 5. Section 7 contains a reduction showing $W[1]$-hardness of the Edge Unique Label Cover problem, when parameterized by the size of the cutset only. Finally, in section 8 we discuss extensions of our framework to weighted graphs. As the introduction included an extensive discussion of related work and possible extensions, we skip the conclusions section.

2. Illustration. In this section we present the outline of the technique, illustrating it with a running example of the Edge Unique Label Cover problem. Since this section serves as an introduction and illustration, the arguments here are mostly informal. Note that a more general problem, Node Unique Label Cover, is formally proven to be fixed-parameter tractable in section 4.

Edge Unique Label Cover	**Parameter:** $k + s$
Input: An undirected (multi)graph G, a finite alphabet Σ of size s, an integer k, and for each edge $e \in E(G)$ and each of its endpoints v a permutation $\psi_{e,v}$ of Σ, such that if $e = uv$, then $\psi_{e,u} = \psi_{e,v}^{-1}$.	
Question: Does there exist a set $X \subseteq E(G)$ of size at most k and a function $\Psi : V(G) \to \Sigma$ such that for any $uv \in E(G) \setminus X$ we have $(\Psi(u), \Psi(v)) \in \psi_{uv,u}$?	

The permutations $\psi_{e,u}$ are called *constraints*, the function Ψ is called a *labeling*, and the set X is the *deletion set*.

As we consider the edge-deletion version, we use edge cuts throughout this section. However, as our general framework also can be applied to node-deletion problems, we comment throughout the description where additional argumentation is needed in the node-deletion setting.

We assume that the graph given in the input is connected, as it is easy to reduce the problem to considering each connected component separately. This is true for all the considered problems. Connectivity of the graph will be maintained during the whole course of the algorithm. Note that this means that the graph after excluding X can have at most $k + 1$ connected components.

The algorithm, at the very high level, closely follows the approach of

Kawarabayashi and Thorup [37]. We distinguish two cases: either the graph has a somewhat balanced separator or it is highly connected in the following sense: any cut of bounded size can separate only a very small part of the graph. More formally, we use the following notion of *good edge separation*.

DEFINITION 6. *Let G be a connected graph. A partition (V_1, V_2) of $V(G)$ is called a* (q, k)-good edge separation *if*

- $|V_1|, |V_2| > q$;
- $|\delta(V_1, V_2)| \leq k$, *where* $\delta(V_1, V_2)$ *is the set of edges with one endpoint in* V_1 *and second endpoint in* V_2;
- $G[V_1]$ *and* $G[V_2]$ *are connected.*

In the first phase of the algorithm, named *recursive understanding*, we iteratively find a good edge separation and reduce one of its sides up to the size bounded by a function of the parameter. We use the lower bound on the number of vertices of either side to ensure that we indeed make some simplification. The applied reduction step needs introduction of a more general problem, in which, intuitively, we have to prepare for every possible behavior on a bounded number of distinguished vertices of the graph, called *border terminals*.

When no good edge separation can be found, by Menger's theorem we know that between every two disjoint connected subgraphs of size larger than q we can find $k + 1$ edge-disjoint paths. Then we proceed to the second phase, named, the *high connectivity phase*, where we exploit this highly connected structure to identify the solution.

While the structure of the first phase is the same as in [37], our work differs in two important aspects. First, using Lemma 1 we show a simple efficient way to find a balanced separator to recurse. Second, we show a general methodology for how to apply Lemma 1 again for the second, high connectivity phase, to highlight important parts of the graph and find the solution efficiently.

2.1. Recursive understanding. First, we show a simple way to find a good edge separation in the graph. A full proof of the following lemma can be found in section 3.

LEMMA 7. *There exists a deterministic algorithm that, given an undirected, connected graph G on n vertices along with integers q and k, in time $O(2^{O(\min(q,k)\log(q+k))} n^3 \log n)$ either finds a (q, k)-good edge separation or correctly concludes that no such separation exists.*

Proof. Consider a family $\mathcal{F}$ obtained via Lemma 1 for the universe $U = E(G)$ and integers $a = 2q$ and $b = k$. Let (V_1, V_2) be a good separation in G and, for $i = 1, 2$, let T_i be any tree with q edges that is a subgraph of $G[V_i]$. By the properties of $\mathcal{F}$, there exists $S \in \mathcal{F}$ such that $E(T_1), E(T_2) \subseteq S$, but $S \cap E(V_1, V_2) = \emptyset$. Consider a (multi)graph G_S obtained from G by contracting the edges of S (we preserve multiple edges in the contraction process), and let $v \in V(G_S)$ be called *heavy* if more than q vertices of G were contracted onto it. It is easy to see that the good separation (V_1, V_2) corresponds to a cut between two heavy vertices in G_S of size at most k; moreover, any such cut yields a good separation in G. Such a desired cut can be found in polynomial time; the claimed running time follows if we first apply the sparsifying technique of Nagamochi and Ibaraki [47] and then the classical algorithm of Ford and Fulkerson to find a minimum cut between each pair of heavy vertices. We note that, using instead a variant of the classical Karger's algorithm for minimum cut [35], the problem can be solved in $\tilde{O}(2^{O(\min(q,k)\log(q+k))}(|V(G)| + |E(G)|))$ time at the cost of being randomized. ∎

The general methodology of the proof of Lemma 7—to use color coding to pick a set of undeletable edges that is disjoint from the solution, but highlights it—is the main engine of our work. We will see this idea exploited much more deeply in the high connectivity phase.

Having found a good edge separation we can proceed to simplification of one of the sides. To this end, following [37], we consider a more general problem, where the input graph is equipped with a set of *border terminals* T_b, whose number is bounded by a function of the budget for edge deletions. Intuitively, each considered instance of the border problem corresponds to solving some small part of the graph, which can be adjacent to the remaining part only via a small boundary—the border terminals. Our goal in the border version is, for every fixed behavior on the border terminals, to find some minimum size solution or to conclude that the size of the minimum solution exceeds the given budget. Of course, the definition of behavior is problem-dependent; therefore, we present this concept on the example of the Edge Unique Label Cover problem.

Luckily, the definition in this case is natural and simple. The behavior on the border terminals, whose number will be bounded by $4k$, is defined as a function $\Psi_b : T_b \to \Sigma$ expressing the labeling we expect on the border terminals. More formally, for an instance of the border problem $\mathcal{I}_b = (G, \Sigma, k, (\psi_{e,v})_{e \in E(G), v \in e})$ with border terminals T_b, by $\mathbb{P}(\mathcal{I}_b)$ we denote the set of all possible functions $\Psi_b : T_b \to \Sigma$. For any $\Psi_b \in \mathbb{P}(\mathcal{I}_b)$, we say that a pair (X, Ψ) is a solution to $(\mathcal{I}_b, \Psi_b)$ if it is a solution to Edge Unique Label Cover on $\mathcal{I}_b$ (ignoring the border terminals) and, additionally, $\Psi|_{T_b} = \Psi_b$. The border problem is defined as follows.

Border E-ULC
Input: An Edge Unique Label Cover instance $\mathcal{I} = (G, \Sigma, k, (\psi_{e,v})_{e \in E(G), v \in e})$ with G being connected, and a set $T_b \subseteq V(G)$ of size at most $4k$; denote $\mathcal{I}_b = (\mathcal{I}, T_b)$.
Output: For each $\Psi_b \in \mathbb{P}(\mathcal{I}_b)$ output a solution $\mathtt{sol}_{\Psi_b} = X_{\Psi_b}$ to $(\mathcal{I}_b, \Psi_b)$ with $|X_{\Psi_b}|$ minimum possible, or output $\mathtt{sol}_{\Psi_b} = \perp$ if such a solution does not exist.

Border E-ULC generalizes Edge Unique Label Cover: we may ask for $T_b = \emptyset$ and take the output for the empty function Ψ_b.

Note that for a Border E-ULC instance $\mathcal{I}_b$, we have $|\mathbb{P}(\mathcal{I}_b)| = |\Sigma|^{|T_b|} \leq s^{4k}$, and the total number of edges output for $\mathcal{I}_b$ is bounded by ks^{4k}. Define $q = ks^{4k} + 1$. In the recursive understanding phase for the Edge Unique Label Cover problem we seek $(q, 2k)$-good separations. The reason why we allow cuts of size $2k$ in the recursion, even though the solution is allowed to cut only k edges, will become more clear in the high connectivity phase. There, we would like to rely on the fact that any two connected subgraphs of more than q vertices are connected by at least $2k+1$ edge-disjoint paths, and the *majority* of these paths do not intersect the solution we are looking for.

Assume that, using the algorithm of Lemma 7, we have found a $(q, 2k)$-good separation (V_1, V_2) of the graph G, for the input instance $\mathcal{I}_b$. As $|T_b| \leq 4k$, at least one of the sides contains at most $2k$ border terminals. Without loss of generality we assume that $|V_1 \cap T_b| \leq 2k$. Now consider an instance $\widehat{\mathcal{I}}_b$ that equals $\mathcal{I}_b$ restricted to vertices V_1, with border terminals $\widehat{T}_b = (V_1 \cap T_b) \cup N_G(V_2)$. In other words, we treat all the endpoints of the cut $\delta(V_1, V_2)$ that lie in V_1 as border terminals. Note that, as $|\delta(V_1, V_2)| \leq 2k$, we have $|\widehat{T}_b| \leq 4k$ and $\widehat{\mathcal{I}}_b$ is a valid Border E-ULC instance.

Now, recursively solve the instance $\widehat{\mathcal{I}}_b$, and let Z be the union of all edges that

appear in any of the solutions output for $\widehat{\mathcal{I}}_b$; note that $|Z| \leq q-1$. It is not hard to see that, for any $\Psi_b \in \mathbb{P}(\mathcal{I}_b)$, if there exists a solution to $(\mathcal{I}_b, \Psi_b)$, then there exists one that does not delete any edge of $E(G[V_1]) \setminus Z$. Indeed, for any solution (X, Ψ) to $(\mathcal{I}_b, \Psi_b)$ one can replace the part of this solution living in $G[V_1]$ with the output to $\widehat{\mathcal{I}}_b$ that is consistent with the appropriate behavior on the border terminals $\widehat{T}_b$, that is, with a solution to $(\widehat{\mathcal{I}}_b, \Psi|_{\widehat{T}_b})$.

Thus, all the edges of $E(G[V_1]) \setminus Z$ can be made undeletable. In most edge-deletion problems, an undeletable edge can be contracted. However, in the case of Edge Unique Label Cover the situation is slightly more involved, as when contracting an edge uv we need also to adjust the constraints on the edges incident to u and v, to take into the account how $\psi_{uv,v}$ translates the label $\Psi(u)$ into $\Psi(v)$ and vice versa. This issue, together with a need of some reduction rule to reduce superfluous parallel edges, causes some technical trouble in the formal proof but does add any real difficulty to the problem. Hence, in this illustration we simply assume that the undeletable edges may be contracted.

We remark that the operation applied to reduce parts of the graph determined to be undeletable is problem-dependent. More complex problems, in particular node-deletion versions, may require even more careful simplification rules.

We now note that the assumptions $|Z| \leq q-1$ and $|V_1| > q$ ensure that at least one edge is contracted and we make progress due to the recursion step. Even more, we infer that there are only at most q vertices left in V_1 after the contraction. With this observation, we proceed to the estimation of the running time of the algorithm. By Lemma 7 the time required to find a $(q, 2k)$-good edge separation is $O(2^{O(k \log q)} n^3 \log n) = O(2^{O(k^2 \log s)} n^3 \log n)$; hence, the total running time is $O(2^{O(k^2 \log s)} n^4 \log n)$. We note that if q, more or less equal the bound on the number of behaviors on the border terminals, is only a function of k, then we always obtain a running time of the form $O(g(k) n^4 \log n)$ for some function g.

2.2. High connectivity phase. We are left with the more involved part of our approach, namely, what to do when no $(q, 2k)$-good edge separation is present in the graph. Note that we can assume that the graph has more than $q(k+1)$ vertices, as otherwise a brute-force search, which checks all the subsets of edges of size at most k, runs within the claimed time complexity bound.

The following simple lemma formalizes the structural properties of the graph after removing the solution. Note that this structure is precisely the gain of the first phase of the algorithm.

Lemma 8. *Let G be a connected graph that admits no $(q, 2k)$-good edge separation. Let F be a set of edges of size at most k, such that $G \setminus F$ has connected components $C_0, C_1, \ldots, C_\ell$. Then* (i) *$\ell \leq k$, and* (ii) *all the components C_i except at most one contain at most q vertices.*

Moreover, for any two connected subgraphs Z_1 and Z_2 of G that are vertex-disjoint and both containing more than q vertices, there exist at least $2k+1$ edge-disjoint paths between vertices of Z_1 and vertices of Z_2.

We would like to remark that if we apply the framework directly to the node-deletion problems, we do not have any bound on ℓ, i.e., the number of components—in the node-deletion setting we need additional tools here.

Fix some behavior on the border terminals $\Psi_b : T_b \to \Sigma$; we iterate through all of them, which gives $2^{O(k \log s)}$ overhead to the running time. Assume that there exists a solution $X \subseteq E(G)$ for this particular choice. Without loss of generality let X be

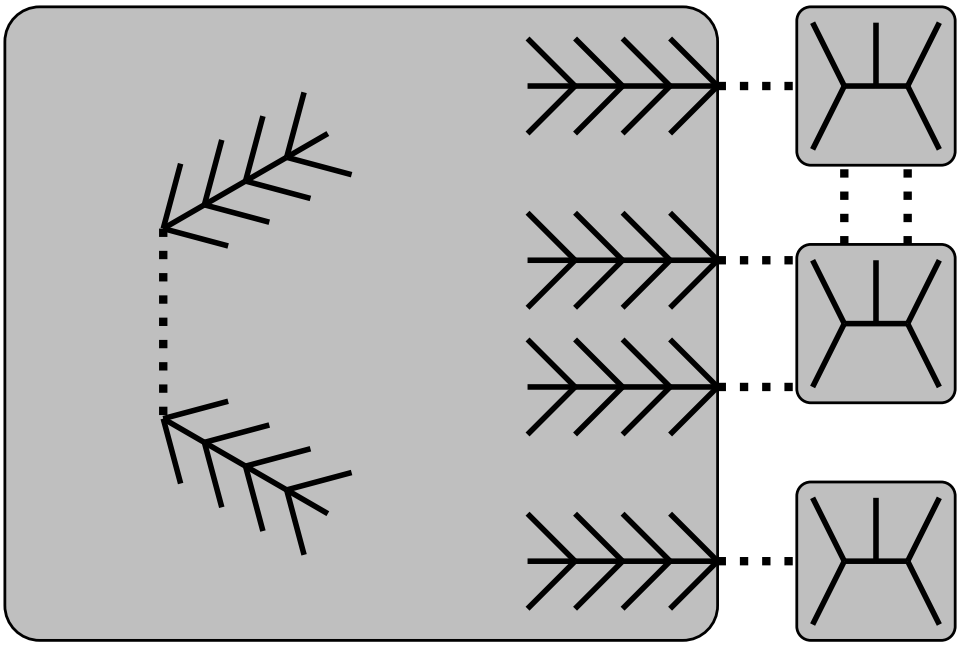

FIG. 1. *An illustration of the application of Lemma* 1 *in the high connectivity phase. The edges of the solution* X *are dotted, and the edges required to be in the interrogating set* S *are thick. We require that* S *contains spanning trees of all small connected components and large subgraphs attached to endpoints of the edges of* X *in the big connected component. Note that it is possible that an edge of the solution has both endpoints in the big connected component. In this case we require that* S *contains large subgraphs attached to its both endpoints.*

of minimum size. Let $C_0, \ldots, C_\ell$ be components of $G \setminus X$, as in Lemma 8, where $|V(C_i)| \leq q$ for $i = 1, 2, \ldots, \ell$. Note that the assumption $|V(G)| > q(k+1)$ implies that $|V(C_0)| > q$, that is, the connected component of unbounded size is actually huge. We call C_0 the *big component*, and other components are *small components*.

We now explain the general methodology for how to highlight the solution X, using Lemma 1. Let $V(X)$ denote the set of endpoints of the edges of X. For every component C_i, choose its arbitrary spanning tree T_i. Let $A_1 = \bigcup_{i=1}^{\ell} E(T_i)$ be the set of edges of the spanning trees of small components. As $\ell \leq k$, we have that $|A_1| \leq (q-1)k$. For every vertex $u \in V(X) \cap V(C_0)$ construct an arbitrary subtree T_0^u of T_0 such that $u \in V(T_0^u)$ and $|V(T_0^u)| = q+1$, and let $A_2 = \bigcup_{u \in V(X) \cap V(C_0)} E(T_0^u)$. We have that $|V(X)| \leq 2k$ and hence $|A_2| \leq 2qk$.

We say that a set $S \subseteq E(G)$ *interrogates* the solution X if $S \cap X = \emptyset$ but $A_1 \cup A_2 \subseteq S$. Note that a family $\mathcal{F}$ constructed by Lemma 1 for the universe $E(G)$ and constants $a = (3q-1)k$ and $b = k$ contains a set that interrogates X. Hence we may branch into $|\mathcal{F}|$ cases, guessing a set S that interrogates the solution we are looking for. We refer to Figure 1 for an illustration.

The set S is our way to highlight the solution X. Note that there are three main properties of an interrogating set:

1. it is disjoint with the solution;
2. it spans all the small connected components; and
3. it spans a large connected subgraph around each endpoint of an edge of X

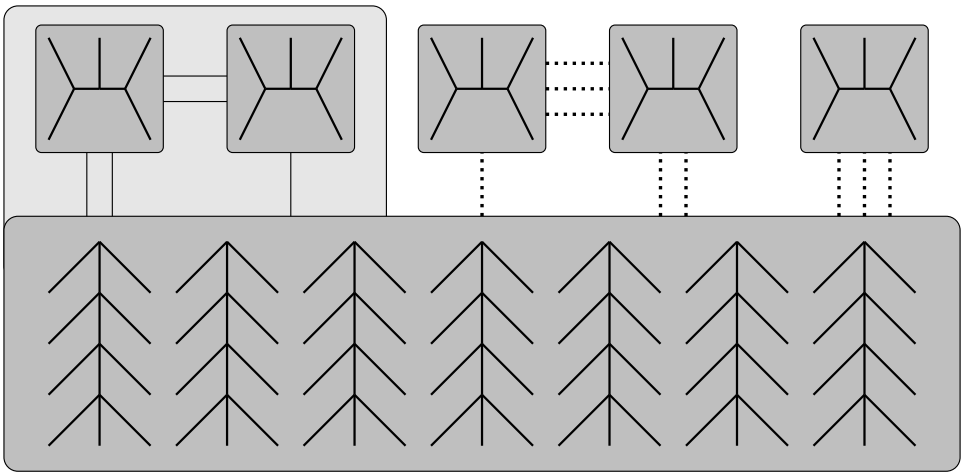

FIG. 2. *An illustration of the situation after the set S is guessed in the high connectivity phase: the bottom half is the union of all big stains, $S^{\mathtt{big}}$; for each connected component of $G \setminus S^{\mathtt{big}}$ we need to decide whether it goes entirely to C_0 (the component on the left) or whether we cut it according to the stains (the components in the middle and on the right).*

that belongs to the big connected component.

In the subsequent arguments we will heavily exploit all three properties. Our goal is to deduce X using its interrogating set S; formally, we are going to find a minimum solution to $(\mathfrak{I}_b, \Psi_b)$ that is additionally interrogated by S.

We analyze connected components of the graph $(V(G), S)$. Each such connected component is called a *stain*. A stain is *big* if it contains more than q vertices and *small* otherwise. Note that any (unknown to us) connected component C_i for $i \geq 1$ is a small stain, whereas all big stains are contained in C_0. Let $S^{\mathtt{big}}$ be the union of vertex sets of all big stains. The following structural observation greatly limits the number of possible sets X to consider.

LEMMA 9. *For any connected component D of $G \setminus S^{\mathtt{big}}$, exactly one of the following is true:*

1. *no edge incident to D is contained in X, and $D \subseteq C_0$;*
2. *D contains no vertex of C_0, and the small stains contained in D are in one-to-one correspondence with components C_i of $G \setminus X$ that are contained in D.*

Proof. If D contains no vertex of C_0, the second property in the second point follows from the assumption that S contains a spanning tree T_i of each connected component C_i for $i \geq 1$ and that S is disjoint from the solution.

If D contains a vertex of C_0, but the first point is not satisfied, then there exists a vertex v that is both in $D \cap C_0$ and is an endpoint of an edge of X. However, then S should contain T_0^v and v belongs to a big stain, a contradiction to the definition of D. ∎

We remark that in the node-deletion setting the situation is a bit more complex, but an equivalent of Lemma 9 can still be proven and exploited. We also refer to Figure 2 for an illustration for Lemma 9.

The main difficulty of the high connectivity phase is to deduce, for each connected component D of $G \setminus S^{\mathtt{big}}$, which option of Lemma 9 is true for D. Once this decision is made, in all problems considered by us it is easy to deduce the entire set X. Let us

now illustrate this claim with the running example of Edge Unique Label Cover. We remark that we now work closely in the Edge Unique Label Cover setting; the following argumentation is highly problem-dependent.

We need the following observation: as we seek a solution X disjoint with S, if we fix a label of a vertex $v \in Z$ for some stain Z, the constraints on the edges of S in Z propagate the labeling to the whole stain Z. Here, we heavily rely on the fact that the constraints in the Unique Label Cover problem are permutations. A labeling of Z that originated from a labeling of a single vertex, propagated through the constraints of S, is called a *reasonable* labeling of Z. Note that there are at most $|\Sigma|$ reasonable labelings of a single stain.

Thus, if for a connected component D of $G \setminus S^{\mathtt{big}}$ we know that the second option of Lemma 9 is true, for each small stain Z contained in D we may find (by enumerating the reasonable labelings of Z) a labeling of Z that minimizes the number of unsatisfied constraints in $G[Z]$. On the other hand, if the first option of Lemma 9 is true for D, then we may forget D for a moment, solve the problem in the rest of the graph, and extend the obtained labeling of $S^{\mathtt{big}}$ to D. The last step should be possible, as we decided not to delete any constaint incident to D.

Thus, we are left with the quest to decide, for each connected component of $G \setminus S^{\mathtt{big}}$, which option of Lemma 9 to choose. In Edge Unique Label Cover, the main trick in this quest is to correctly label $S^{\mathtt{big}}$. Consequently, we now focus on big stains. As in the high connectivity phase our graph does not admit $(q, 2k)$-good separation, any two big stains Z_1 and Z_2 are connected by a family $\mathcal{P}$ of at least $2k+1$ edge-disjoint paths. Let us focus on one path $P \in \mathcal{P}$ and assume that P is disjoint with the solution X. Using again the fact that all constraints in the Unique Label Cover problem are permutations, we infer that for any label assigned to the first vertex of P, there exists a unique way to label all the vertices of P while satisfying all the constraints on the edges of P.

Fix now one of at most $|\Sigma|$ reasonable labelings of Z_1; denote it Ψ_1. Assuming Z_1 is labeled according to Ψ_1, there is a unique way to label the vertices of a path $P \in \mathcal{P}$ assuming P is disjoint with the solution X. Moreover, the obtained (unique) label of the endpoint of P yields a unique reasonable labeling of Z_2. As the *majority* of the paths of $\mathcal{P}$ are disjoint with the solution X, the *majority* of paths of $\mathcal{P}$ should yield *the same* labeling of Z_2, given the labeling Ψ_1 of Z_1. Consequently, fixing a reasonable labeling on one big stain provides us with a unique way to label all other big stains, even without knowing the set X. Therefore, we may branch into at most $|\Sigma|$ ways, guessing the labeling of *all* big stains, that is, of the set $S^{\mathtt{big}}$.

We remark that the argumentation in the previous paragraph is the sole reason for considering cuts of size $2k$ instead of only k in the recursive understanding phase.

Hence, we have obtained a labeling $\Psi^{\mathtt{big}}$ of $S^{\mathtt{big}}$. Consider a component D of $G \setminus S^{\mathtt{big}}$ and assume that the first option of Lemma 9 is true for D. Consequently, the labeling $\Psi^{\mathtt{big}}$ can be (uniquely) extended to D without violating any constraint incident to D. Moreover, observe that an implication is true in the other direction as well: if $\Psi^{\mathtt{big}}$ can be extended to D without violating any constraint incident to D, then we can greedily choose the first option of Lemma 9 for D, as there is no need to delete any edge incident to D (assuming labeling $\Psi^{\mathtt{big}}$).

This finishes the description of high connectivity phase and the entire algorithm for Edge Unique Label Cover.

3. Preliminaries. In this section we prepare ground for formal proofs of the theorems stated in the introduction. We start with setting up the notation, and then we give definitions and preliminary results on "good separations," in both the edge- and node-deletion variants.

3.1. Notation. We use standard graph notation. As the definitions vary among the algorithms, we introduce problem-specific notation at the beginning of each corresponding section, describing whether we work on graphs, multigraphs, or some other structures. Generally, by a graph we denote the pair $G = (V, E)$ consisting vertex set V and edge set E. By $V(G)$ we denote the vertex set of G and by $E(G)$ the edge set. For $F \subseteq E(G)$ by $V(F)$ we denote the set of endpoints of F. For $V_1, V_2 \subseteq V(G)$, by $\delta(V_1, V_2)$ we denote the set of edges with one endpoint in V_1 and second in V_2. For $W \subseteq V(G)$, by $G[W]$ we denote the graph induced by W. For $u \in V(G)$, by $N(u)$ we denote the neighborhood of u, i.e., $N(u) = \{v \mid uv \in E(G)\}$, and the closed neighborhood is defined by $N[u] = N(u) \cup \{u\}$. We extend this notion to subsets in the following manner: for $W \subseteq V(G)$, $N[W] = \bigcup_{u \in W} N[u]$, and $N(W) = N[W] \setminus W$. If X is a set of vertices or edges, by $G \setminus X$ we denote the graph G with edges or vertices of X removed.

3.2. Contractions. In this section we gather the definitions and simple facts connected to the notion of an *edge contraction*. Our definition works in multigraphs.

DEFINITION 10. *Given a multigraph G and an edge $uv \in E(G)$,* contraction *of uv is the operation that yields a new multigraph G' with following properties:*

- *$V(G') = V(G) \setminus \{u, v\} \cup \{w_{uv}\}$, where $w_{uv} \notin V(G)$ is a new vertex;*
- *$E(G')$ is first constructed from $E(G)$ by deleting all edges uv and then substituting all occurrences of u or v by w_{uv} in all the other edges.*

In other words, we preserve multiple edges but delete loops. With contraction of an edge uv we can associate a mapping $\iota_{uv} : V(G) \to V(G')$ by setting $\iota_{uv}(u) = \iota_{uv}(v) = w_{uv}$ and $\iota_{uv}(t) = t$ for all $t \in V(G) \setminus \{u, v\}$. For $w \in V(G)$, we say that vertex w is *contracted onto* $\iota_{uv}(w)$. By somewhat abusing the notation we identify all the edges of $E(G')$ with the edges from $E(G)$ in which they originated. By contracting the edge set $S \subseteq E(G)$ we mean consecutively contracting edges of S in an arbitrary order. Note that if some edge already disappeared from the graph because of becoming a loop, we omit this contraction. We usually use ι to denote the composition of all the mappings ι_{uv} corresponding to the performed contractions. The following lemma, which can be considered folklore, implies that the order of performing the contractions does not matter.

LEMMA 11. *Let G be a multigraph, $D \subseteq E(G)$ be a set of edges, and G' be the graph obtained by contracting D in an arbitrary order. Then the following holds:*

- *$\iota(u) = \iota(v)$ iff u and v can be connected via a path consisting of edges from D for $u, v \in V(G)$.*
- *$\iota^{-1}(v)$ induces a connected subgraph of G for $v \in V(G')$;*
- *$E(G') \subseteq E(G)$;*
- *an edge $vw \in E(G)$ is contained also in $E(G')$ iff $\iota(v) \neq \iota(w)$;*
- *if $X \subseteq V(G')$, then $G'[X]$ is a maximal connected component iff $G[\iota^{-1}(X)]$ is;*
- *in particular, G is connected iff G' is;*
- *for every set F such that $D \cap F = \emptyset$, $G' \setminus F$ can be obtained by contracting D in $G \setminus F$.*

From Lemma 11 it follows that given a graph $G = (V, E)$ and the set $D \subseteq E$, in time $O(|V| + |E|)$ we can construct the graph G' obtained by contracting edges of D. We simply find connected components of the graph (V, D), construct a new vertex for each of them, and for every edge of E check whether it should be introduced in G', and where.

3.3. Preliminary results. We start with a formal proof of Lemma 1.

Proof of Lemma 1. For $a = 0$ or $b = 0$ the lemma is trivial; assume then $a, b \geq 1$.

We use the standard technique of splitters. A (n, r, r^2)-*splitter* is a family of functions from $\{1, 2, \ldots, n\}$ to $\{1, 2, \ldots, r^2\}$, such that for any subset $X \subseteq \{1, 2, \ldots, n\}$ of size r, one of the functions in the family is injective on X. Naor, Schulman, and Srinivasan [49] gave an explicit construction of an (n, r, r^2)-splitter of size $O(r^6 \log r \log n)$ using $O(\text{poly}(r) \cdot n \log n)$ time.

Without loss of generality, assume that $a \leq b$ and that $U = \{1, 2, \ldots, n\}$. Let $c = \min(a + b, n)$. We construct a (n, c, c^2)-splitter using the algorithm of Naor, Schulman, and Srinivasan and, for each function f in the splitter and for each subset $S' \subseteq \{1, 2, \ldots, c^2\}$ of size a, we put into the family $\mathcal{F}$ the set $f^{-1}(S') \subseteq U$. Assume now that we have $A, B \subseteq U$ such that $|A| \leq a$ and $|B| \leq b$. Obtain A' and B' by adding arbitrary elements of $U \setminus (A \cup B)$ to A and B so that $|A'| + |B'| = c$. By definition of the splitter, there exists some f in the splitter that is injective on $A' \cup B'$. To finish the proof one needs to observe that if we take $S = f^{-1}(f(A'))$, then $A \subseteq S$ and $B \cap S = \emptyset$.

The time bound and the size of the constructed family $\mathcal{F}$ follow from the bound on the size of the splitter and the fact that there are at most $\binom{(a+b)^2}{a} = 2^{O(a \log(a+b))}$ choices for the set S'; note that for fixed f and S', the set $f^{-1}(S')$ can be computed in linear time. ∎

A well-known result by Nagamochi and Ibaraki [47] states that the graph can be efficiently sparsified while preserving all the essential connectivity.

LEMMA 12 (see [47]). *Given an undirected graph $G = (V, E)$ and an integer k, in $O(k(|V| + |E|))$ time we can obtain a set of edges $E_0 \subseteq E$ of size at most $(k + 1)(|V| - 1)$, such that for any edge $uv \in E \setminus E_0$ in the graph (V, E_0) there are at least $k + 1$ edge-disjoint paths between u and v.*

Proof. The algorithm performs exactly $k + 1$ iterations. In each iteration it finds a spanning forest F of the graph G, adds all the edges of F to E_0, and removes all the edges of F from the graph G.

Observe that for any edge uv remaining in the graph G, the vertices u and v are in the same connected components in each of the forests found. Hence in each of those forests we can find a path between u and v; thus, we obtain $k + 1$ edge-disjoint paths between u and v. ∎

3.4. Good separations in edge-deletion problems. For the sake of completeness, let us recall the definition of a (q, k)-good edge separation.

DEFINITION 13. *Let G be a connected graph. A partition (V_1, V_2) of $V(G)$ is called a* (q, k)-good edge separation *if*

- $|V_1|, |V_2| > q$;
- $|\delta(V_1, V_2)| \leq k$;
- $G[V_1]$ *and* $G[V_2]$ *are connected.*

We are ready to present proofs of lemmas regarding algorithms finding good edge separations.

LEMMA 14. *There exists a deterministic algorithm that, given an undirected, connected graph G on n vertices along with integers q and k, in time $O(2^{O(\min(q,k)\log(q+k))} n^3 \log n)$ either finds a (q,k)-good edge separation or correctly concludes that no such separation exists.*

Proof. The algorithm iterates through all the sets from the family $\mathcal{F}$, obtained from Lemma 1 for universe $U = E(G)$ and constants $a = 2q$ and $b = k$. For a set $S \in \mathcal{F}$, we obtain a new graph H by contracting all the edges of S. Let $\iota : V(G) \to V(H)$ be the mapping that maps every vertex of G to the vertex it is contracted onto. We say that a vertex $u \in V(H)$ is *big* if $|\iota^{-1}(u)| > q$. Now, for every pair of big vertices $u_1, u_2 \in V(H)$ we compute some minimum edge cut between u_1 and u_2 if it is of size at most k, or we find that it has to have larger size. This can be done in $O(k^2 n^3)$ time, since first we can sparsify the graph by removing all the edges outside of the set E_0 returned by Lemma 12, and next for each of the $O(n^2)$ pairs of big vertices using the classical algorithm by Ford and Fulkerson in $O(k^2 n)$ time find a cut of size at most k if it exists. Assume that for some pair of big vertices u_1, u_2 we have found a minimum edge cut F_{u_1,u_2}, of size at most k. We claim that F_{u_1,u_2} induces a (q,k)-good edge separation of G, which can be returned as the output of the algorithm.

Let $v_1 \in \iota^{-1}(u_1)$ and $v_2 \in \iota^{-1}(u_2)$ be arbitrary vertices. Let V_1, V_2 be the sets of vertices reachable from v_1, v_2 in $G \setminus F_{u_1,u_2}$, respectively. We claim that (V_1, V_2) is a (q,k)-good edge separation of G. First, observe that V_1 and V_2 are disjoint. Otherwise there would be a path from v_1 to v_2 in G that avoids F_{u_1,u_2}, which after applying the contractions would become a path from u_1 to u_2 in H that avoids F_{u_1,u_2}. Second, observe that $V_1 \cup V_2 = V(G)$. It follows from the well-known properties of minimum cuts that in $H \setminus F_{u_1,u_2}$ every vertex is reachable either from u_1 or from u_2. As graphs $G[\iota^{-1}(u)]$ are connected for $u \in H$, we find that in G every vertex is reachable either from v_1 or from v_2. Third, observe that $|V_1|, |V_2| > q$, as $\iota^{-1}(u_1) \subseteq V_1$ and $\iota^{-1}(u_2) \subseteq V_2$.

We are left with proving that if the graph admits a (q,k)-good edge separation, then for at least one set $S_0 \in \mathcal{F}$ we obtain two big vertices that can be separated by an edge cut of size at most k. This ensures that if no solution has been found for any $S \in \mathcal{F}$, then the algorithm can safely provide a negative answer. Fix some (q,k)-good edge separation (V_1, V_2) and let T_1, T_2 be arbitrary subtrees of $G[V_1]$ and $G[V_2]$, respectively, each having exactly $q+1$ vertices. By the choice of family $\mathcal{F}$, there exists $S_0 \in \mathcal{F}$ that contains all the edges of T_1 and T_2 but is disjoint with $\delta(V_1, V_2)$. In the step when S_0 is considered, after applying contractions all the vertices of T_1 are contracted onto one vertex u_1, all the vertices of T_2 are contracted onto one vertex u_2, but edges from $\delta(V_1, V_2)$ are not being contracted. Hence, we obtain big vertices u_1, u_2 that can be separated by an edge cut of size at most k. ∎

LEMMA 15. *Let G be a connected graph that admits no (q,k)-good edge separation. Let F be a set of edges of size at most k, such that $G \setminus F$ has connected components $C_0, C_1, \ldots, C_\ell$. Then* (i) *$\ell \leq k$, and* (ii) *all the components C_i except at most one contain at most q vertices.*

Proof. Claim (i) follows directly from the fact that removing an edge from the graph can increase the number of connected components by at most one. For claim (ii), observe that if two components had at least q vertices, then F could serve as an edge cut between their vertex sets of size at most k. It follows that the minimum edge cut between their vertex sets would also have size bounded by k, hence it would induce a (q,k)-good edge separation in G. ∎

We now show that we can improve the polynomial factor in the running time of the procedure of Lemma 14, at the cost of randomization.

LEMMA 16. *There exists a randomized algorithm that, given an undirected, connected graph $G = (V, E)$ along with integers q and k, in time $\tilde{O}(2^{O(\min(q,k)\log(q+k))}(|V|+|E|))$ either finds a (q,k)-good edge separation or correctly concludes that no such separation exists with probability at least $(1 - 1/|V|^2)$.*

Proof. Let (V_1, V_2) be a (q,k)-good edge separation. Intuitively, we want to have an edge-contraction process such that no edge of $\delta(V_1, V_2)$ is contracted and each vertex which remains is big, because then any cut of size at most k gives a (q,k)-good edge separation, which we can find by using Karger's algorithm. We use Lemma 1 in a fashion very similar to the proof of Lemma 14; however, as a few details are different, we repeat the entire proof.

The algorithm iterates through all the sets from the family $\mathcal{F}$, obtained from Lemma 1 for universe $U = E(G)$ and constants $a = 2qk$ and $b = k$. For a set $S \in \mathcal{F}$, we obtain a new graph H' by contracting all the edges of S. Let $\iota' : V(G) \to V(H')$ be the mapping that maps every vertex of G to the vertex it is contracted onto. We say that a vertex $u' \in V(H')$ is *big* if $|\iota'^{-1}(u')| > q$ and *small* otherwise. Let $S' \subseteq E(H')$ be the set of edges of H' having at least one small endpoint. We construct a graph H by contracting all the edges of S' in H'. Let $\iota : V(G) \to V(H)$ be the mapping from the graph G to the graph H. Note that after contracting all the edges of S' all the vertices are big in the graph H with respect to ι. By using Karger's algorithm [35], in $\tilde{O}(k\log(qk)(|V|+|E|))$ time we find the minimum cut in the graph H with probability at least $(1 - \frac{1}{2^{ck\log(qk)}|V|^2\log|V|})$ for some constant c. If the minimum cut found is of size at most k, it immediately gives a (q,k)-good edge separation in the graph G, since all the vertices of H are big.

We are left with proving that if G admits a (q,k)-good edge separation (V_1, V_2), then for at least one set $S_0 \in \mathcal{F}$ the graph H contains a cut of size at most k, providing some (possibly different) (q,k)-good edge separation. This ensures that if no solution has been found for any $S \in \mathcal{F}$, then the algorithm can safely provide a negative answer. For each vertex $u \in N(V_2) \subseteq V_1$ let T^u be an arbitrary subtree of $G[V_1]$ containing the vertex u, having exactly $q+1$ vertices. Similarly, for each vertex $u \in N(V_1)$ let T^u be an arbitrary subtree of $G[V_2]$ containing u, having exactly $q+1$ vertices. By the choice of the family $\mathcal{F}$, there exists $S_0 \in \mathcal{F}$ that contains all the edges of T^u for each $u \in V(\delta(V_1, V_2))$, but at the same time S_0 is disjoint with $\delta(V_1, V_2)$. In the step when S_0 is considered, after applying contractions, for each $u \in V(\delta(V_1, V_2))$ all the vertices of T^u are contracted onto one vertex u', which is big. However, the edges from $\delta(V_1, V_2)$ are not being contracted. Observe that in the graph H' no edge of $\delta(V_1, V_2)$ has a small endpoint, and consequently all of the edges of $\delta(V_1, V_2)$ are present in the graph H, and they induce a cut of size at most k.

Note that the algorithm of Karger is used $O(2^{O(k\log(qk))}\log|V|)$ times, and therefore, by the union bound, if our algorithm does not find a (q,k)-good edge separation, with probability at least $(1 - 1/|V|^2)$ it does not exist. ∎

3.5. Good separations in node-deletion problems. As we consider node-deletion problems in most of our results, we need to define an appropriate variant of good separations; that is the main goal of this section. In the edge-deletion variant, we might have assumed that we only consider cuts that separate the graph into exactly two connected components; this is no longer a case in the node-deletion variant.

Moreover, the applications require us to handle the possibility that some vertices are undeletable.

It turns our that in the node-deletion problems we need to use two types of separations. In the first one, we require that after removal of the separator, at least two connected components are large.

DEFINITION 17. *Let G be a connected graph and $V^\infty \subseteq V(G)$ a set of undeletable vertices. A triple (Z, V_1, V_2) of subsets of $V(G)$ is called a* (q,k)-good node separation *if*

- $|Z| \leq k$,
- $Z \cap V^\infty = \emptyset$,
- *V_1 and V_2 are vertex sets of two different connected components of $G \setminus Z$, and*
- $|V_1 \setminus V^\infty|, |V_2 \setminus V^\infty| > q$.

In the second one we require a bunch of connected components with the same neighborhood.

DEFINITION 18. *Let G be a connected graph, $V^\infty \subseteq V(G)$ a set of undeletable vertices, and $T_b \subseteq V(G)$ a set of border terminals in G. A pair $(Z, (V_i)_{i=1}^\ell)$ is called a* (q,k)-flower separation *in G (with regard to border terminals T_b) if the following holds:*

- *$1 \leq |Z| \leq k$ and $Z \cap V^\infty = \emptyset$; the set Z is the* core *of the flower separation $(Z, (V_i)_{i=1}^\ell)$;*
- *V_i are vertex sets of pairwise different connected components of $G \setminus Z$ each set V_i is a* petal *of the flower separation $(Z, (V_i)_{i=1}^\ell)$;*
- *$V(G) \setminus (Z \cup \bigcup_{i=1}^\ell V_i)$, called a* stalk*, contains more than q vertices of $V \setminus V^\infty$;*
- *for each petal V_i we have $V_i \cap T_b = \emptyset$, $|V_i \setminus V^\infty| \leq q$, and $N_G(V_i) = Z$;*
- *$|(\bigcup_{i=1}^\ell V_i) \setminus V^\infty| > q$.*

We now show how to detect the aforementioned separations using Lemma 1, similarly as in the case of good edge separations.

LEMMA 19. *Given a connected graph G with undeletable vertices $V^\infty \subseteq V(G)$ and integers q and k, one may find in $O(2^{O(\min(q,k)\log(q+k))} n^3 \log n)$ time a (q,k)-good node separation of G or correctly conclude that no such separation exists.*

Proof. The algorithm iterates through all the sets from the family $\mathcal{F}$, obtained from Lemma 1 for universe $U = V(G) \setminus V^\infty$ and constants $a = 2q+2$ and $b = k$. For a set $S \in \mathcal{F}$, we obtain a new graph H by contracting all the edges between vertices of $S \cup V^\infty$ in G. Let $\iota : V(G) \to V(H)$ be the mapping that maps every vertex of G to the vertex it is contracted onto, and let $S' = \iota(S \cup V^\infty)$. We say that a vertex $u \in S'$ is *big* if $|\iota^{-1}(u) \setminus V^\infty| > q$.

In the graph H, we assign weight ∞ to all vertices of S' and weight 1 to all vertices of $V(H) \setminus S'$. In this weighted graph, for every pair of big vertices u_1 and u_2, we compute a minimum node cut between u_1 and u_2 if it is of size at most k or find that it has to have larger size. This can be done in $O(kn^3)$ time using the Gomory–Hu tree extended to node weighted separations by Granot and Hassin [30]. That is, we can use $|V(H)|-1$ applications of the classic Ford–Fulkerson algorithm, each of which consumes $O(kn^2)$ time, since after finding $k+1$ vertex disjoint paths we may stop the algorithm. Assume that for some pair of big vertices u_1, u_2 we have found a minimum node cut F_{u_1,u_2}, of size at most k. We claim that F_{u_1,u_2} induces a (q,k)-good node separation of G, which can be returned as the output of the algorithm.

Let $v_1 \in \iota^{-1}(u_1) \setminus V^\infty$ and $v_2 \in \iota^{-1}(u_2) \setminus V^\infty$ be arbitrary vertices. Note that $F_{u_1,u_2} \subseteq V(G) \setminus V^\infty$, as only vertices of $V(H) \setminus S' = V(G) \setminus (S \cup V^\infty)$ have finite weights. Let V_1, V_2 be the sets of vertices reachable from v_1, v_2 in $G \setminus F_{u_1,u_2}$, respectively. We claim that (F_{u_1,u_2}, V_1, V_2) is a (q,k)-good node separation of G. Indeed, V_1 and V_2 are defined as vertex sets of two connected components of $G \setminus F_{u_1,u_2}$; moreover, $V_1 \neq V_2$ as F_{u_1,u_2} separates u_1 from u_2 in H, and therefore v_1 from v_2 in G. Finally, observe that $|V_1 \setminus V^\infty|, |V_2 \setminus V^\infty| > q$, as $\iota^{-1}(u_1) \subseteq V_1$ and $\iota^{-1}(u_2) \subseteq V_2$.

We are left with proving that if the graph admits a (q,k)-good node separation, then for at least one set $S_0 \in \mathcal{F}$ we obtain two big vertices that can be separated by a node cut of size at most k. This ensures that if no solution has been found for any $S \in \mathcal{F}$, then the algorithm can safely provide a negative answer. Fix some (q,k)-good node separation (Z, V_1, V_2) and let T_1, T_2 be arbitrary subtrees of $G[V_1]$ and $G[V_2]$, respectively, each having exactly $q+1$ vertices that are in $V(G) \setminus V^\infty$. As $|Z| \leq k$, by the choice of family $\mathcal{F}$, there exists $S_0 \in \mathcal{F}$ that contains $(V(T_1) \cup V(T_2)) \setminus V^\infty$ but is disjoint with Z. In the step when S_0 is considered, after applying contractions all the vertices of T_1 are contracted onto one vertex u_1, all the vertices of T_2 are contracted onto one vertex u_2, but vertices of Z get weight 1. Hence, we obtain big vertices u_1, u_2 that can be separated by a node cut of size at most k (note that the algorithm does not necessarily find precisely the cut Z in this step). ∎

LEMMA 20. *Given a connected graph G with undeletable vertices $V^\infty \subseteq V(G)$ and border terminals $T_b \subseteq V(G)$ and integers q and k, one may find in $O(2^{O(\min(q,k)\log(q+k))} n^3 \log n)$ time a (q,k)-flower separation in G w.r.t. T_b or correctly conclude that no such flower separation exists.*

Proof. We first note that given a set $Z \subseteq V(G)$ of size at most k, we can in $O(n^2)$ time verify whether there exists a (q,k)-flower separation with Z as the core, that is, $(Z, (V_i)_{i=1}^\ell)$ for some choice of the family of petals $(V_i)_{i=1}^\ell$. Indeed, we may simply iterate over connected components of $G \setminus Z$ using a simple dynamic program. For each prefix of the sequence of connected components and for each $n' \leq n$ we compute whether some of the components can be chosen to be petals so that the total number of vertices of $V(G) \setminus V^\infty$ in the petals is equal to n'. When we consider the next connected component, if it does not satisfy requirements for a petal, then we cannot take it as a petal (and we take the value of the cell computed in the last iteration for the same value of n'). However, if it does satisfy these requirements, then we either do not take it to be a petal (and do the same as previously) or take it (and we take the value of the cell computed in the last iteration for the value n' decremented by the number of vertices from $V \setminus V^\infty$ in the considered component). There exists a flower separation with Z as the center iff some of the values for $q + 1 \leq n' \leq |V(G) \setminus (V^\infty \cup Z)| - q - 1$ are true in the last iteration. It is trivial to augment the dynamic program with backlinks, so that the flower separation can be retrieved.

To prove the lemma, we iterate through all the sets from the family $\mathcal{F}$, obtained from Lemma 1 for universe $U = V(G) \setminus V^\infty$ and constants $a = q$ and $b = k$. For a set $S \in \mathcal{F}$, we obtain a new graph H by contracting all the edges between vertices of $S \cup V^\infty$ in G. Let $\iota : V(G) \to V(H)$ be the mapping that maps every vertex of G to the vertex it is contracted onto, and let $S' = \iota(S \cup V^\infty)$. We say that a vertex $u \in S'$ is *interesting* if $|\iota^{-1}(u) \setminus V^\infty| \leq q$. For each interesting vertex u with $|N_H(u)| \leq k$ we verify whether there exists a (q,k)-flower separation $(Z, (V_i)_{i=1}^\ell)$ in G w.r.t. T_b with the core $Z = N_H(u)$; note that $N_H(u) \subseteq V(G)$. We output such a flower separation if we find one. If no flower separation is found for any choice of S and u, we conclude

that no (q,k)-flower separation exists in G w.r.t. T_b. The time bound follows from the fact that for each vertex u, we can verify whether u is interesting and compute $N_H(u)$ in $O(n^2)$ time, and then within the same complexity check if $N_H(u)$ is the core of some (q,k)-flower separation. To finish the proof of the lemma we need to show that if the algorithm concludes that there is no appropriate flower separation in the graph, then this conclusion is correct.

To this end, assume that there exists a (q,k)-flower separation $(Z,(V_i)_{i=1}^{\ell})$ in G w.r.t. T_b. Note that $|V_1 \setminus V^\infty| \leq q$ ($\ell \geq 1$ since $|(\bigcup_{i=1}^{\ell} V_i) \setminus V^\infty| > q$) and $|Z| \leq k$, so by the properties of the family $\mathcal{F}$ there exists a set $S_0 \in \mathcal{F}$ with $(V_1 \setminus V^\infty) \subseteq S_0$ and $Z \cap S_0 = \emptyset$. Recall that $N_G(V_1) = Z$; thus, in the graph H constructed for the set S_0 there is a vertex $u \in V(H)$ with $\iota^{-1}(u) = V_1$. Note that u is an interesting vertex (as $|V_1 \setminus V^\infty| \leq q$) and $N_H(u) = Z$ (as $N_G(V_1) = Z$ by the definition of the flower separation). Therefore the algorithm considers $Z = N_H(u)$ and finds a (q,k)-flower separation in G w.r.t. T_b. ∎

We conclude this section with a lemma that shows that if we do not have any good node or flower separations, then any k-cut not only cannot split the graph into two large components but also cannot split the graph into too many small ones.

LEMMA 21. *If a connected graph G with undeletable vertices $V^\infty \subseteq V(G)$ and border terminals $T_b \subseteq V(G)$ does not contain a (q,k)-good node separation or a (q,k)-flower separation w.r.t. T_b, then, for any $Z \subseteq V(G) \setminus V^\infty$ of size at most k, the graph $G \setminus Z$ contains at most $(2q+1)(2^k-1)+|T_b|+1$ connected components containing a vertex of $V \setminus V^\infty$, out of which at most one has more than q vertices not in V^∞.*

Proof. Let $Z \subseteq V(G) \setminus V^\infty$, $|Z| \leq k$. First, if there are at least two connected components of $G \setminus Z$ with more than q vertices in $V(G) \setminus V^\infty$, then a minimal subset of Z separating these two components would induce a (q,k)-good node separation in G. Thus, in $G \setminus Z$ we have at most one connected component with more than q vertices outside V^∞ and at most $|T_b|$ connected components that contain a vertex from T_b. We denote the remaining connected components containing at least one vertex of $V \setminus V^\infty$ as *nice* ones; they have at most q vertices outside V^∞ each. Let us partition them with respect to their neighborhood (which is a subset of Z). Note that if there exists a set $Z' \subseteq Z$, such that at least $2q+2$ nice connected components of $G \setminus Z$ that are adjacent to exactly Z', then there exists a (q,k)-flower separation in G w.r.t. T_b with core Z' and petals being $q+1$ of aforementioned nice connected components of $G \setminus Z$. As there are at most $2^k - 1$ nonempty subsets of Z, the lemma follows. ∎

4. The algorithm for Node Unique Label Cover. This section is devoted to fixed-parameter tractability of the NODE UNIQUE LABEL COVER problem, parameterized by both the size of the cutset and the size of the alphabet. We solve a bit more general version of the problem, where we allow arbitrary unary relations and we allow the binary relations to be only partial permutations. This generalizations appear naturally in our branching and reduction rules.

More formally, for an alphabet Σ, a binary relation $\psi \subseteq \Sigma \times \Sigma$ is called a *partial permutation* if for every $\alpha \in \Sigma$ both $(\{\alpha\} \times \Sigma) \cap \psi$ and $(\Sigma \times \{\alpha\}) \cap \psi$ are of size at most one; in other words, for every $\alpha \in \Sigma$, at most one value β satisfies $\phi(\alpha,\beta)$ and at most one value β' satisfies $\phi(\beta',\alpha)$. For a partial permutation ψ, its reverse is defined as $\psi^{-1} = \{(\beta,\alpha) : (\alpha,\beta) \in \psi\}$. For any two partial permutations ψ_1, ψ_2 their composition is defined as

$$\psi_2 \circ \psi_1 = \{(\alpha,\gamma) : \exists_{\beta \in \Sigma} (\alpha,\beta) \in \psi_1 \wedge (\beta,\gamma) \in \psi_2\}.$$

Note that a composition of two partial permutations is a partial permutation itself and behaves in a similar manner as a composition of functions.

It is more convienient notationally to treat the deletion of a vertex as another, special label ☠. Formally, we consider the following problem.

NODE UNIQUE LABEL COVER	**Parameter:** $k+s$
Input: An undirected graph G, a finite alphabet Σ of size s, an integer k, for each vertex $v \in V(G)$ a set $\phi_v \subseteq \Sigma$ and for each edge $e \in E(G)$ and each its endpoint v a partial permutation $\psi_{e,v}$ of Σ, such that if $e = uv$, then $\psi_{e,u} = \psi_{e,v}^{-1}$.	
Question: Does there exist a function $\Psi : V(G) \to \Sigma \cup \{☠\}$ such that at most k vertices are assigned value ☠, for every $v \in V(G)$ we have $\Psi(v) \in \phi_v \cup \{☠\}$ and for every $uv \in E(G \setminus X)$ we have $\Psi(u) = ☠$, $\Psi(v) = ☠$, or $(\Psi(u), \Psi(v)) \in \psi_{uv,u}$?	

The relations $\psi_{e,u}$ are called *edge constraints*, the sets ϕ_v are called *vertex constraints*, the function Ψ is called a *labeling*, and the set $\Psi^{-1}(☠)$ is the *deletion set*. For a vertex v with $\Psi(v) = ☠$, we say that v *is deleted by* Ψ.

Before we start, we note that the edge-deletion variant (where we look for a deletion set being a subset of edges; we are to label all vertices, but we do not need to satisfy the constraints on the deleted edges) reduces to the defined above node-deletion variant.

Indeed, first observe that in the NODE UNIQUE LABEL COVER problem we can assume that additionally we are given in the input a set of undeletable vertices $V^\infty \subseteq V(G)$ and we are to find a labeling Ψ that does not delete any vertex of V^∞: we can reduce this variant to the original one by replacing each undeletable vertex with a clique on $k+1$ vertices, with constraints on the edges of the clique being identities. Second, given an EDGE UNIQUE LABEL COVER instance $(G, \Sigma, k, (\phi_v)_{v \in V(G)}, (\psi_{e,v})_{e \in E(G), v \in e})$, we can first make all vertices of G undeletable and then subdivide each edge, so that the edge constraints on the two halves of the edge $e = uv$ of G compose to the constraint $\psi_{e,u}$; the new vertices introduced in this operation are kept deletable.

We remark that the aforementioned reduction blows up the number of vertices and edges of the input graph; thus the obtained algorithm for EDGE UNIQUE LABEL COVER has worse dependency on n than $n^4 \log n$ we obtain for the node-deletion variant. Note that section 2 contains a sketch of an algorithm of EDGE UNIQUE LABEL COVER with $n^4 \log n$ dependency on n in the running time. As the main result of our work is fixed-parameter tractability of considered problems, and the proven dependency on n is far from being linear or even quadratic, we refrain from formally showing an algorithm for EDGE UNIQUE LABEL COVER with $n^4 \log n$ dependency on n in the running time.

Note that we may assume that the input graph G in the NODE UNIQUE LABEL COVER problem is connected; otherwise, we may solve the problem on each connected component, for all budgets between 0 and k, separately. During the course of the algorithm, we maintain the connectivity of G. We denote by n the number of vertices of the graph of the currently considered NODE UNIQUE LABEL COVER instance.

We also assume that the elements of Σ can be compared in constant time.

The description of the algorithm consists of a sequence of *steps*. Each step is accompanied with some lemmas and a discussion that justifies its correctness and verifies complexity bounds.

4.1. Labelings. We first extend the notion of labeling to arbitrary subsets of $V(G)$.

DEFINITION 22. *Given an instance* $\mathcal{I} = (G, \Sigma, k, (\phi_v)_{v \in V(G)}, (\psi_{e,v})_{e \in E(G), v \in e})$ *and a set* $S \subseteq V(G)$*, a function* $\Psi^S : S \to \Sigma \cup \{☠\}$ *is called a* labeling *if it satisfies all constraints on* $G[S]$ *in* $\mathcal{I}$*, that is, for each* $v \in S$ *we have* $\Psi^S(v) \in \phi_v \cup \{☠\}$ *and for each* $uv \in E(G[S])$ *we have* $\Psi^S(u) = ☠$*,* $\Psi^S(v) = ☠$*, or* $(\Psi^S(u), \Psi^S(v)) \in \psi_{uv,u}$*.*

A labeling is deletion-free *if it does not assign the value* ☠ *to any vertex.*

For a labeling Ψ, by $\mathrm{dom}(\Psi)$ we denote its domain.

The following lemma is a straightforward corollary of the fact that the edge constraints are partial permutations.

LEMMA 23. *Let* $\mathcal{I} = (G, \Sigma, k, (\phi_v)_{v \in V(G)}, (\psi_{e,v})_{e \in E(G), v \in e})$ *be a* NODE UNIQUE LABEL COVER *instance and let* $A \subseteq V(G)$ *be an arbitrary subset of the vertex set that induces a connected subgraph of* G*. Then, for every* $v \in A$ *and* $\alpha \in \Sigma$ *there exists at most one deletion-free labeling* $\Psi^A : A \to \Sigma$ *such that* $\Psi^A(v) = \alpha$*. Furthermore, in* $O(s|A|^2)$ *time one can find such a labeling or correctly conclude that it does not exist. Consequently, for each set* $A \subseteq V(G)$ *such that* $G[A]$ *is connected, there are at most* s *deletion-free labelings of* A *and those can be enumerated in* $O(s^2|A|^2)$ *time.*

Proof. Note that for every $uw \in E(G[A])$, if $\Psi^A(u)$ is fixed, then there exists at most one value $\Psi^A(w)$ such that $(\Psi^A(u), \Psi^A(w)) \in \psi_{uw,u}$, and such a value $\Psi^A(w)$ can be found in $O(s)$ time. The first claim of the lemma follows from the assumption that $G[A]$ is connected: the labeling Ψ^A can be found using a breadth-first search and then verified to satisfy all the constraints in $O(s|A|^2)$ time. For the second claim, we simply iterate over all possible values $\Psi^A(v)$ for one fixed vertex $v \in A$. ∎

4.2. Operations on the input graph. In this section we define two basic operations the algorithm repetitively applies on the graph and show their key properties.

DEFINITION 24. *Let* $\mathcal{I} = (G, \Sigma, k, (\phi_v)_{v \in V(G)}, (\psi_{e,v})_{e \in E(G), v \in e})$ *be a* NODE UNIQUE LABEL COVER *instance, let* $u, v \in V(G)$*, and let* ψ *be a partial permutation of* Σ*. By* updating an edge uv with a constraint ψ *we mean the following operation: if* $uv \notin E(G)$*, then we add an edge* uv *to the graph* G *with constraints* $\psi_{uv,u} = \psi$*,* $\psi_{uv,v} = \psi^{-1}$*; otherwise, we modify the constraints on the edge* uv *in* G *by replacing* $\psi_{uv,u}$ *with* $\psi_{uv,u} \cap \psi$ *and* $\psi_{uv,v}$ *with* $\psi_{uv,v} \cap \psi^{-1}$*.*

Informally speaking, updating an edge uv with ψ is equivalent to adding a new edge between u and v with this constraint; however, we use the definition above to avoid multiple edges in G. Note that obviously updating an edge cannot spoil the assumption of connectivity of G. The following lemma is immediate.

LEMMA 25. *Let* $\mathcal{I}'$ *be a* NODE UNIQUE LABEL COVER *instance obtained from* $\mathcal{I}$ *by updating and edge* uv *with a constraint* ψ*. Then* Ψ *is a solution to* $\mathcal{I}'$ *iff it is a solution to* $\mathcal{I}$ *that satisfies the following additional property: either* $\Psi(u) = ☠$*,* $\Psi(v) = ☠$*, or* $(\Psi(u), \Psi(v)) \in \psi$*.*

The second operation allows us to remove a vertex that, for some reason, will not be deleted by an optimum solution.

DEFINITION 26. *Let* $\mathcal{I} = (G, \Sigma, k, (\phi_v)_{v \in V(G)}, (\psi_{e,v})_{e \in E(G), v \in e})$ *be a* NODE UNIQUE LABEL COVER *instance and* $v \in V(G)$*. By* bypassing the vertex v *we mean the following operation:*

1. *remove the vertex* v *with its incident edges from the graph* G*;*
2. *for each* $u \in N_G(v)$ *we replace* ϕ_u *with* $\phi_u \cap \{\beta : \exists_{\alpha \in \phi_v} (\alpha, \beta) \in \psi_{uv,v}\}$*;*

3. *for each* $u_1, u_2 \in N_G(v)$, $u_1 \neq u_2$, *we update an edge* u_1u_2 *with a constraint* $\psi_{vu_2,v} \circ \psi_{vu_1,u_1}$.

In the next lemma we formally check that bypassing a vertex has the same meaning as proclaiming it undeletable.

LEMMA 27. *Let* $\mathcal{I}'$ *be a* NODE UNIQUE LABEL COVER *instance obtained from an instance*

$$\mathcal{I} = (G, \Sigma, k, (\phi_v)_{v \in V(G)}, (\psi_{e,v})_{e \in E(G), v \in e})$$

by bypassing a vertex v *with* $\phi_v \neq \emptyset$. *Then the following holds:*

- *if* Ψ *is a solution to* $\mathcal{I}'$, *then there exists* $\alpha \in \Sigma$ *such that* $\Psi \cup \{(v, \alpha)\}$ *is a solution to* $\mathcal{I}$;
- *if* Ψ *is a solution to* $\mathcal{I}$ *that satisfies* $\Psi(v) \neq$ ☠, *then* $\Psi|_{V(G)\setminus\{v\}}$ *is a solution to* $\mathcal{I}'$.

Proof. For the first claim, pick α as follows. If there exists a neighbor of v whose value in Ψ is not ☠, pick any such neighbor w, set α such that $(\Psi(w), \alpha) \in \psi_{vw,w}$ and $\alpha \in \phi_v$. Note that such α exists as, by the definition of the bypassing operation, in $\mathcal{I}'$ the vertex constraint for w are contained in $\{\beta : \exists_{\alpha' \in \phi_v}(\beta, \alpha') \in \psi_{vw,w}\}$. If such a neighbor w does not exist, pick $\Psi(v)$ to be an arbitrary element of ϕ_v.

We claim that $\Psi \cup \{(v, \alpha)\}$ is a solution to $\mathcal{I}$. Clearly, $\Psi \cup \{(v, \alpha)\}$ satisfies all vertex constraints of $V(G)$ as well as all edge constraints on edges not incident to v, as those constraints in $\mathcal{I}$ are supersets of the corresponding constraints in $\mathcal{I}'$. Moreover, the choice of α ensures that $\alpha \in \phi_v$. We are left with verifying edge constraints $\psi_{uv,v}$ for $u \in N_G(v)$. If $\Psi(u) =$ ☠, then we are done, and if $u = w$, then clearly $(\alpha, \Psi(w)) \in \psi_{vw,v}$ by the choice of α. Otherwise, by the definition of the bypassing operation, $(\Psi(w), \Psi(u)) \in \psi_{uv,v} \circ \psi_{vw,w}$. Since $(\Psi(w), \alpha) \in \psi_{vw,w}$, we infer that $(\alpha, \Psi(u)) \in \psi_{uv,v}$ and the claim is proven.

For the second claim, denote $\alpha = \Psi(v)$. To prove the claim we need to verify that $\Psi|_{V(G)\setminus\{v\}}$ satisfies vertex constraints on $N_G(v)$ (that may shrink during the bypassing operation) and edge constraints on edges between vertices in $N_G(v)$ (that are updated during the bypassing operation). First consider a vertex $u \in N_G(v)$. If $\Psi(u) =$ ☠, there is nothing to check, so assume otherwise. Since Ψ is a solution to $\mathcal{I}$ and $\Psi(v) \neq$ ☠, we have $\Psi(u) \in \phi_u$, $\alpha \in \phi_v$ and $(\Psi(u), \alpha) \in \psi_{uv,u}$. Thus $\Psi(u) \in \{\beta : \exists_{\alpha' \in \phi_v}(\alpha', \beta) \in \psi_{uv,v}\}$ and $\Psi(u)$ satisfies the vertex constraint at u in $\mathcal{I}'$. Second, consider two vertices $u_1, u_2 \in N_G(v)$, $u_1 \neq u_2$ and $\Psi(u_1) \neq$ ☠, $\Psi(u_2) \neq$ ☠. Since Ψ is a solution to $\mathcal{I}$ and $\Psi(v) \neq$ ☠, we have $(\Psi(u_1), \alpha) \in \psi_{vu_1,u_1}$ and $(\alpha, \Psi(u_2)) \in \psi_{vu_2,v}$. Therefore $(\Psi(u_1), \Psi(u_2)) \in \psi_{vu_2,v} \circ \psi_{vu_1,u_1}$ and the claim is proven. ∎

During the course of the algorithm we perform bypassing operations multiple times, which can drastically increase the number of edges, even if the graph was sparse in the beginning. Therefore, we measure the complexity of our algorithm only in n, the number of vertices, and always use only the trivial quadratic bound on the number of edges.

4.3. Borders and recursive understanding. As discussed in the illustration, in the case of the NODE UNIQUE LABEL COVER problem, the definition of the border variant is completely natural: informally speaking, for each vertex on the border, we need to know whether it is deleted and, if not, what label is assigned to it. More formally, given a NODE UNIQUE LABEL COVER instance $\mathcal{I} = (G, \Sigma, k, (\phi_v)_{v \in V(G)}, (\psi_{e,v})_{e \in E(G), v \in e})$ and a set of border terminals $T_b \subseteq V(G)$, for

every function $\mathcal{P} : T_b \to \Sigma \cup \{☠\}$, we say that a solution Ψ to $\mathcal{I}$ *is consistent with* $\mathcal{P}$ if $\Psi|_{T_b} = \mathcal{P}$. Let $\mathbb{P}(\mathcal{I})$ be the set of all functions from T_b to $\Sigma \cup \{☠\}$. We define the border problem as follows.

BORDER N-ULC
Input: A NODE UNIQUE LABEL COVER instance $\mathcal{I} = (G, \Sigma, k, (\phi_v)_{v\in V(G)}, (\psi_{e,v})_{e\in E(G), v\in e})$ with G being connected, and a set $T_b \subseteq V(G)$ of size at most $4k$.
Output: For each $\mathcal{P} \in \mathbb{P}(\mathcal{I})$, output a solution $\mathtt{sol}_{\mathcal{P}} = \Psi_{\mathcal{P}}$ to the instance $\mathcal{I}$ that is consistent with $\mathcal{P}$ and deletes (assigns ☠) to minimum possible number of vertices, or output $\mathtt{sol}_{\mathcal{P}} = \perp$ if no such solution exists.

Note that $|\mathbb{P}(\mathcal{I})| \leq (s+1)^{4k}$ and all output solutions $\Psi_{\mathcal{P}}$ delete at most $k(s+1)^{4k}$ different vertices in total. Let $q = k(s+1)^{4k} + 2k$; if $|V(G)| > q + 2k$, then there are at least $|V(G)| - q - 2k$ vertices in G that are not in T_b nor are deleted by any of the output solutions $\Psi_{\mathcal{P}}$.

In the next lemma we formalize what a recursive step looks like and verify its correctness. The statement and its proof, although technical and notationally quite heavy, is completely standard: we essentially need to verify that the information carried by the boundary terminals in the definiton of BORDER N-ULC is sufficient to independently substitute partial solutions on different sides of a separation.

LEMMA 28. *Assume we are given a* BORDER N-ULC *instance*

$$\mathcal{I}_b = (G, \Sigma, k, (\phi_v)_{v\in V(G)}, (\psi_{e,v})_{e\in E(G), v\in e}, T_b)$$

and two disjoint sets of vertices $Z, \widehat{V} \subseteq V(G)$, *such that* $|Z| \leq 2k$, $N_G(\widehat{V}) \subseteq Z$, $|\widehat{V} \cap T_b| \leq 2k$ *and the subgraph of* G *induced by* $W := \widehat{V} \cup Z_W$ *is connected, where* $Z_W := N_G(\widehat{V})$. *Denote* $\widehat{T_b} = (T_b \cup Z_W) \cap W$ *and*

$$\widehat{\mathcal{I}}_b = (G[W], \Sigma, k, (\phi_v)_{v\in W}, (\psi_{e,v})_{e\in E(G[W]), v\in e}, \widehat{T_b}).$$

Then $\widehat{\mathcal{I}}_b$ *is a proper* BORDER N-ULC *instance. Moreover, if we denote by* $(\widehat{\mathtt{sol}}_{\mathcal{P}})_{\mathcal{P}\in\mathbb{P}(\widehat{\mathcal{I}}_b)}$ *an arbitrary output to the* BORDER N-ULC *instance* $\widehat{\mathcal{I}}_b$ *and*

$$U(\widehat{\mathcal{I}}_b) = \widehat{T_b} \cup \bigcup \{\widehat{\Psi}_{\mathcal{P}}^{-1}(☠) : \mathcal{P} \in \mathbb{P}(\widehat{\mathcal{I}}_b), \widehat{\mathtt{sol}}_{\mathcal{P}} = \widehat{\Psi}_{\mathcal{P}} \neq \perp\},$$

then there exists a correct output $(\mathtt{sol}_{\mathcal{P}})_{\mathcal{P}\in\mathbb{P}(\mathcal{I}_b)}$ *to the* BORDER N-ULC *instance* $\mathcal{I}_b$ *such that every vertex that is deleted by some solution* $\mathtt{sol}_{\mathcal{P}} \neq \perp$ *belongs to* $U(\widehat{\mathcal{I}}_b)$.

Proof. The claim that $\widehat{\mathcal{I}}_b$ is a proper BORDER N-ULC instance follows directly from the assumptions that $G[W]$ is connected, $|Z_W| \leq |Z| \leq 2k$, and $|\widehat{V} \cap T_b| \leq 2k$. In the rest of the proof we justify the second claim of the lemma.

Fix $\mathcal{P} \in \mathbb{P}(\mathcal{I}_b)$. Assume that there exists a solution to the instance $\mathcal{I}_b$ that is consistent with $\mathcal{P}$; let $\Psi_{\mathcal{P}}$ be such a solution with the minimum possible number of deleted vertices. To prove the lemma we need to show a second solution $\Psi'_{\mathcal{P}}$ to $\mathcal{I}_b$ that deletes no more vertices than $\Psi_{\mathcal{P}}$ does, is consistent with $\mathcal{P}$, and such that all vertices from W that are deleted by $\Psi'_{\mathcal{P}}$ lie in $U(\widehat{\mathcal{I}}_b)$.

Let $\widehat{\mathcal{P}}$ be the restriction of $\Psi_{\mathcal{P}}$ to $\widehat{T_b}$. Note that $\widehat{\mathcal{P}} \in \mathbb{P}(\widehat{\mathcal{I}}_b)$ and $\Psi_{\mathcal{P}}|_W$ is a solution to $\widehat{\mathcal{I}}_b$ consistent with $\widehat{\mathcal{P}}$. Therefore the output $\widehat{\mathtt{sol}}_{\widehat{\mathcal{P}}}$ to $\widehat{\mathcal{I}}_b$ is different from $\perp$; denote

it $\widehat{\Psi}_{\widehat{\mathcal{P}}}$. By definition, this output deletes minimum possible number of vertices; in particular

$$|\widehat{\Psi}_{\widehat{\mathcal{P}}}^{-1}(☠)| \leq |\Psi_{\mathcal{P}}^{-1}(☠) \cap W|.$$

Define $\Psi'_{\mathcal{P}} : V(G) \to \Sigma \cup \{☠\}$ as follows: $\Psi'_{\mathcal{P}}(v) = \widehat{\Psi}_{\widehat{\mathcal{P}}}(v)$ for $v \in W$ and $\Psi'_{\mathcal{P}}(v) = \Psi_{\mathcal{P}}(v)$ otherwise. By the optimality of $\widehat{\Psi}_{\widehat{\mathcal{P}}}$, the number of vertices deleted by $\Psi'_{\mathcal{P}}$ is not larger than the number of vertices deleted by $\Psi_{\mathcal{P}}$. Since $\Psi'_{\mathcal{P}}$ is a blend of two labelings, it clearly satisfies all vertex constraints. The fact that $Z_W = N(\widehat{V}) \subseteq \widehat{T}_b$ ensures that $\Psi'_{\mathcal{P}}$, $\widehat{\Psi}_{\widehat{\mathcal{P}}}$, and $\Psi_{\mathcal{P}}$ agree on Z_W and thus $\Psi'_{\mathcal{P}}$ satisfies all edge constraints. Finally, $T_b \cap W \subseteq \widehat{T}_b$ and $\widehat{\mathcal{P}}$ is a restriction of $\Psi_{\mathcal{P}}$, which in turn is consistent with $\mathcal{P}$, and the labeling $\Psi'_{\mathcal{P}}$ is consistent with $\mathcal{P}$ as well. This finishes the proof of the lemma. ∎

Note that in Lemma 28 we have $|U(\widehat{\mathcal{I}}_b) \cap \widehat{V}| \leq q$. We are now ready to present the recursive steps of the algorithm.

STEP 4.1. *Assume we are given a* BORDER N-ULC *instance*

$$\mathcal{I}_b = (G, \Sigma, k, (\phi_v)_{v \in V(G)}, (\psi_{e,v})_{e \in E(G), v \in e}, T_b).$$

Invoke first the algorithm of Lemma 19 *in a search for a* $(q, 2k)$*-good node separation (with* $V^\infty = \emptyset$*). If it returns a good node separation* (Z, V_1, V_2)*, let* $j \in \{1, 2\}$ *be such that* $|V_j \cap T_b| \leq 2k$ *and denote* $\widehat{Z} = Z$*,* $\widehat{V} = V_j$*. Otherwise, if it returns that no such good node separation exists in* G*, invoke the algorithm of Lemma* 20 *in a search for a* (q, k)*-flower separation w.r.t.* T_b *(with* $V^\infty = \emptyset$ *again). If it returns that no such flower separation exists in* G*, pass the instance* $\mathcal{I}_b$ *to the next step. Otherwise, if it returns a flower separation* $(Z, (V_i)_{i=1}^{\ell})$*, denote* $\widehat{Z} = Z$ *and* $\widehat{V} = \bigcup_{i=1}^{\ell} V_i$*.*

In the case we have obtained $\widehat{Z}$ *and* $\widehat{V}$ *(from either Lemma* 19 *or Lemma* 20*), invoke the algorithm recursively for the* BORDER N-ULC *instance* $\widehat{\mathcal{I}}_b$ *defined as in the statement of Lemma* 28 *for separator* $\widehat{Z}$ *and set* $\widehat{V}$*, obtaining an output* $(\widehat{\texttt{sol}}_{\mathcal{P}})_{\mathcal{P} \in \mathbb{P}(\widehat{\mathcal{I}}_b)}$*. Compute the set* $U(\widehat{\mathcal{I}}_b)$*. Bypass (in an arbitrary order) all vertices of* $\widehat{V} \setminus U(\widehat{\mathcal{I}}_b)$ *to obtain a new instance* $\mathcal{I}'_b$ *(observe that for each bypassed vertex* v *we have* $\phi_v \neq \emptyset$*, which is a necessary condition for bypassing). Recall that* $\widehat{T}_b \subseteq U(\widehat{\mathcal{I}}_b)$*, so no border terminal gets bypassed. Restart the algorithm on the new instance* $\mathcal{I}'_b$ *and obtain a family of solutions* $(\texttt{sol}'_{\mathcal{P}})_{\mathcal{P} \in \mathbb{P}(\mathcal{I}_b)}$*. For every* $\mathcal{P} \in \mathbb{P}(\mathcal{I}_b)$*, if* $\texttt{sol}'_{\mathcal{P}} = \perp$*, then output* $\texttt{sol}_{\mathcal{P}} = \perp$ *as well, while if* $\texttt{sol}_{\mathcal{P}} = \Psi'_{\mathcal{P}}$*, then obtain* $\Psi_{\mathcal{P}}$ *by extending* $\Psi'_{\mathcal{P}}$ *on* $U(\widehat{\mathcal{I}}_b)$ *using Lemma* 23 *(we justify that such an extension exists in Lemma* 29*) and output* $\texttt{sol}_{\mathcal{P}} = \Psi_{\mathcal{P}}$*.*

Let us first verify that the application of Lemma 28 is justified. Indeed, by the definitions of the good node separation and the flower separation, as well as the choice of $\widehat{V}$, we have in both cases $|\widehat{V} \cap T_b| \leq 2k$ and that $G[\widehat{V} \cup N_G(\widehat{V})]$ is connected. Moreover, note that the recursive call is applied to the graph with strictly smaller number of vertices than G: in the case of a good node separation, V_2 is removed from the graph, and in the case of a flower separation, recall that the definition of the flower separation requires $Z \cup \bigcup_{i=1}^{\ell} V_i$ to be a proper subset of $V(G)$. Finally, in both cases $|\widehat{V}| > q$, and $|\widehat{V} \setminus U(\widehat{\mathcal{I}}_b)| \geq |\widehat{V}| - q \geq 1$ vertices are bypassed in Step 4.1.

The following lemma verifies the correctness of Step 4.1.

LEMMA 29. *Assume we are given a* BORDER N-ULC *instance*

$$\mathcal{I}_b = (G, \Sigma, k, (\phi_v)_{v \in V(G)}, (\psi_{e,v})_{e \in E(G), v \in e}, T_b)$$

on which Step 4.1 *is applied, and let* $\mathcal{I}'_b$ *be an instance after all bypassing operations of Step* 4.1 *are applied. Let* $(\mathtt{sol}'_{\mathcal{P}})_{\mathcal{P}\in\mathbb{P}(\mathcal{I}'_b)}$ *be a correct output to* $\mathcal{I}'_b$*. Then there exists a correct output* $(\mathtt{sol}_{\mathcal{P}})_{\mathcal{P}\in\mathbb{P}(\mathcal{I}_b)}$ *to* $\mathcal{I}_b$*, such that*

- $\mathtt{sol}_{\mathcal{P}} = \bot$ *if* $\mathtt{sol}'_{\mathcal{P}} = \bot$*;*
- *if* $\mathtt{sol}'_{\mathcal{P}} = \Psi'_{\mathcal{P}}$*, then* $\Psi'_{\mathcal{P}}$ *can be consistently extended to* $V(G)$ *and for every such extension* $\Psi_{\mathcal{P}}$ *is a correct output for* $\mathcal{P}$ *in* $\mathcal{I}_b$*;*

Proof. The lemma is a straightforward corollary of Lemma 28 and the properties of the bypassing operation described in Lemma 27. Lemma 28 ensures us that each vertex of $\widehat{V} \setminus U(\widehat{\mathcal{I}}_b)$ is omitted by some optimal solution for every $\mathcal{P} \in \mathbb{P}(\mathcal{I}_b)$, which enables us to use Lemma 27. Note that existence of the extension is asserted by the second claim of Lemma 27. ∎

We are left with an analysis of the time complexity of Step 4.1. The applications of Lemmas 19 and 20 use $O(2^{O(\min(q,2k)\log(q+2k))}n^3 \log n) = O(2^{O(k^2 \log s)}n^3 \log n)$ time. Let $n' = |\widehat{V}|$; the recursive step is applied to the graph with at most $n' + 2k$ vertices and, after bypassing, there are at most $n - n' + q$ vertices left. Moreover, each bypassing operation takes $O(sn^2)$ time, the computation of $U(\widehat{\mathcal{I}}_b)$ takes $O((s+1)^{4k}n)$ time, and extending the labelings from the trimmed instance takes $O((s+1)^{4k}sn^2)$ time. The values of $s = |\Sigma|$ and k do not change in this step. Therefore, we have the following recursive formula for time complexity as a function of the number of vertices of G:

$$T(n) \leq \max_{q+1\leq n'\leq n-2k-1} \Big(2^{O(k^2 \log s)}n^3 \log n + T(n'+2k) + T(n-n'+q)\Big). \tag{1}$$

Note that the function $p(t) = t^4 \log t$ is convex, so the maximum of the expression is attained at one of the ends. A straightforward inductive check of both of the ends proves that we have indeed the claimed bound on the complexity, i.e., $T(n) = O(2^{O(k^2 \log s)}n^4 \log n)$.

We conclude this section with a note that Lemma 21 asserts that if Step 4.1 is not applicable, then for every set $Z \subseteq V(G)$ of size at most k, the graph $G \setminus Z$ contains at most $t := (2q+1)(2^k - 1) + 4k + 1$ connected components, out of which at most one has more than q vertices.

4.4. Brute-force approach. If the graph reduced by Step 4.1 is small, the algorithm may apply a straightforward brute-force approach to the Border N-ULC problem. In this section we describe this method formally.

Lemma 30. *Assume we are given a* Border N-ULC *instance*

$$\mathcal{I}_b = (G, \Sigma, k, (\phi_v)_{v\in V(G)}, (\psi_{e,v})_{e\in E(G), v\in e}, T_b).$$

Let $X \subseteq V(G)$ *be a set of size at most* k *and let* $\mathcal{P} \in \mathbb{P}(\mathcal{I}_b)$*. Then, in time* $O(s^2n^2)$*, one can compute a function* $\Psi : V(G) \to \Sigma \cup \{☠\}$ *such that* $\Psi^{-1}(☠) = X$ *and* Ψ *is a solution to* $\mathcal{I}_b$ *consistent with* $\mathcal{P}$ *or correctly conclude that no such function exists.*

Proof. We apply Lemma 23 to the vertex set of every connected component of the graph induced by $A = V(G) \setminus X$. For each output labeling, we verify if it is consistent with $\mathcal{P}$. ∎

Lemma 31. *A correct output to a* Border N-ULC *instance*

$$\mathcal{I}_b = (G, \Sigma, k, (\phi_v)_{v\in V(G)}, (\psi_{e,v})_{e\in E(G), v\in e}, T_b)$$

can be computed in $O((s+1)^{4k}s^2kn^{k+2})$ *time.*

Proof. We apply Lemma 30 for each $\mathcal{P} \in \mathbb{P}(\mathcal{I}_b)$ (there are at most $(s+1)^{4k}$ choices) and for each deletion set $X \subseteq V(G)$ with $|X| \leq k$ (at most $(k+1)n^k$ choices). ∎

STEP 4.2. *If $|V(G)| \leq qt + k$, apply Lemma* 31 *to find a correct output to a* BORDER N-ULC *instance* $\mathcal{I}_b = (G, \Sigma, k, (\phi_v)_{v \in V(G)}, (\psi_{e,v})_{e \in E(G), v \in e}, T_b)$.

Recall that $q = (s+1)^{2k}k = 2^{O(k \log s)}$ and $t = (2q+2)(2^k - 1) + 2k + 1 = 2^{O(k \log s)}$. Thus, if Step 4.2 is applicable, its running time is $2^{O(k^2 \log s)}$.

4.5. High connectivity phase. Assume we have a BORDER N-ULC instance $\mathcal{I}_b = (G, \Sigma, k, (\phi_v)_{v \in V(G)}, (\psi_{e,v})_{e \in E(G), v \in e}, T_b)$ where Steps 4.1 and 4.2 are not applicable. In this section we show how to exploit high connectivity of the graph implied by Lemma 21 to compute a correct output to $\mathcal{I}_b$. To this end, fix $\mathcal{P} \in \mathbb{P}(\mathcal{I}_b)$; we focus on finding the solution $\texttt{sol}_{\mathcal{P}}$. First, let us solve some simple cases.

STEP 4.3. *For each $\mathcal{P} \in \mathbb{P}(\mathcal{I}_b)$, verify using Lemma* 30 *whether there exists solution $\texttt{sol}_{\mathcal{P}} = \Psi_{\mathcal{P}}$ that does not delete any vertex at all. If yes, output such a solution.*

Note that if $|V(G)|$ is too large for Step 4.2 to be applicable, for every set $Z \subseteq V(G)$ of size at most k, the bound on the number of connected components from Lemma 21 implies that there exists exactly one connected component of $G \setminus Z$ with more than q vertices; denote its vertex set by $\texttt{big}(Z)$. We extend this notion to labelings: for a labeling Ψ that deletes at most k vertices, we denote $\texttt{big}(\Psi) = \texttt{big}(\Psi^{-1}(☠))$.

4.5.1. Interrogating sets. We now use Lemma 1 to get some more structure of the graph G.

DEFINITION 32. *Let $Z \subseteq V(G)$ be a set of size at most k and let $S \subseteq V(G)$. We say that S* interrogates *Z if the following holds:*

1. *$S \cap Z = \emptyset$;*
2. *for every connected component C of $G \setminus Z$ with at most q vertices, all vertices of C belong to S;*
3. *for every $v \in Z$, such that $N_G(v) \cap \texttt{big}(Z) \neq \emptyset$, there exists a connected component of $G[S]$ that has more than q vertices and contains at least one neighbour of v.*

We say that S interrogates *a labeling Ψ if it interrogates the deletion set $\Psi^{-1}(☠)$.*

Note that in the third point, the considered component has to be entirely contained in $\texttt{big}(Z)$ due to its size.

LEMMA 33. *Let $\mathcal{F}$ be a family obtained by the algorithm of Lemma* 1 *for universe $U = V(G)$ and constants $a = qt + (q+1)k$ and $b = k$. Then, for every $Z \subseteq V(G)$ with $1 \leq |Z| \leq k$, there exists a set $S \in \mathcal{F}$ that interrogates Z.*

Proof. Fix $Z \subseteq V(G)$ with $|Z| \leq k$. Let A_1 be the union of vertex sets of all connected components of $G \setminus Z$ that have at most q vertices; by Lemma 21, $|A_1| \leq qt$. For each $v \in Z$ such that $N_G(v) \cap \texttt{big}(Z) \neq \emptyset$, fix $w_v \in N_G(v) \cap \texttt{big}(Z)$ and a tree T_v with exactly $q+1$ vertices that contains w_v and is contained in $\texttt{big}(Z)$; note that this is possible due to $|\texttt{big}(Z)| > q$. Let A_2 be the union of vertex sets of all trees T_v for $v \in Z$; clearly $|A_2| \leq (q+1)k$. By Lemma 1, as $|A_1 \cup A_2| \leq qt + (q+1)k$ and $|Z| \leq k$, there exists a set $S \in \mathcal{F}$ that contains $A_1 \cup A_2$ and is disjoint with Z. By the construction of the sets A_1 and A_2, the set S interrogates Z and the lemma is proven. ∎

Note that, as $q, t = 2^{O(k \log s)}$, the family $\mathcal{F}$ of Lemma 33 is of size $2^{O(k^2 \log s)} \log n$

and can be computed in $O(2^{O(k^2 \log s)} n \log n)$ time. Therefore we may branch, guessing a set S that interrogates the solution $\mathtt{sol}_{\mathcal{P}} = \Psi_{\mathcal{P}}$ we are looking for.

STEP 4.4. *Compute the family $\mathcal{F}$ from Lemma 33 and branch into $|\mathcal{F}|$ subcases, indexed by sets $S \in \mathcal{F}$. In a branch S we seek for a solution $\Psi_{\mathcal{P}}$ with the minimum possible number of deleted vertices that not only is a solution to $\mathcal{I}_b$ consistent with $\mathcal{P}$ but is also interrogated by S.*

Lemma 33 verifies the correctness of the branching of Step 4.4; as discussed, the step is applied in $O(2^{O(k^2 \log s)} n \log n)$ time and leads to $O(2^{O(k^2 \log s)} \log n)$ subcases.

After choosing a set S, we may now slightly modify the set S to make it more regular.

DEFINITION 34. *A vertex $v \in V(G)$ is said to be* forsaken *if*
- *$v \in T_b$ and $\mathcal{P}(v) =$ ☠ or*
- *$\phi_v = \emptyset$.*

The forsaken vertices are those that are necessarily deleted by any solution $\Psi_{\mathcal{P}}$ consistent with $\mathcal{P}$.

STEP 4.5. *As long as there exists a vertex $v \in V(G)$ that is not forsaken and $N_G[v] \cap S = \emptyset$, add v to S.*

Step 4.5 can clearly be applied in $O(sn^2)$ time (for all vertices it is applied to; note that Step 4.5 is applied to one vertex v at a time and, by its application to the vertex v, it may become not applicable to the neighbors of v). We now discuss its correctness. Let $\Psi_{\mathcal{P}}$ be a solution to $\mathcal{I}_b$ that is interrogated by S and consistent with $\mathcal{P}$. Then $\Psi_{\mathcal{P}}$ is interrogated by $S \cup \{v\}$ unless $\Psi_{\mathcal{P}}(v) =$ ☠. If this is the case, then since $N_G[v] \cap S = \emptyset$, by the last property of an interrogating set v is not adjacent to any vertex of $\mathtt{big}(\Psi_{\mathcal{P}})$. Moreover, by the second property of an interrogating set, v is not adjacent to any vertex of connected component of $G \setminus \Psi_{\mathcal{P}}^{-1}($☠$)$ of size at most q. We infer that all vertices of $N_G[v]$ are deleted by the labeling $\Psi_{\mathcal{P}}$. Since v is not a forsaken vertex, there exists a value $\alpha \in \phi_v$ such that if we assign α to v in the labeling $\Psi_{\mathcal{P}}$, we obtain a solution to $\mathcal{I}_b$ that is consistent with $\mathcal{P}$ (but not necessarily interrogated by S). Therefore $\Psi_{\mathcal{P}}$ is not a solution to $\mathcal{I}_b$ consistent with $\mathcal{P}$ with minimum possible number of deleted vertices, and we may omit it from consideration.

Step 4.5 gives us the following property of the set S.

LEMMA 35. *After Step 4.5 is applied, any vertex v that is not forsaken is contained in $N_G[S]$.*

Proof. If v is not forsaken and $v \notin N_G[S]$, then $N_G[v] \cap S = \emptyset$ and Step 4.5 is applicable to v. ∎

4.5.2. Labelings of big stains. Let us now focus on a fixed branch $S \in \mathcal{F}$.

DEFINITION 36. *Each connected component of $G[S]$ is called a* stain. *A stain is* big *if it has more than q vertices and is* small *otherwise.*

Let $S^{\mathtt{big}} \subseteq S$ be the union of all vertex sets of big stains of $G[S]$. We now establish a crucial observation that the fact that G admits no $(q, 2k)$-good node separations implies that there are only very few reasonable labelings for $S^{\mathtt{big}}$.

LEMMA 37. *One can in $O(ksn^3 + ks^2n^2)$ time compute a family* **PSI** *of at most s deletion-free labelings of $S^{\mathtt{big}}$ such that for every solution Ψ to $\mathcal{I}_b$ such that S interrogates Ψ, there exists $\Psi^{\mathtt{big}} \in$ **PSI** with $\Psi|_{S^{\mathtt{big}}} = \Psi^{\mathtt{big}}$.*

Proof. If $S^{\mathtt{big}} = \emptyset$ the lemma is trivial; assume then $S^{\mathtt{big}} \neq \emptyset$. For any big stain with vertex set C in $G[S]$, by Lemma 23 there are at most s deletion-free labelings of C and all these labelings can be computed in $O(s^2|C|^2)$ time. Moreover, we know that any such a labeling is induced by fixing a label of one vertex of C; in other words, for every two different labelings Ψ', Ψ'' of $G[C]$ and for every $v \in C$, we have $\Psi'(v) \neq \Psi''(v)$.

Let C_1 and C_2 be vertex sets of two different big stains in $G[S]$. As G admits no $(q, 2k)$-good node separations, by Menger's theorem there exists a sequence $P^0, P^1, \ldots, P^{2k}$ of $2k+1$ paths, where each path starts in C_1 and ends in C_2 and the sets of internal vertices of those paths are pairwise disjoint. Moreover, such a sequence of paths $P^0, P^1, \ldots, P^{2k}$ can be found in $O(kn^2)$ time by the classic Ford–Fulkerson algorithm.

Let Ψ be a solution to $\mathfrak{I}_b$ that is interrogated by S. The crucial observation is that for at least $k+1$ indices $0 \leq i \leq 2k$ (i.e., for the majority of them), the path P^i does not contain a vertex deleted by Ψ (note that the endpoints of P^i are in $C_1 \cup C_2 \subseteq S$ and thus not deleted by Ψ). Denote the endpoints of P^i as $v_1^i \in C_1$ and $v_2^i \in C_2$. If P^i does not contain any vertex deleted by Ψ, the composition of all edge constraints on P^i (denote it by ψ^i) is a partial permutation such that $(\Psi(v_1^i), \Psi(v_2^i)) \in \psi^i$. We infer that for every $0 \leq i \leq 2k$ and every labeling Ψ' of $G[C_1]$ there exists at most one labeling $\Psi'_{C_2,i}$ of $G[C_2]$ such that $(\Psi'(v_1^i), \Psi'_{C_2,i}(v_2^i)) \in \psi^i$. Moreover, given Ψ', all labelings $\Psi'_{C_2,i}$ for all $0 \leq i \leq 2k$ can be computed in $O(ks(n+s|C_2|^2))$ time using Lemma 23.

Let $\Psi' = \Psi|_{C_1}$. As at least $k+1$ paths P^i do not contain any vertices deleted by Ψ, for a majority of indices $0 \leq i \leq 2k$, the labelings Ψ'_{i,C_2} are the same labelings. For a fixed big stain C_1 and for each labeling Ψ' of $G[C_1]$, we can compute this majority labeling Ψ'_{maj,C_2} of $G[C_2]$ for every big stain $C_2 \neq C_1$ in time $O(kn^2 + ks(n+s|C_2|^2))$ (including the time needed to compute paths P^i). As there are at most n big stains, and s labelings of a fixed big stain C_1, the lemma follows. ∎

Note that for every $Z \subseteq V(G)$, $1 \leq |Z| \leq k$, there exists the component $\mathtt{big}(Z)$ and, if S interrogates Z, then there exists at least one big stain in $G[S]$ (note that we require here that Z is nonempty; the solutions with empty deletion sets are found by Step 4.3). This observation, together with Lemma 37, justifies the following step.

STEP 4.6. *For each $\mathcal{P} \in \mathfrak{I}$, in a branch with index S, if $G[S]$ contains no big stains, terminate this branch, and otherwise invoke Lemma 37 to obtain a family* **PSI** *and branch into at most s subcases, indexed by labelings $\Psi^{\mathtt{big}} \in$ **PSI**. For fixed $\mathcal{P}$, in a branch with indices S and $\Psi^{\mathtt{big}}$, we seek a solution $\Psi_{\mathcal{P}}$ with minimum possible size of the deletion set, such that $\Psi_{\mathcal{P}}$ is a solution to $\mathfrak{I}_b$ consistent with $\mathcal{P}$, interrogated by S and $\Psi_{\mathcal{P}}|_{S^{\mathtt{big}}} = \Psi^{\mathtt{big}}$.*

Each application of Step 4.6 takes $O(ksn^3 + ks^2n^2)$ time and leads to $O(2^{O(k^2 \log s)} \log n)$ subcases in total.

4.5.3. Final bounded search tree algorithm. In this section we show how to finish the search for an appropriate output $\mathtt{sol}_{\mathcal{P}}$ for $\mathcal{P} \in \mathbb{P}(\mathfrak{I}_b)$ in a fixed branch with indices S and $\Psi^{\mathtt{big}}$. This is done in a standard framework of a bounded search tree algorithm. Informally speaking, the goal is to look at all connected components of G after removal of all already deleted vertices and $N_G[S^{\mathtt{big}}]$ and decide, one by one, whether it should be merged into the $\mathtt{big}(\mathtt{sol}_{\mathcal{P}})$.

Formally speaking, we maintain a labeling Ψ, initiated as $\Psi^{\mathtt{big}} \cup \mathcal{P}$ together with

all forsaken vertices deleted, and our goal is to extend it to the desired solution via a bounded search algorithm. That is, at every recursive call of the branching algorithm, we look for a solution with minimum number of deleted vertices that additionally extends Ψ.

At every step, either no branching is performed and some new value is added to Ψ or we branch into $O(\Sigma)$ directions, in every branch deleting at least one new vertex.

The branching algorithm is described as a set of four *rules*. At each moment, we apply the first applicable rule.

First, let us define the stopping condition for the branching algorithm.

RULE 4.1 (finishing rule). *If there is some inconsistensy in Ψ (it deletes more than k vertices or violates one of the constraints), then terminate the current branch. If Ψ can be extended to $V(G)$ without deleting any additional vertex, then return such an extension as a solution for the current branch.*

Note that recognizing whether the finishing rule can be applied takes $O(s^2n^2)$ time using Lemma 23.

Given the labeling Ψ, let $N(\Psi)$ be the set of vertices that do not belong to the domain of Ψ but have a neighbor that is assigned a value from Σ by Ψ (i.e., in the domain of Ψ but not deleted by Ψ). Furthermore define $N[\Psi] = N(\Psi) \cup \mathrm{dom}(\Psi)$. Consider now the following task: given a labeling Ψ, we would like to extend Ψ to $N(\Psi)$ without deleting any new vertex. Observe that there is essentially at most only one candidate Ψ^N for such an extension: for every $v \in N(\Psi)$, we fix a neighbor $w(v)$ of v with $\Psi(w(v)) \in \Sigma$ and assign to v the unique value that satisfies the constraint $\psi_{vw(v),v}$; note that such a value may not exist as $\psi_{vw(v),v}$ is a partial permutation or may not belong to ϕ_v.

We use this observation in the following two rules. First, since we are looking for a solution interrogated by S, we can immediately extend Ψ to vertices in $N(\Psi) \cap S$.

RULE 4.2 (extension rule). *If there exists a vertex $v \in N(\Psi) \cap S$ with a neighbor $w(v)$ satisfying $\Psi(w(v)) \in \Sigma$, then assign to v the unique value that satisfies the constraint $\psi_{vw(v),v}$.*

Note that we can check if the extension rule can be applied in $O(sn^2)$ time, and in total it cannot be applied more than n times in a single root-to-leaf path in the bounded search tree.

For the second rule, we observe in the next lemma that if Ψ^N does not exist (i.e., it violates some constraint), then there is a witnessing contradiction on at most two vertices of $N(\Psi)$.

LEMMA 38. *If Ψ^N is undefined or violates some constraint (and hence is not a labeling), then there exists a set $B \subseteq N(\Psi)$ of size at most two such that any labeling extending Ψ needs to delete at least one vertex of B. Moreover, such a set B can be found on $O(sn^2)$ time.*

Proof. First, if for some vertex $v \in N(\Psi)$, the value $\Psi^N(v)$ is undefined (there is no value matching $\Psi(w(v))$ in the constraint $\psi_{vw(v),v}$) or such a value does not belong to ψ_v, we can take $B = \{v\}$. Otherwise, Ψ^N needs to violate some edge constraint, say, $\psi_{vu,v}$ for some $v, u \in N[\Psi]$. As Ψ satisfies all edge constraints, either u or v is not in $\mathrm{dom}(\Psi)$, assume then $v \notin \mathrm{dom}(\Psi)$. If $u \in \mathrm{dom}(\Psi)$, then $B = \{v\}$ satisfies the conditions of the lemma: any assignment of a value from Σ to v would violate either $\psi_{vu,v}$ or $\psi_{vw(v),v}$. If $u \notin \mathrm{dom}(\Psi)$, then $B = \{u, v\}$ satisfies the conditions of the lemma: however, we assign values from Σ to u and v, so we would either violate $\psi_{vw(v),v}$, violate $\psi_{uw(u),u}$, or assign the values as in Ψ^N and violate $\psi_{uv,v}$.

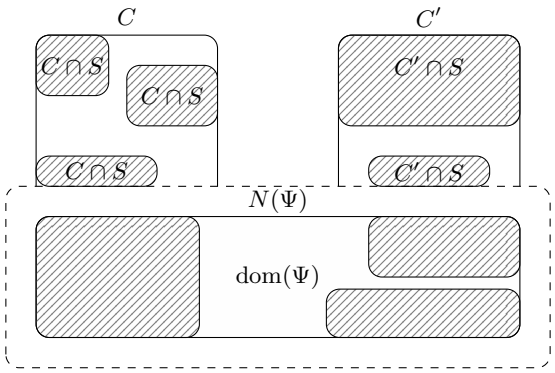

FIG. 3. *Situation before the last step of the branching algorithm. The stripped rectangles represent the set S.*

We note that we can compute Ψ^N and find a constraint not satisfied by Ψ^N in $O(sn^2)$ time. ∎

RULE 4.3 (neighborhood branching rule). *Find Ψ^N. If it is undefined or is not a labeling, apply Lemma 38 obtaining a set B, and branch into $|B|$ subcases, deleting one of the vertices of B (i.e., assigning it value ☠ in Ψ).*

Unfortunately, the neighborhood branching rule is not always applicable. However, we now show that if it is not applicable, then we can make decisions on different components of $G \setminus N[\Psi]$ nearly independently. Henceforth we assume that Ψ^N is well-defined and is a labeling; see Figure 3 for an illustration.

LEMMA 39. *Let C be a vertex set of an arbitrary connected component of $G \setminus N[\Psi]$ and let Ψ_0 be a labeling of $V(G)$ that extends Ψ and is interrogated by S. Then either*

- *$C \subseteq \mathtt{big}(\Psi_0)$, in particular no vertex of C is deleted by Ψ_0, or*
- *$C \setminus S$ is exactly the set of vertices from C that are deleted by Ψ_0 and all vertices of $N(C)$ are deleted by Ψ_0.*

Furthermore, if one can extend Ψ^N to a labeling Ψ^C of $N[\Psi] \cup C$ without deleting any new vertex, then a function Ψ_1 defined as equal Ψ^C on C and equal Ψ_0 on $V(G) \setminus C$ is a labeling of $V(G)$ extending Ψ that deletes not more vertices than Ψ_0 does.

Proof. For the first part of the lemma, assume there exists $v \in C \cap \mathtt{big}(\Psi_0)$. From the connectivity of $G[C]$, we have that either $C \subseteq \mathtt{big}(\Psi_0)$ or there exists $uw \in E(G[C])$ with $\Psi_0(w) =$ ☠ and $u \in \mathtt{big}(\Psi_0)$. Recall that S interrogates Ψ_0; from the last property of the interrogating set we infer that w is adjacent to a big stain of S, a contradiction with the fact that all big stains are in $\mathrm{dom}(\Psi)$ but $w \notin N[\Psi]$. Therefore either $C \subseteq \mathtt{big}(\Psi_0)$ or $C \cap \mathtt{big}(\Psi_0) = \emptyset$.

In the latter case, pick any $v \in C$ that is not deleted by Ψ_0. As $C \cap \mathtt{big}(\Psi_0) = \emptyset$, v is contained in a connected component $C(v)$ of $G \setminus \Psi_0^{-1}($☠$)$ with at most q vertices. By the second property of the interrogating set S, $C(v)$ is a small stain of S. Consequently, we have $C(v) \subseteq C$, as any vertex of $C(v) \cap N(\Psi)$ would be amenable to the extension rule.

Consider now a vertex $w \in N(C)$ with a neighbor $u \in C$. Assume that w is not deleted by Ψ_0, that is, $\Psi_0(w) \neq$ ☠. If $w \in \mathtt{big}(\Psi_0)$, then $\Psi_0(u) =$ ☠ as we have assumed that $C \cap \mathtt{big}(\Psi_0) = \emptyset$. However, then, from the last property of the interrogating set, we infer that u is adjacent to a big stain of S, a contradiction with the fact that all big stains are in $\mathrm{dom}(\Psi)$ but $u \notin N[\Psi]$. In the other case,

if $w \notin \texttt{big}(\Psi_0)$, then the extension rule is applicable to w if $w \in N(\Psi)$ or to u if $w \in \mathrm{dom}(\Psi)$. As we have reached a contradiction in all subcases, we infer that w is deleted by Ψ_0. Since the choice of w was arbitrary, we have that all vertices of $N(C)$ are deleted by Ψ_0. This finishes the proof of the first part of the lemma.

For the second part of the lemma, first note that Ψ_1 deletes a subset of the vertices deleted by Ψ_0, as Ψ^C does not delete any vertex of C. Furthermore, since both Ψ_0 and Ψ_C are labelings, Ψ_1 satisfies all vertex contraints.

As for edge constraints, clearly Ψ_1 satisfies all edge constraints for edges completely contained either in $G[C]$ or $G \setminus C$. Consider now an edge uv with $u \in C$ and $v \in N(C)$. If $\Psi_0(v) = \text{☠}$, then we are done. Otherwise, as C is a connected component of $G \setminus N[\Psi]$, we have $v \in N(\Psi)$. Since Ψ_0 extends Ψ, we have $\Psi_0(v) = \Psi^N(v)$. Since Ψ^C extends Ψ^N and is a labeling, we have that Ψ_1 satisfies the constaint on the edge uv. This completes the proof of the lemma. □

Consider now the following step: for every connected component C of $G \setminus N[\Psi]$ we apply Lemma 23 to check if there exists an extension Ψ^C of Ψ^N to $N[\Psi] \cup C$. Note that if such an extension exists, Lemma 39, together with the fact that Ψ^N exists and is a labeling, implies that the union of all such labelings Ψ^C is a labeling of $V(G)$ that extends Ψ and does not delete any new vertex, and the finishing rule is applicable. Note that such an output labeling may not be interrogated by S, but it is not an issue at this point. Otherwise, we can use the limited options given by Lemma 39 to perform branching.

RULE 4.4 (small stains rule). *For every connected component C of $G \setminus N[\Psi]$ apply Lemma 23 to check if there exists an extension Ψ^C of Ψ^N to $N[\Psi] \cup C$. Since the finishing rule is not applicable, there exists a component for which such an extension does not exist; we pick such a component C and branch into at most $s+1$ cases.*

1. *Apply Lemma 23 to find at most s deletion-free labelings of $G[C]$, and for every such labeling consider a branch with Ψ extended with this labeling.*
2. *Furthermore, in an additional branch assign ☠ to every vertex of $N(C) \cup (C \setminus S)$, and for every connected component of $G[C \cap S]$ apply Lemma 23 to find a deletion-free labeling of this component.*

First, note that the small stains rule is applicable in $O(s^2n^2)$ time.

The first option of the small stains rule corresponds to the case $C \subseteq \texttt{big}(\Psi_0)$ of Lemma 39. Note that the inexistence of Ψ^C implies that in every subcase of the first option, the neighborhood branching rule will execute on some vertices of $N(C)$, leading to an increase in the number of deleted vertices.

The second option corresponds to the second case of Lemma 39. Note that the inexistence of Ψ^C implies that either this branch is not executed at all (due to the fact that some component of $G[C \cap S]$ does not admit any deletion-free labeling) or either $C \setminus S$ or $N(C)$ is nonempty, leading to an increase in the number of deleted vertices. Note that after all vertices $N(C) \cup (C \setminus S)$ are assigned ☠, every connected component C' of $G[C \setminus S]$ has all its neighbors deleted and can freely choose any deletion-free labeling output by Lemma 23.

Consequently, the small stains rule, coupled with a possible following application of the neighborhood branching rule, gives at most $2s+1$ subcases, and in every such subcase at least one new vertex is deleted. As the small stains rule is always applicable if no previous rule is applicable, we have concluded the description of the branching algorithm. Recall that we terminate the algorithm after more than k vertices are deleted. As each rule is applicable in $O(s^2n^2)$ time and the number of rule applications before reaching a leaf is bounded by n the whole branching algorithm

works in $O(2^{O(k \log s)} n^3)$ time. This finishes the description of the fixed-parameter algorithm for Border N-ULC and the proof of Theorem 2.

5. The algorithm for Steiner Cut. This section is devoted to the proof of Theorem 4, i.e., to the Steiner Cut problem, parameterized by the size of the cutset.

Steiner Cut	**Parameter:** k
Input: A graph G, a set of terminals $T \subseteq V(G)$, and integers s and k.	
Question: Does there exist a set X of at most k edges of G, such that in $G \setminus X$ at least s connected components contain at least one terminal?	

For a Steiner Cut instance (G, T, s, k), a set $X \subseteq E(G)$ is called a *solution* if $|X| \leq k$ and $G \setminus X$ contains at least s connected components that contain at least one terminal.

First, observe that by Lemma 12 in $O(kn^2)$ time we can ensure that the graph G has $O(kn)$ edges, by removing the edges outside of the set E_0. Correctness of this step follows from the fact that in this operation all cuts of size at most k are preserved and moreover no new cut of size at most k appears, since for each of the removed edges at least $k+1$ edge-disjoint paths between its endpoints remain. We are going to use the assumption that there are $O(kn)$ edges in the graph during the course of our algorithm. To this end, we use Lemma 12 after each reduction, thus always ensuring that the graph has at most $O(kn)$ edges, where n is the current number of vertices. We note that the only reason for using Lemma 12 is caring about the polynomial factor.

Second, we observe that in the Steiner Cut problem we may assume that the graph G is connected. Indeed, otherwise we may add to G a clique on $k+2$ vertices (so that the clique cannot be split with removal of k edges), make all vertices of the clique adjacent to exactly one vertex of each connected component of G, and decrease s by the number of connected components of G containing a terminal minus one.

If G is connected, removal of k edges may lead to at most $k+1$ connected components. Thus, we may assume that $s \leq k+1$, as otherwise the answer is trivially negative.

Moreover, in the course of the algorithm we repetitively contract some edges of G. In the process of contraction, we remove loops, but we keep multiple edges. Thus we allow G to be a multigraph with multiple edges, but without loops. Note that if an edge $uv \in E(G)$ has multiplicity more than k, the aforementioned sparsification process using Lemma 12 reduces the multiplicity down to at most $k+1$.

The algorithm performs a number of steps. Description of each step is accompanied by discussion of correctness and analysis of running time.

5.1. Operations on the input graph. The basic operation the algorithm performs on the graph is an edge contraction. As mentioned in the last section, we assume that after performing a series of contractions we reduce all multiplicities of the multiedges that exceed $k+1$ down to $k+1$. Moreover, if in G a set of terminals $T \subseteq V(G)$ is given, if $T \cap \{u, v\} \neq \emptyset$, we replace T with $(T \setminus \{u, v\}) \cup \{w_{uv}\}$, i.e., we put w_{uv} into T iff u or v belongs to T.

The following straightforward corollary of Lemma 11 shows when we may contract an edge of G in the Steiner Cut case.

Lemma 40. *Let $\mathcal{I} = (G, T, s, k)$ be a* Steiner Cut *instance, let $D \subseteq E(G)$ and let $\mathcal{I}' = (G', T', s, k)$ be the instance $\mathcal{I}$ with the edges D contracted in an arbitrary order. Then,*

1. *any solution X to $\mathcal{I}'$ is a solution to $\mathcal{I}$ as well (recall that we treat $E(G')$ as a subset of $E(G)$);*
2. *for any solution X to $\mathcal{I}$ that is disjoint with D, the set*

$$X' = \{\iota(u)\iota(v) : uv \in X, \iota(u) \neq \iota(v)\} \subseteq E(G')$$

is a solution to $\mathcal{I}'$.

We also use the notion of *identifying* two vertices.

DEFINITION 41. *Given a multigraph G and two vertices u, v,* identification *of u and v is the operation of adding an edge uv and contracting it.*

As identification is modeled by edge addition and contraction, we apply the same terminology also to this notion.

5.2. Borders and recursive understanding. In the border problem the graph is additionally equipped with at most $2k$ border terminals T_b. For a border STEINER CUT instance $\mathcal{I}_b = (G, T, k, T_b)$, we need to remember an equivalence relation $\mathcal{R}_b$ on T_b that corresponds to how the border terminals are to be distributed among connected components, a set $Y_b \subseteq T_b$, that carries information about which border terminal is in connected component with some terminal of T, and an integer s_b, which means that in $\mathcal{I}_b$ we are to obtain s_b connected components that contain a terminal. Formally speaking, we define $\mathbb{P}(\mathcal{I}_b)$ as the set of all triples $\mathcal{P} = (\mathcal{R}_b, Y_b, s_b)$, where $\mathcal{R}_b$ is an equivalence relation on T_b, $Y_b \subseteq T_b$ and $0 \leq s_b \leq k+1$ is an integer. Moreover, we require that if $(u, v) \in \mathcal{R}_b$, then $u \in Y_b$ iff $v \in Y_b$.

We say that a set $X \subseteq E(G)$ is *a solution* to $(\mathcal{I}_b, \mathcal{P})$ for a triple $\mathcal{P} = (\mathcal{R}_b, Y_b, s_b) \in \mathbb{P}(\mathcal{I}_b)$ if

- two border terminals $u, v \in T_b$ are in the same connected component of $G \setminus X$ iff $(u, v) \in \mathcal{R}_b$;
- for any border terminal $u \in T_b$, the connected component of $G \setminus X$ which contains u contains a vertex of T iff $u \in Y_b$;
- $G \setminus X$ contains exactly s_b connected components that contain a vertex of T;
- $|X| \leq k$.

We formally define the border problem as follows.

BORDER STEINER CUT
Input: A STEINER CUT instance $\mathcal{I} = (G, T, s, k)$ with G being connected, and a set $T_b \subseteq V(G)$ of size at most $2k$; denote $\mathcal{I}_b = (G, T, k, T_b)$.
Output: For each $\mathcal{P} \in \mathbb{P}(\mathcal{I}_b)$ output a solution $\texttt{sol}_{\mathcal{P}} = X_{\mathcal{P}}$ to $(\mathcal{I}_b, \mathcal{P})$ with minimum possible $|X_{\mathcal{P}}|$, or $\texttt{sol}_{\mathcal{P}} = \perp$ if such a solution does not exist.

Observe that BORDER STEINER CUT generalizes STEINER CUT as we may ask for $T_b = \emptyset$ and check the value of a solution consistent with $(\emptyset, \emptyset, s)$, as we can assume that after removing the minimum size solution there are exactly s connected components containing a terminal.

Note that $|\mathbb{P}(\mathcal{I}_b)| \leq (2k)^{2k} \cdot 2^{2k} \cdot (k+2)$, as there are at most $|T_b|^{|T_b|}$ choices for an equivalence relation $\mathcal{R}_b$, $2^{|T_b|}$ choices for Y_b, and $k+2$ choices for the value of s_b. Denote

$$q = k(2k)^{2k}2^{2k}(k+2) + 1 = 2^{O(k \log k)}.$$

Let $\mathcal{I}_b = (G, T, k, T_b)$ be the given instance of BORDER STEINER CUT. Assume that G admits a (q, k)-good edge separation (V_1, V_2).

As V_1 and V_2 are disjoint, at least one of them contains at most k border terminals from T_b. Without loss of generality assume that $|T_b \cap V_1| \leq k$. Let $\widehat{G} = G[V_1]$, $\widehat{T} = T \cap V_1$, and $\widehat{T}_b = (T_b \cap V_1) \cup N_G(V_2)$. Consider an instance $\widehat{\mathcal{I}}_b = (\widehat{G}, \widehat{T}, k, \widehat{T}_b)$. Note that $\widehat{\mathcal{I}}_b$ is a correct instance of Border Steiner Cut, as $|(T_b \cap V_1) \cup N_G(V_2)| \leq 2k$. Apply the algorithm recursively to the instance $\widehat{\mathcal{I}}_b$ (note that it is a strictly smaller instance as the vertex set V_2 is removed) and let $U(\widehat{\mathcal{I}}_b)$ be the set of edges that are contained in any output solution for any behavior on the border terminals of $\widehat{\mathcal{I}}_b$. Observe that $|U(\widehat{\mathcal{I}}_b)| \leq q-1$. Let $R = E(\widehat{G}) \setminus U(\widehat{\mathcal{I}}_b)$ be the set of remaining edges in $\widehat{G} = G[V_1]$. Contract the edges of R in G to obtain the new graph G' with terminals T' and border terminals T'_b. Let V'_1 be the set of vertices of G' onto which vertices of V_1 were contracted. Observe that $G'[V'_1]$ is still connected as a contraction of a connected graph, and has at most $q-1$ edges, as $|U(\widehat{\mathcal{I}}_b)| \leq q-1$. Therefore, $|V'_1| \leq q$. The contraction induces a mapping $\iota : V(G) \to V(G')$ that maps every vertex of G to the vertex of G' onto which it is contracted.

The following lemma is useful in arguing safeness of the described operation.

Lemma 42. *Let $\mathcal{P} \in \mathbb{P}(\mathcal{I}_b)$ and let $X_{\mathcal{P}}$ be a solution to $(\mathcal{I}_b, \mathcal{P})$. Then there exists a second solution $X'_{\mathcal{P}}$ to $(\mathcal{I}_b, \mathcal{P}')$ such that $|X'_{\mathcal{P}}| \leq |X_{\mathcal{P}}|$ and additionally $X'_{\mathcal{P}} \cap R = \emptyset$.*

Proof. Consider the graph $\widehat{G} = G[V_1]$ and the set $X_{\mathcal{P}} \cap E(\widehat{G})$. Define

- $\widehat{\mathcal{R}}_b$ to be an equivalence relation on $\widehat{T}_b$ such that for any $u, v \in \widehat{T}_b$ we have $(u,v) \in \widehat{\mathcal{R}}_b$ iff u and v are in the same connected component of $\widehat{G} \setminus X_{\mathcal{P}}$;
- $\widehat{Y}_b$ to be a set of those vertices $v \in \widehat{T}_b$, such that the connected component of $\widehat{G} \setminus X_{\mathcal{P}}$ that contains v contains a terminal from $\widehat{T} = T \cap V_1$ as well;
- $\widehat{s}_b$ to be the number of connected components of $\widehat{G} \setminus X_{\mathcal{P}}$ that contain a vertex of $\widehat{T}$.

Let $\widehat{\mathcal{P}} = (\widehat{\mathcal{R}}_b, \widehat{Y}_b, \widehat{s}_b)$. Clearly, $\widehat{\mathcal{P}} \in \mathbb{P}(\widehat{\mathcal{I}}_b)$ and $X_{\mathcal{P}} \cap E(\widehat{G})$ is a solution to $(\widehat{\mathcal{I}}_b, \widehat{\mathcal{P}})$ (note that $\widehat{s}_b \leq |X_{\mathcal{P}} \cap E(\widehat{G})| + 1 \leq k+2$, as $\widehat{G}$ is connected). Therefore $\mathtt{sol}_{\widehat{\mathcal{P}}} = \widehat{X}_{\widehat{\mathcal{P}}} \neq \bot$, that is, there exists a solution $\widehat{X}_{\widehat{\mathcal{P}}}$ to $(\widehat{\mathcal{I}}_b, \widehat{\mathcal{P}})$, such that $|\widehat{X}_{\widehat{\mathcal{P}}}| \leq |X_{\mathcal{P}} \cap E(\widehat{G})|$ and $\widehat{X}_{\widehat{\mathcal{P}}} \cap R = \emptyset$.

Define $X'_{\mathcal{P}} = (X_{\mathcal{P}} \setminus E(\widehat{G})) \cup \widehat{X}_{\widehat{\mathcal{P}}}$. Clearly $|X'_{\mathcal{P}}| \leq |X_{\mathcal{P}}|$. To finish the proof of the lemma we need to show that $X'_{\mathcal{P}}$ is a solution to $(\mathcal{I}_b, \mathcal{P})$.

First, we show the following claim: for any $u, v \in T_b \cup \widehat{T}_b$, u and v are in the same connected component of $G \setminus X_{\mathcal{P}}$ iff u and v are in the same connected component of $G \setminus X'_{\mathcal{P}}$. We show only a proof in one direction, as proofs in both directions are totally symmetric and use only the facts that $X_{\mathcal{P}} \setminus E(\widehat{G}) = X'_{\mathcal{P}} \setminus E(\widehat{G})$ and that both $X_{\mathcal{P}} \cap E(\widehat{G})$ and $X'_{\mathcal{P}} \cap E(\widehat{G})$ are solutions to $(\widehat{\mathcal{I}}_b, \widehat{\mathcal{P}})$.

Let $u, v \in T_b \cup \widehat{T}_b$ be two vertices that are connected by a path P in $G \setminus X_{\mathcal{P}}$. Let $u = v_0, v_1, \ldots, v_r = v$ be the sequence of all vertices on P that belong to $T_b \cup \widehat{T}_b$, in the order they appear on P. To prove the claim we need to show that for any $0 \leq i < r$, the vertices v_i and v_{i+1} belong to the same connected component of $G \setminus X'_{\mathcal{P}}$. By definition, as $N_G(V_2) \subseteq \widehat{T}_b$, the subpath P_i of P between v_i and v_{i+1} lies entirely in $\widehat{G}$ or entirely in $G \setminus E(\widehat{G})$. In the first case, we infer that $v_i, v_{i+1} \in \widehat{T}_b$, $(v_i, v_{i+1}) \in \widehat{\mathcal{R}}_b$ and v_i and v_{i+1} are in the same connected component of $\widehat{G} \setminus X'_{\mathcal{P}}$ as $X'_{\mathcal{P}} \cap E(\widehat{G})$ is a solution to $(\widehat{\mathcal{I}}_b, \widehat{\mathcal{P}})$. In the second case, we infer that v_i and v_{i+1} are in the same connected component of $(G \setminus E(\widehat{G})) \setminus X'_{\mathcal{P}}$, as $X_{\mathcal{P}} \setminus E(\widehat{G}) = X'_{\mathcal{P}} \setminus E(\widehat{G})$. This finishes the proof of the claim.

As a straightforward corollary of the aforementioned claim, we infer that for any $u, v \in T_b$, we have $(u, v) \in \mathcal{R}_b$ iff u and v are in the same connected component of $G \setminus X'_{\mathcal{P}}$. We now show that for any $v \in T_b \cup \widehat{T}_b$, its connected component of $G \setminus X_{\mathcal{P}}$ contains a vertex of T iff its connected component of $G \setminus X'_{\mathcal{P}}$ contains a vertex of T. The proofs in both directions are again totally symmetric, and thus we present only the forward implication.

Let P be a path that connects the vertex v with a terminal $w \in T$ in $G \setminus X_{\mathcal{P}}$. Let u be the last (closest to w) vertex on P that belongs to $T_b \cup \widehat{T}_b$ (as $v \in T_b \cup \widehat{T}_b$, such a vertex u exists). From the claim we infer that u and v are in the same connected component of $G \setminus X'_{\mathcal{P}}$. Let P_u be the subpath of P from u to w. We have two cases: P_u is contained either in $\widehat{G}$ or in $G \setminus E(\widehat{G})$. In the first case, we infer that $u \in \widehat{T}_b$, $u \in \widehat{Y}_b$ (as $X_{\mathcal{P}} \cap E(\widehat{G})$ is a solution to $(\widehat{\mathcal{I}}_b, \widehat{\mathcal{P}})$) and that the connected component of $\widehat{G} \setminus X'_{\mathcal{P}}$ that contains u contains a vertex from $\widehat{T} = T \cap V_1$ (not necessarily the vertex w). In the second case, we infer that the path P_u is present in $(G \setminus E(\widehat{G})) \setminus X'_{\mathcal{P}}$. This finishes the proof that $v \in Y_b$ iff there exists a terminal in the connected component of $G \setminus X'_{\mathcal{P}}$ that contains v.

To finish the proof of the lemma we need to show that the number of connected components of $G \setminus X'_{\mathcal{P}}$ that contain a terminal equals s_b. To this end, we partition the connected components of $G \setminus X_{\mathcal{P}}$ and $G \setminus X'_{\mathcal{P}}$ containing terminals into three types:

1. those that contain a vertex from $T_b \cup \widehat{T}_b$;
2. those that do not contain such a vertex but are contained in $\widehat{G}$;
3. and the rest—those that do not contain a vertex from $T_b \cup \widehat{T}_b$ and are contained in $G \setminus V_1$.

Our goal is to prove that for each type, the numbers of connected components containing terminals of corresponding types in $G \setminus X_{\mathcal{P}}$ and $G \setminus X'_{\mathcal{P}}$ are equal.

For the first type, the claim is a straightforward corollary of already proven facts that (i) any two vertices $u, v \in T_b \cup \widehat{T}_b$ are in the same connected component of $G \setminus X_{\mathcal{P}}$ iff they are in the same connected component of $G \setminus X'_{\mathcal{P}}$ and (ii) for every vertex $u \in T_b \cup \widehat{T}_b$, the connected component of $G \setminus X_{\mathcal{P}}$ containing u contains a terminal iff the connected component of $G \setminus X'_{\mathcal{P}}$ containing u contains a terminal.

For the second type, note that we are to count the number of connected components of $\widehat{G} \setminus X_{\mathcal{P}}$ and $\widehat{G} \setminus X_{\mathcal{P}}$ that do not contain a vertex of $T_b \cup \widehat{T}_b$, or, equivalently, $\widehat{T}_b$, but contain a terminal from $\widehat{T}$. As both $X_{\mathcal{P}} \cap E(\widehat{G})$ and $X'_{\mathcal{P}} \cap E(\widehat{G})$ are solutions to $(\widehat{\mathcal{I}}_b, \widehat{\mathcal{P}})$, this number is equal to $\widehat{s}_b$ minus the number of equivalence classes of $\widehat{\mathcal{R}}_b$ that contain vertices from Y_b.

For the third type, recall that $X_{\mathcal{P}} \setminus E(\widehat{G}) = X'_{\mathcal{P}} \setminus E(\widehat{G})$, so the sets of connected components of the third type in $G \setminus X_{\mathcal{P}}$ and $G \setminus X'_{\mathcal{P}}$ are equal. This finishes the proof of the lemma. ∎

We now show that the output for the new instance $\mathcal{I}'_b = (G', T', k, T'_b)$ can be easily transformed to the output for the original instance $\mathcal{I}_b$.

LEMMA 43. *Let $(\mathtt{sol}'_{\mathcal{P}})_{\mathcal{P} \in \mathbb{P}(\mathcal{I}'_b)}$ be a correct output for the instance $\mathcal{I}'_b$. For any $\mathcal{P} = (\mathcal{R}_b, Y_b, s_b) \in \mathbb{P}(\mathcal{I}_b)$ define $\mathtt{sol}_{\mathcal{P}}$ as follows:*

- *If ι maps two border terminals $u, v \in T_b$ with $(u, v) \notin \mathcal{R}_b$ to the same vertex of T'_b, then we take $\mathtt{sol}_{\mathcal{P}} = \bot$.*
- *Otherwise, we define $\mathcal{P}' = (\mathcal{R}'_b, Y'_b, s_b)$ as follows: $(u', v') \in \mathcal{R}'_b$ if $\iota^{-1}(u') \cap T_b$ are contained in the same equivalence class as $\iota^{-1}(v') \cap T_b$, and $v' \in Y'_b$ if $\iota^{-1}(v') \cap T_b \subseteq Y_b$; and take $\mathtt{sol}_{\mathcal{P}} = \mathtt{sol}'_{\mathcal{P}'}$.*

Then the sequence $(\mathtt{sol}_{\mathcal{P}})_{\mathcal{P} \in \mathbb{P}(\mathcal{I}_b)}$ is a correct output to the instance $\mathcal{I}_b$.

Proof. The lemma is a straightforward corollary of the contraction properties of Lemmas 11 and 40, as well as Lemma 42. ∎

We can now formally define the first step of the algorithm.

STEP 5.1. *Using Lemma* 14 *we check whether* G *admits a* (q,k)*-good edge separation. If this is not the case, we proceed to the second phase, i.e., the high connectivity phase. Otherwise let* (V_1, V_2) *be this separation and without loss of generality assume that* $|T_b \cap V_1| \leq k$*. Construct the instance* $\widehat{\mathcal{I}}_b = (G[V_1], T \cap V_1, k, (T_b \cap V_1) \cup N_G(V_2))$*, apply Lemma* 12 *to it, solve it recursively, and compute* $U(\widehat{\mathcal{I}}_b)$*, the set of edges that appear in any solution given in the output. Contract all the remaining edges of* $G[V_1]$ *in* G *to obtain new graph* G' *with terminals* T' *and border terminals* T'_b*. Define* $\mathcal{I}'_b = (G', T', k, T'_b)$*; recall that a vertex belongs to (border) terminals iff some (border) terminal was contracted onto it. Apply Lemma* 12 *to* $\mathcal{I}'_b$*, recursively solve the instance* $\mathcal{I}'_b$*, and transform the output according to Lemma* 43.

Let us now estimate the running time. First, we spend $O(2^{O(k^2 \log k)} n^3 \log n)$ time to check whether there exists a (q,k)-good edge separation. We apply the algorithm recursively to the instance $\widehat{\mathcal{I}}_b$, which has n' vertices for some $q+1 \leq n' \leq n-q-1$. Construction of the instance $\widehat{\mathcal{I}}_b$ takes $O(kn)$ time, construction of $U(\widehat{\mathcal{I}}_b)$ takes $O(2^{O(k \log k)} n)$ time, and construction of the instance $\mathcal{I}'_b$ takes $O(kn)$ time. Then, we apply the algorithm recursively to the instance $\mathcal{I}'_b$ that has at most $n - n' + q$ vertices. Therefore, we can derive the following recurrential inequality:

$$T(n) \leq \max_{q+1 \leq n' \leq n-q-1} \Big(O(2^{O(k^2 \log k)} n^3 \log n) + T(n') + T(n - n' + q) \Big). \tag{2}$$

Note that the function $p(t) = t^3 \log t$ is convex, so the maximum of the expression is attained at one of the ends. A straightforward inductive check of both of the ends proves that we have indeed the claimed bound on the complexity, i.e., $O(2^{O(k^2 \log k)} n^4 \log n)$.

Observe that if we use the randomized algorithm for finding good edge separations from Lemma 16, we obtain $T(v) \leq \tilde{O}(2^{O(k^2 \log k)} n^2)$ time complexity with success probability at least $(1 - 1/n)$, since the graph is partitioned using good edge separations less than n times.

5.3. Brute-force approach. If the graph returned by Step 5.1 turns out to be small, we apply a brute-force approach. In this section we describe this step formally.

LEMMA 44. *A correct output to a* BORDER STEINER CUT *instance* $\mathcal{I}_b = (G, T, k, T_b)$ *can be computed in* $O(2^{O(k \log k)} n^{2k+2})$ *time.*

Proof. For every $\mathcal{P} \in \mathbb{P}(\mathcal{I}_b)$, and for every set $X \subseteq E(G)$ of at most k edges that, for all $u, v \in V(G)$, takes either all or zero edges uv, in $O(n^2)$ time we verify whether X is a solution to $(\mathcal{I}_b, \mathcal{P})$. The time bound follows from the fact that $|\mathbb{P}(\mathcal{I}_b)| \leq 2^{O(k \log k)}$ and there are at most $(k+1)n^{2k}$ choices of the set X. ∎

We are ready to provide the step of the algorithm that finishes resolving the problem, providing that the graph is sufficiently small.

STEP 5.2. *If* $|V(G)| \leq (k+1)q$*, then apply Lemma* 44 *to resolve the given* BORDER STEINER CUT *instance* $\mathcal{I}_b = (G, T, k, T_b)$.

The correctness of this step is obvious, while from Lemma 44 we find that the running time is $O(2^{O(k^2 \log k)})$ as $q = 2^{O(k \log k)}$. Therefore, from now on we can assume that $|V(G)| > (k+1)q$.

5.4. High connectivity phase. We now show how to solve Border Steiner Cut in $\tilde{O}(2^{O(k^2 \log k)} n \log n)$ time for the remaining case, when the graph G does not admit a (q,k)-good edge separation yet is still too big to apply brute force. We need to output answers for all the possible triples $\mathcal{P} \in \mathbb{P}(\mathcal{I}_b)$. We iterate through all such $\mathcal{P}$; note that this gives only $2^{O(k \log k)}$ overhead in the running time. Therefore, from now on we may assume that $\mathcal{P} = (\mathcal{R}_b, Y_b, s_b)$ is fixed.

First, we make a quick check for whether an empty deletion set is sufficient for our needs. We formally need this step in order to be able to use nontriviality of the solution in some technical reasonings.

Step 5.3. *Given* Border Steiner Cut *instance* $\mathcal{I}_b = (G, T, k, T_b)$ *and* $\mathcal{P} \in \mathbb{P}(\mathcal{I}_b)$*, verify in* $O(kn)$ *time whether* $\emptyset$ *is a solution to* $(\mathcal{I}_b, \mathcal{P})$*. If this is the case, output* $\mathtt{sol}_{\mathcal{P}} = \emptyset$*.*

The described step requires $O(kn)$ time and its correctness is obvious. From now on we may assume that the minimum deletion set is nonempty.

5.4.1. Interrogating sets. We now prepare ourselves to use Lemma 1 to extract more structure of the graph G.

Definition 45. *Let* $X \subseteq E(G)$*,* $1 \leq |X| \leq k$*, and let* $C_0, C_1, \ldots, C_\ell$ *be connected components of* $G \setminus X$*, where, due to Lemma* 15*,* $\ell \leq k$ *and* $|V(C_i)| \leq q$ *for* $i \geq 1$*. We say that a set of edges* $S \subseteq E(G)$ interrogates X *if the following properties are satisfied:*

- $X \cap S = \emptyset$*;*
- *for every component* C_i*,* $i \geq 1$*,* S *contains a spanning tree of* C_i*;*
- *for every vertex* $u \in V(C_0) \cap V(X)$*,* u *is contained in a connected component of* $(V(G), S)$ *of size at least* $q+1$*.*

Note that the first property together with $|V(C_i)| \leq q$ for $i \geq 1$ implies that the connected component considered in the third property has to be entirely contained in C_0. We now prove that a sufficiently large family given by Lemma 1 contains a set interrogating a solution.

Lemma 46. *Let* $\mathcal{F}$ *be a family obtained by an application of the algorithm of Lemma* 1 *for universe* $U = E(G)$ *and constants* $a = 3qk$ *and* $b = k$*. Then, for any* $X \subseteq E(G)$ *with* $1 \leq |X| \leq k$*, there exists a set* $S \in \mathcal{F}$ *that interrogates* X*.*

Proof. Let $C_0, C_1, \ldots, C_\ell$ be connected components of $G \setminus X$, where, due to Lemma 15, $\ell \leq k$ and $|V(C_i)| \leq q$ for $i \geq 1$. As the algorithm did not finish when performing Step 5.2, we have that $|V(G)| > (k+1)q$, so $|V(C_0)| \geq q+1$. Fix a spanning tree T_i of each component C_i. Let $A_1 = \bigcup_{i=1}^{\ell} E(T_i)$; note that $|A_1| \leq qk$. For every vertex $u \in V(C_0) \cap V(X)$ fix an arbitrary subtree T_0^u of T_0 that contains exactly $q+1$ vertices, and define $A_2 = \bigcup_{u \in V(C_0) \cap V(X)} E(T_0^u)$; this is possible due to $|V(C_0)| \geq q+1$. By Lemma 1, there exists a set $S \in \mathcal{F}$ such that $A_1 \cup A_2 \subseteq S$ and $S \cap X = \emptyset$. It follows from the construction that S interrogates X. ∎

This gives raise to the following branching step.

Step 5.4. *Using Lemma* 1 *generate family* $\mathcal{F}$ *for universe* $U = E(G)$ *and constants* $a = 3qk$ *and* $b = k$*. Branch into* $|\mathcal{F}|$ *subcases, labeled with* $S \in \mathcal{F}$*. In branch* S *we seek a solution* X *to* $(\mathcal{I}_b, \mathcal{P})$ *such that* S *interrogates* X *and, moreover,* $|X|$ *is minimum among these.*

Lemma 46 asserts that the deletion set of an optimal solution is interrogated by some set from family $\mathcal{F}$. Therefore, in order to find a solution with minimum

possible size of deletion set it suffices to take minimum over solutions given by the branches. Note that in this manner we introduce $O(2^{O(\min(q,k)\log(q+k))}\log n) = O(2^{O(k^2\log k)}\log n)$ branches. Moreover, as the family $\mathcal{F}$ can be computed in $O(2^{O(k^2\log k)}n\log n)$ time and the construction of every branch takes $O(kn)$ time, the whole branching procedure takes $O(2^{O(k^2\log k)}n\log n)$ time. We now describe the subroutine performed in each branch; let it be labeled by $S \in \mathcal{F}$. To simplify the presentation we assume that S interrogates X_0 for some minimum solution X_0 and examine what happens with X_0 during the operations performed on the graph.

Let us contract all the edges from S to obtain a new graph H_0. Let ι_0 be the mapping from $V(G)$ to $V(H_0)$ corresponding to these contractions. Then, we obtain the new graph H by identifying all the vertices $u \in H_0$ for which $|\iota_0^{-1}(u)| > q$ into a single vertex; such vertices u are called *heavy*. If there are no heavy vertices, we can safely terminate the branch, as a set that interrogates an nonempty solution must induce at least one connected component that has at least $q+1$ vertices. Otherwise, denote b the vertex resulting in their identification; we will further refer to it as the *core* vertex. Let ι_1 be the mapping from $V(H_0)$ to $V(H)$ corresponding to these identifications. Moreover, let $\iota = \iota_1 \circ \iota_0$ be the mapping from $V(G)$ to $V(H)$ corresponding to the composition of these operations.

We claim that the feasible deletion set X_0 "survives" both steps.

LEMMA 47. *Let $v, w \in V(G)$ such that $\iota(v) = \iota(w)$. Then v and w are in the same connected component of $G \setminus X$ for any set $X \subseteq E(G)$ interrogated by S.*

Proof. Assume otherwise, that is, that we have $\iota(v) = \iota(w)$ but $v \in V(C_i)$ and $w \in V(C_j)$ for $i \neq j$. Without loss of generality assume that $i \neq 0$.

Assume first that $\iota_0(v) = \iota_0(w)$. As v and w are contracted onto the same vertex, there exists a path from v to w in G that consists of edges of S. As $S \cap X = \emptyset$, this means that v and w are in the same connected component of $G \setminus X$, which is a contradiction.

Assume now that $\iota_0(v) \neq \iota_0(w)$. This means that $\iota_0(v)$ and $\iota_0(w)$ have to be identified while constructing graph H from H_0. It follows that $|\iota_0^{-1}(v)| \geq q+1$. Therefore, there are at least q vertices of G that are reachable from v via paths contained in S, hence disjoint with X. However, $i \neq 0$ so $|V(C_i)| \leq q$, which is a contradiction. ∎

From Lemma 47 we infer that all the edges of X_0 are still present in H, as from the minimality of X_0 we may assume that the edges of X_0 connect different connected components of $G \setminus X_0$. Let us define the sets of terminals T' and border terminals T'_b in H by setting $u \in T'$ iff $\iota^{-1}(u) \cap T \neq \emptyset$ and $u \in T'_b$ iff $\iota^{-1}(u) \cap T_b \neq \emptyset$.

Moreover, due to Lemma 47 and the fact that X_0 is a solution to $(\mathcal{I}_b, \mathcal{P})$, we infer that for any $v, w \in T_b$ with $\iota(v) = \iota(w)$, we have $(v,w) \in \mathcal{R}_b$ and $v \in Y_b$ iff $w \in Y_b$. Thus we can project $\mathcal{R}_b$ and Y_b on $T'_b \subseteq V(H)$, by defining a relation $\mathcal{R}'_b$ by taking $(\iota(v), \iota(w)) \in \mathcal{R}'_b$ iff $(v,w) \in \mathcal{R}_b$ and a set Y'_b by taking $\iota(v) \in Y'_b$ iff $v \in Y_b$.

Define $\mathcal{I}'_b = (H, T', k, T'_b)$ and $\mathcal{P}' = (\mathcal{R}'_b, Y'_b, s_b)$; note that $\mathcal{P}' \in \mathbb{P}(\mathcal{I}'_b)$. The next lemma can be proven by a straightforward check of the definition of the solution, as Lemma 47 asserts connected components of $G \setminus X_0$ correspond in a one-to-one manner to connected components of $H \setminus X_0$.

LEMMA 48. *The set X_0 is a solution to $(\mathcal{I}'_b, \mathcal{P}')$.*

In the same manner we can also obtain a converse implication.

LEMMA 49. *If a set $X' \subseteq E(H)$ is a solution to $(\mathcal{I}'_b, \mathcal{P}')$ of minimum possible size, then X' is a solution to $(\mathcal{I}_b, \mathcal{P})$ as well.*

Lemmas 48 and 49 justify correctness of the following step, which we now state formally.

STEP 5.5. *Contract the edges of S to obtain the graph H_0. If there are no heavy vertices in H_0, terminate the branch as S cannot interrogate any feasible deletion set. Otherwise, identify all heavy vertices into one core vertex b and denote by H the resulting graph. Define the instance $\mathcal{I}'_b = (H, T', k, T'_b)$ and $\mathcal{P}'$ in a natural manner, described in this section. Run the remaining part of the algorithm on the instance $\mathcal{I}'_b$ and triple $\mathcal{P}'$ to obtain a solution* `sol`*, which then output as* $\texttt{sol}_{\mathcal{P}}$.

Note that Step 5.5 can be performed in $O(kn)$ time.

5.4.2. Connected components of $H \setminus \{b\}$ and dynamic programming. We now establish some structural properties of the behavior of X_0 in H. The goal is to limit the class of possible solutions in $\mathcal{I}'_b$ we need to search through. Note that in the constructed graph H the vertex b plays a special role, as we know that $V(X_0) \cap V(C_0) \subseteq \iota^{-1}(b)$.

Let $B'_1, B'_2, \ldots, B'_p$ be the components of $H \setminus \{b\}$ and let $B_i = H[V(B'_i) \cup \{b\}]$ for $i = 1, 2, \ldots, p$. Observe that B_i are connected and edge-disjoint and b separates them. Moreover, we can compute them in $O(kn)$ time. We now claim that for each component B_i, the solution either takes $E(B_i)$ entirely or is disjoint with it.

LEMMA 50. *For each $i = 1, 2, \ldots, p$, either $E(B_i) \cap X_0 = \emptyset$ or $E(B_i) \subseteq X_0$.*

Proof. We consider two cases. Assume first that no vertex in $V(B'_i)$ is in the same connected component of $H \setminus X_0$ as the core b. In particular, this implies that edges connecting b with $V(B'_i)$ belong to X_0. Moreover, in G the set $\iota^{-1}(V(B'_i))$ is a union of vertex sets of some components C_i for $i \geq 1$. As S interrogates X_0, each component C_i for $i \geq 1$ is projected by ι onto a single vertex in H. Consider an edge $e \in E(B'_i)$. We infer that e connects two different connected components of $G \setminus X_0$, thus $e \in X_0$. Therefore $E(B_i) \subseteq X_0$ in this case.

Assume now that there exists a vertex $v \in V(B'_i)$ that is in the same connected component of $H \setminus X_0$ as b, i.e., $\iota^{-1}(v) \subseteq V(C_0)$. Let $w \in N_{B'_i}(v)$. As $v \neq b$, we have $|\iota^{-1}(v)| \leq q$ and $|\iota^{-1}(w)| \leq q$, so we have $vw \notin X_0$, as otherwise S does not interrogate X_0. Since the choice of v and its neighbor w was arbitrary, and since B'_i is connected, we infer that all vertices of B'_i belong to the same connected component of $H \setminus X_0$. As $\iota^{-1}(v) \subseteq V(C_0)$, we find that $\iota^{-1}(V(B'_i)) \subseteq V(C_0)$. From the minimality of X_0 we infer that $E(B_i) \cap X_0 = \emptyset$. ∎

Lemma 50 ensures us that we may seek the optimal solution among the ones that do not intersect edge sets of components B_i nontrivially. We now show how to construct the optimal solution in the remaining instance in $O(k^2 n)$ time. First, we resolve components B'_i that contain border terminals.

STEP 5.6. *We define $D \subseteq T'_b$ as follows. If $b \in T'_b$, we define D to be the equivalence class of $\mathcal{R}'_b$ that contains b. Otherwise, we branch into at most $1 + |T'_b| \leq 1 + 2k$ subcases, taking D to be an empty set or one of the equivalence classes of $\mathcal{R}'_b$. Given D, we seek a solution X where the set of border terminals being in the same connected component of $H \setminus X$ as b equals D.*

For a fixed choice of D, we may immediately resolve the connected components B'_i that contain a border terminal of T'_b. Initiate a counter $s_0 = 0$. For each component B'_i with $V(B'_i) \cap T'_b \neq \emptyset$ perform the following:

1. *If there exists a border terminal in $V(B'_i) \cap D$, as well as a border terminal in $(V(B'_i) \cap T'_b) \setminus D$, terminate this branch, as for any of the two cases given by Lemma 50, we cannot satisfy the conditions implied by $\mathcal{R}'_b$ and the set D.*
2. *If all border terminals in $V(B'_i) \cap T'_b$ belong to D, contract all edges of B_i (we have $E(B_i) \cap X = \emptyset$ in this case).*
3. *If all border terminals in $V(B'_i) \cap T'_b$ do not belong to D, include $E(B_i)$ into the constructed solution: decrease k by $|E(B_i)|$ and increase s_0 by $|V(B'_i) \cap T'|$ (by including $E(B_i)$ into a solution, we delete $|E(B_i)|$ edges and create $|V(B'_i) \cap T'|$ new connected components that contain a terminal).*

Note that Step 5.6 can be performed in $O(k^2 n)$ time and results in at most $2k+1$ branches. Its correctness is asserted by Lemma 50. After Step 5.6 is applied, all terminals of T'_b are either contracted onto b (if they belong to D) or became isolated vertices after the removal of edges included to the constructed solution. If some equivalence class of $\mathcal{R}'_b$ different than D is larger than a single vertex of H, or we do not satisfy the conditions implied by the set Y'_b for some vertex of T'_b, we may immediately reject the current branch. Otherwise, we may forget the relation $\mathcal{R}'_b$ (as all conditions imposed by it are already satisfied). Moreover we may also forget almost all information carried by the set Y'_b, except for the fact whether $D \subseteq Y'_b$. This is done in the following step.

Step 5.7. *Terminate the current branch if one of the following conditions is satisfied:*

1. *There exists an equivalence class of $\mathcal{R}'_b$ that is different from D and contains at least two border terminals.*
2. *There is a vertex $v \in T'_b \setminus D$ such that v is exactly in one of the sets T' and Y'_b.*
3. *$D \neq \emptyset$, $D \cap Y'_b = \emptyset$, and $b \in T'$ after Step 5.6 is applied.*

Otherwise, denote $\alpha = \bot$ if $D \neq \emptyset$ and $D \cap Y'_b = \emptyset$, and $\alpha = \top$ otherwise.

We note that from the minimality of X and the connectivity of G, we have that any connected component of $G \setminus X$ that does not contain a border terminal contains a terminal from T. Indeed, otherwise, if $G \setminus X$ contains a connected component C that does not contain a terminal or a border terminal, one edge incident to C may be removed from X and still X would be a solution to $(\mathcal{I}_b, \mathcal{P})$, a contradiction. Therefore, if $D = \emptyset$, we may assume that the connected component of $G \setminus X$ that contains $\iota^{-1}(b)$ contains at least one terminal.

From now on we know that all the remaining components B'_i do not contain border terminals. Without loss of generality, let $B'_1, B'_2, \ldots, B'_{p'}$ be the remaining components. For every remaining component B_i we have two numbers: $a_i = |E(B_i)|$, the cost of incorporating it to the solution, and $b_i = |V(B'_i) \cap T'|$, the number of separated terminals. Computation of a_i, b_i can be done in $O(kn)$ time. We would like to know what is the optimal number of edges needed to separate exactly $s_1 = s_b - s_0$ connected components with terminals, with the additional constraint that the connected component containing b contains a terminal iff $\alpha = \top$.

This can be solved in time $O(s_b p')$ via a standard dynamic programming routine. We create a three-dimensional table $T[j, \ell, \mathfrak{t}]$ for $\ell = 0, 1, \ldots, s_1$, $j = 0, 1, \ldots, p'$, $\mathfrak{t} = \{\bot, \top\}$ with the following meaning: $T[j, \ell, \mathfrak{t}]$ is the minimum cost of a solution

contained in the prefix $B_1, B_2, \ldots, B_j$ that separates exactly ℓ isolated vertices being terminals and $\mathfrak{t}$ denotes whether the remaining connected component with b contains a terminal different from b (or $+\infty$ if such a solution does not exist). Formally,

$$T[j,\ell,\mathfrak{t}] = \min\left\{\sum_{\gamma\in\Gamma} a_\gamma \mid \Gamma\subseteq\{1,2,\ldots,j\} \wedge \sum_{\gamma\in\Gamma} b_\gamma = \ell \wedge (\mathfrak{t}{=}\top \Leftrightarrow \sum_{\gamma\in\{1,2,\ldots,j\}\setminus\Gamma} b_\gamma{>}0)\right\}.$$

Observe that T admits the following recurrential formula (by somewhat abusing notation, we assume that cells of T with negative coordinates contain $+\infty$):

$$T[j,\ell,\bot] = \begin{cases} +\infty \text{ if } j=0 \text{ and } \ell>0,\\ 0 \text{ if } j=\ell=0,\\ \min(T[j-1,\ell,\bot], a_j + T[j-1,\ell-b_j,\bot]) \text{ if } j>0 \text{ and } b_j=0,\\ a_j + T[j-1,\ell-b_j,\bot] \text{ otherwise.}\end{cases}$$

$$T[j,\ell,\top] = \begin{cases} +\infty \text{ if } j=0,\\ \min(T[j-1,\ell,\top], a_j + T[j-1,\ell-b_j,\top]) \text{ if } j>0, \text{ and } b_j=0,\\ \min(T[j-1,\ell,\bot], T[j-1,\ell,\top], a_j + T[j-1,\ell-b_j,\top]) \text{ otherwise.}\end{cases}$$

Hence, we can fill the table T in time $O(s_1 p') = O(kn)$; the optimal value can be deduced from the cells $T[p', s_1, \bot]$ and $T[p', s_1, \top]$. Although we presented here only the algorithm for computing the optimal value, it is straightforward to implement the dynamic program so that it also maintains backlinks via which one can retrieve the corresponding set Γ from the definition of T. Thus, we can formally present the final step of our algorithm.

Step 5.8. *Compute numbers a_i and b_i and fill table T in $O(kn)$ time. Let $\mathfrak{t}\in\{\bot,\top\}$ be defined as $\mathfrak{t}=\bot$ if $\alpha=\bot$, $\mathfrak{t}=\top$ if $\alpha=\top$ and $b\notin T'$, and otherwise pick $\mathfrak{t}\in\{\bot,\top\}$ to minimize the value $T[p', s_1, \mathfrak{t}]$. Let $\Gamma\subseteq\{1,2,\ldots,p'\}$ be the set from the definition of the value $T[p', s_1, \mathfrak{t}]$, computed in $O(n)$ time by following backlinks in the table T. If the value $T[p', s_1, \mathfrak{t}]$ exceeds the remaining budget k, terminate the branch. Otherwise, incorporate the set $\bigcup_{\gamma\in\Gamma} E(B_\gamma)$ to the constructed solution.*

Step 5.8 can be performed in $O(kn)$ time and its correctness follows from the definition of the table T and the previous steps of the algorithm.

This finishes the description of the fixed-parameter algorithm for Steiner Cut and the proof of Theorem 4.

6. The algorithm for N-MWCU. In this section we show an FPT algorithm for the following generalization of the well-known Multiway Cut problem.

N-MWCU	**Parameter:** k
Input: A graph G together with a set of terminals $T\subseteq V(G)$, an equivalence relation $\mathcal{R}$ on the set T, and an integer k.	
Question: Does there exist a set $X\subseteq V(G)\setminus T$ of at most k nonterminals such that for any $u,v\in T$, the vertices u and v belong to the same connected component of $G\setminus X$ iff $(u,v)\in\mathcal{R}$?	

In other words, we are to delete at most k vertices from the graph, so that the terminals are split between connected components exactly as it is given by the equivalence relation $\mathcal{R}$. Given an N-MWCU instance $\mathcal{I}=(G,T,\mathcal{R},k)$, a set of vertices

X is called a *solution* to $\mathcal{I}$ the if $|X| \leq k$ and for any $u, v \in T$, the vertices u and v belong to the same connected component of $G \setminus X$ iff $(u, v) \in \mathcal{R}$.

Our algorithm not only resolves the N-MWCU instance, but, in the case of a YES answer, it returns a solution X with minimum possible $|X|$. This property will be used in the course of the algorithm.

6.1. Reduction of the number of equivalence classes. We now show how to reduce the number of equivalence classes of $\mathcal{R}$ in an N-MWCU instance $\mathcal{I} = (G, T, \mathcal{R}, k)$. We use a reduction similar to the one used by Razgon [53, Theorem 5].

LEMMA 51. *Let $\mathcal{I} = (G, T, \mathcal{R}, k)$ be an* N-MWCU *instance and let $v \in V(G) \setminus T$. Assume that there exist $k+2$ paths $P_1, P_2, \ldots, P_{k+2}$ in G, such that*

- *for each $1 \leq i \leq k+2$, the path P_i is a simple path that starts at v and ends at $v_i \in T$;*
- *the paths P_i have pairwise disjoint sets of vertices, except for the vertex v;*
- *for any $i \neq j$, $(v_i, v_j) \notin \mathcal{R}$.*

Then for any solution X in $\mathcal{I}$ we have $v \in X$.

Proof. Let $X \subseteq V(G) \setminus T$ with $v \notin X$ and $|X| \leq k$. As the paths P_i are disjoint (except for v), there exist two indices $1 \leq i < j \leq k+2$ such that P_i and P_j does not contain any vertex from X. A concatenation of P_i and P_j is a path from v_i to v_j that avoids X. As $(v_i, v_j) \notin \mathcal{R}$, we infer that X is not a solution to $\mathcal{I}$. ∎

LEMMA 52. *Let $\mathcal{I} = (G, T, \mathcal{R}, k)$ be an* N-MWCU *instance. For any $v \in V(G)$, we can verify if v satisfies the conditions of Lemma* 51 *in $O(kn^2)$ time.*

Proof. Consider the following auxiliary graph H. For each equivalence class $A \subseteq T$ of $\mathcal{R}$, we attach a new vertex t_A that is adjacent to all vertices of A. We make v an infinite-capacity source and each vertex t_A a unit-capacity sink; each other vertex of H has unit capacity. Clearly, v satisfies the conditions of Lemma 51 iff there exists a flow of size at least $k+2$ in H. As vertices in H have unit capacities (except for v), this can be done in $O(kn^2)$ time by the classic Ford–Fulkerson algorithm. ∎

Lemma 51 justifies the following step.

STEP 6.1. *For each $v \in V(G)$, if v satisfies the conditions of Lemma* 51*, delete v from the graph and decrease k by one; if k becomes negative by this operation, return NO. Afterward, restart the algorithm.*

By Lemma 52, each application of Step 6.1 takes $O(kn^3)$ time. As we cannot apply Step 6.1 more than k times, all applications of this step take $O(k^2n^3)$ time.

Let us now show that Step 6.1 leads to a bound on the number of equivalence classes of $\mathcal{R}$.

LEMMA 53. *Let $\mathcal{I} = (G, T, \mathcal{R}, k)$ be an* N-MWCU *instance where Step* 6.1 *is not applicable. If there exists a connected component of G that contains terminals of more than $k^2 + k$ equivalence classes of $\mathcal{R}$, then $\mathcal{I}$ is a NO-instance to* N-MWCU*.*

Proof. Let C be the vertex set of any connected component of G, and let X be a solution to $\mathcal{I}$. Fix arbitrary $v \in X \cap C$. We say that v *sees* an equivalence class A of $\mathcal{R}$ if there exists a connected component C_A of $G[C \setminus X]$ that contains a terminal of A and such that $N_G(v) \cap C_A \neq \emptyset$. Note that if v sees $k+2$ equivalence classes of $\mathcal{R}$, then in each component C_A for each equivalence class A seen by v we can find a path from v to a terminal of A. Thus v satisfies the assumptions of Lemma 51, as $C_A \neq C_B$ for $A \neq B$ (recall that X is a solution to $\mathcal{I}$). Therefore, each vertex $v \in X$ sees at most $k+1$ equivalence classes of $\mathcal{R}$. From connectivity of $G[C]$ we infer that each

component C_A must be seen by some element of X, so $C \setminus X$ may contain vertices of at most $k(k+1)$ equivalence classes of $\mathcal{R}$. ∎

STEP 6.2. *If there exist $u, v \in T$ such that u and v lie in different connected components of G, but $(u, v) \in \mathcal{R}$, or there exists a connected component of G with terminals of more than $k^2 + k$ equivalence classes or $\mathcal{R}$, return NO.*

Clearly, Step 6.2 can be applied in $O(n^2)$ time.

We now ensure connectivity of G by considering separately all connected components. Recall that we are developing an algorithm that not only resolves the given N-MWCU instance but also, in case of the positive answer, returns a solution of minimum possible size.

STEP 6.3. *For each connected component of G with vertex set C, pass the instance $(G, T \cap C, \mathcal{R}|_{T \cap C}, k)$ to the next step. If any of the subinstances returns NO, or if the union of the solutions to the subcases is larger than k, return NO. Otherwise, return YES and the union of the solutions for the connected components as the solution to the given instance.*

The correctness of Step 6.3 is straightforward (note that Step 6.2 refutes instances where one equivalence class of $\mathcal{R}$ is scattered among more than one connected component of G) and splitting G into connected components takes linear time in the size of G. Thus, from this point we may assume that G is connected and that the number of equivalence classes or $\mathcal{R}$ is bounded by $\ell := k^2 + k$.

6.2. Operations on the input graph. In this section we show basic operations the algorithm repetitively applies to the graph.

DEFINITION 54. *Let $\mathcal{I} = (G, T, \mathcal{R}, k)$ be an* N-MWCU *instance and let $v \in V(G) \setminus T$. By* bypassing *a vertex v we mean the following operation: we delete the vertex v from the graph and for any $u_1, u_2 \in N_G(v)$, we add an edge u_1u_2 if it is not already present in G.*

We now state the properties of the bypassing operation.

LEMMA 55. *Let $\mathcal{I} = (G, T, \mathcal{R}, k)$ be an* N-MWCU *instance, let $v \in V(G) \setminus T$, and let $\mathcal{I}' = (G', T, \mathcal{R}, k)$ be the instance $\mathcal{I}$ with v bypassed. Then,*

- *if X is a solution to $\mathcal{I}'$, then X is a solution to $\mathcal{I}$ as well;*
- *if X is a solution to $\mathcal{I}$ and $v \notin X$ then X is a solution to $\mathcal{I}'$ as well.*

Proof. The claim follows from the following correspondence of the paths in G and G': any path P' in G' has a corresponding walk P in G, where each occurrence of an edge of $E(G') \setminus E(G)$ is replaced with a length-2 subpath via v. Moreover, any path P in G that does not start or end in v has a corresponding path P' in G', where a possible occurrence of v is circumvented by an edge in G' between two neighbors of v. ∎

Apart from the bypassing operation, we need to show a way to reduce the number of terminals.

DEFINITION 56. *Let $\mathcal{I} = (G, T, \mathcal{R}, k)$ be an* N-MWCU *instance and let $u, v \in T$ be two terminals with $u \neq v$, $(u, v) \in \mathcal{R}$. By* identifying *u and v we mean the following operation: we replace vertices u and v with a new vertex w_{uv} that is adjacent to all vertices of $N_G(u) \cup N_G(v)$. Moreover, we update $\mathcal{R}$ by substituting u and v with w in the equivalence class they belong to.*

LEMMA 57. *Let $\mathcal{I} = (G, T, \mathcal{R}, k)$ be an* N-MWCU *instance and let $u, v \in T$ be two different terminals with $(u, v) \in \mathcal{R}$, such that $uv \in E(G)$ or $|N_G(u) \cap N_G(v)| > k$.*

Let $\mathcal{I}'$ be instance $\mathcal{I}$ with terminals u and v identified. Then the set of solutions to $\mathcal{I}'$ and $\mathcal{I}$ are equal.

Proof. The lemma follows from the fact that for any $X \subseteq V(G) \setminus T$ of size at most k, in $G \setminus X$ the vertices u and v lie in the same connected component. ∎

LEMMA 58. *Let $\mathcal{I} = (G, T, \mathcal{R}, k)$ be an* N-MWCU *instance and let $u_1, u_2, u_3 \in T$ be three different terminals of the same equivalence class of $\mathcal{R}$, pairwise nonadjacent and such that $N_G(u_1) = N_G(u_2) = N_G(u_3) \subseteq V(G) \setminus T$. Let $\mathcal{I}'$ be obtained from $\mathcal{I}$ by deleting the terminal u_3 (and all pairs that contain u_3 in $\mathcal{R}$). Then the set of solutions to $\mathcal{I}'$ and $\mathcal{I}$ are equal.*

Proof. Let $X \subseteq V(G) \setminus T$. We claim that for any $u, v \in V(G) \setminus \{u_3\}$, u and v are in the same connected component of $G \setminus X$ iff they are in the same connected component of $G' \setminus X$. Indeed, the backward implication is trivial, whereas for the forward implication observe that any path from u to v in $G \setminus X$ that visits u_3 can be redirected via u_1 or u_2.

The proven equivalence already shows that any solution to $\mathcal{I}$ is a solution to $\mathcal{I}'$ as well. For the other direction, we need to additionally verify that for any $v \in T$, we have $(u_3, v) \in \mathcal{R}$ iff v and u_3 are in the same connected component of $G \setminus X$, assuming that X is a solution to $\mathcal{I}'$. As X is a solution to $\mathcal{I}'$ and $(u_1, u_2) \in \mathcal{R}$, there exists $w \in N_G(u_1) \setminus X$. Therefore u_1, u_2, and u_3 are in the same connected component of $G \setminus X$. As $(u_1, v) \in \mathcal{R}$ iff $(u_3, v) \in \mathcal{R}$, the claim follows. ∎

6.3. Borders and recursive understanding. For the recursive understanding phase, we need to define the bordered problem. Let $\mathcal{I} = (G, T, \mathcal{R}, k)$ be an N-MWCU instance and let $T_b \subseteq V(G) \setminus T$ be a set of border terminals; we assume $|T_b| \leq 2k$. Define $\mathcal{I}_b = (G, T, \mathcal{R}, k, T_b)$ to be an instance of the bordered problem. By $\mathbb{P}(\mathcal{I}_b)$ we define the set of all triples $\mathcal{P} = (X_b, E_b, \mathcal{R}_b)$, such that $X_b \subseteq T_b$, E_b is an equivalence relation on $T_b \setminus X_b$ and $\mathcal{R}_b$ is an equivalence relation on $T \cup (T_b \setminus X_b)$ such that $E_b \subseteq \mathcal{R}_b$ and $\mathcal{R}_b|_T = \mathcal{R}$. For a triple $\mathcal{P} = (X_b, E_b, \mathcal{R}_b)$, by $G_{\mathcal{P}}$ we denote the graph $G \cup E_b$, that is, the graph G with additional edges E_b.

We say that a set $X \subseteq V(G) \setminus T$ is a solution to $(\mathcal{I}_b, \mathcal{P})$ if $|X| \leq k$, $X \cap T_b = X_b$, and for any $u, v \in T \cup (T_b \setminus X_b)$, the vertices u and v are in the same connected component of the graph $G_{\mathcal{P}} \setminus X$ (i.e., we delete vertices X and add edges E_b) iff $(u, v) \in \mathcal{R}_b$.

We also say that X is a solution to $\mathcal{I}_b = (G, T, \mathcal{R}, k, T_b)$ whenever X is a solution to $\mathcal{I} = (G, T, \mathcal{R}, k)$. Note that if X is a solution to $(\mathcal{I}_b, \mathcal{P})$, the set X is not necessarily a solution to $\mathcal{I}_b$; however, X is a solution to the N-MWCU instance $(G_{\mathcal{P}}, T, \mathcal{R}, k)$.

One may think of the set of edges E_b as the "prediction" of which vertices will be connected *outside* the currently considered part of the graph, after the solution edges have been deleted. Since such a definition may not be very intuitive, we provide detailed proofs of all equivalences in this section; note that corresponding equivalence claims in the previous sections were nearly straightforward.

We formally define the bordered problem as follows.

BORDER N-MWCU
Input: An N-MWCU instance $\mathcal{I} = (G, T, \mathcal{R}, k)$ with G being connected and a set $T_b \subseteq V(G) \setminus T$ of size at most $2k$; denote $\mathcal{I}_b = (G, T, \mathcal{R}, k, T_b)$
Output: For each $\mathcal{P} = (X_b, E_b, \mathcal{R}_b) \in \mathbb{P}(\mathcal{I}_b)$, output a $\mathtt{sol}_{\mathcal{P}} = X_{\mathcal{P}}$ being a solution to $(\mathcal{I}_b, \mathcal{P})$ with minimum possible $|X_{\mathcal{P}}|$, or $\mathtt{sol}_{\mathcal{P}} = \perp$ if such a solution does not exist.

Clearly, N-MWCU reduces to BORDER N-MWCU, as we may ask for an instance

with $T_b = \emptyset$. Moreover, in this case the single answer to Border N-MWCU for $\mathcal{P} = (\emptyset, \emptyset, \mathcal{R})$ returns a solution of minimum possible size.

We note that

$$|\mathbb{P}(\mathcal{I}_b)| \leq (1 + |T_b|(|T_b| + \ell))^{|T_b|} \leq (2k^3 + 6k^2 + 1)^{2k} = 2^{O(k \log k)},$$

as $\mathcal{R}_b$ has at most $\ell + |T_b|$ equivalence classes, E_b has at most $|T_b|$ equivalence classes, and each $v \in T_b$ can either go to X_b or choose an equivalence class in $\mathcal{R}_b$ and E_b. Let $q = k(2k^3 + 6k^2 + 1)^{2k} + k$; all output solutions to a Border N-MWCU instance $\mathcal{I}_b$ contain at most $q - k$ vertices in total.

Armed with the previous definitions, we are now ready to prove a somewhat expected at this point lemma showing that if a Border N-MWCU instance $\mathcal{I}_b$ contains another, smaller subinstance $\widehat{\mathcal{I}}_b$, then it suffices to restrict our attention to only one, fixed output to Border N-MWCU on $\widehat{\mathcal{I}}_b$. In other words, we show that there exists a valid output to Border N-MWCU on $\mathcal{I}_b$ that, on $\widehat{\mathcal{I}}_b$, behaves as prescribed by the aforementioned fixed output. Due to an involved definition of E_b as a connectivity prediction, the formal proof is quite involved.

Lemma 59. *Assume we are given a* Border N-MWCU *instance* $\mathcal{I}_b = (G, T, \mathcal{R}, k, T_b)$*, a set* $Z \subseteq V(G) \setminus T$ *with* $|Z| \leq k$*, and a connected component* $\widehat{V}$ *of* $G - Z$*. Denote* $Z_W := N_G(\widehat{V}) \subseteq Z$ *and* $W := \widehat{V} \cup Z_W$*, and assume furthermore that* $G[W]$ *is connected and that* $|\widehat{V} \cap T_b| \leq k$*.*

Denote $\widehat{G} = G[W]$*,* $\widehat{T}_b = (T_b \cup Z_W) \cap W$*,* $\widehat{T} = T \cap W$*,* $\widehat{\mathcal{R}} = \mathcal{R}|_{T \cap W}$*, and* $\widehat{\mathcal{I}}_b = (\widehat{G}, \widehat{T}, \widehat{\mathcal{R}}, k, \widehat{T}_b)$*. Then* $\widehat{\mathcal{I}}_b$ *is a proper* Border N-MWCU *instance. Moreover, if we denote by* $(\widehat{\mathtt{sol}}_{\widehat{\mathcal{P}}})_{\widehat{\mathcal{P}} \in \mathbb{P}(\widehat{\mathcal{I}}_b)}$ *an arbitrary output to the* Border N-MWCU *instance* $\widehat{\mathcal{I}}_b$ *and*

$$U(\widehat{\mathcal{I}}_b) = \widehat{T}_b \cup \bigcup \{\widehat{X}_{\widehat{\mathcal{P}}} : \widehat{\mathcal{P}} \in \mathbb{P}(\widehat{\mathcal{I}}_b), \widehat{\mathtt{sol}}_{\widehat{\mathcal{P}}} = \widehat{X}_{\widehat{\mathcal{P}}} \neq \bot\}$$

to be a set of vertices used by the solutions of $(\widehat{\mathtt{sol}}_{\widehat{\mathcal{P}}})_{\widehat{\mathcal{P}} \in \mathbb{P}(\widehat{\mathcal{I}}_b)}$*, then there exists a correct output* $(\mathtt{sol}_{\mathcal{P}})_{\mathcal{P} \in \mathbb{P}(\mathcal{I}_b)}$ *to the* Border N-MWCU *instance* $\mathcal{I}_b$ *such that whenever* $\mathtt{sol}_{\mathcal{P}} = X_{\mathcal{P}} \neq \bot$*, then* $X_{\mathcal{P}} \cap \widehat{V} \subseteq U(\widehat{\mathcal{I}}_b)$*.*

Proof. The claim that $\widehat{\mathcal{I}}_b$ is a proper Border N-MWCU instance follows directly from the assumptions that $\widehat{G} = G[W]$ is connected, $|Z_W| \leq |Z| \leq k$, and $|\widehat{V} \cap T_b| \leq k$. In the rest of the proof we justify the second claim of the lemma.

Fix $\mathcal{P} = (X_b, E_b, \mathcal{R}_b) \in \mathbb{P}(\mathcal{I}_b)$ and recall that $G_{\mathcal{P}} = G \cup E_b$. Assume that there exists a solution to the instance $(\mathcal{I}_b, \mathcal{P})$; let $X_{\mathcal{P}}$ be such a solution with minimum possible $|X_{\mathcal{P}}|$. To prove the lemma we need to show a second solution $X'_{\mathcal{P}}$ to $(\mathcal{I}_b, \mathcal{P})$, $|X'_{\mathcal{P}}| \leq |X_{\mathcal{P}}|$, and $X'_{\mathcal{P}} \cap \widehat{V} \subseteq U(\widehat{\mathcal{I}}_b)$.

Let us first give an intuition of the proof. We are given a solution $X_{\mathcal{P}}$; our goal is to modify it on $\widehat{V}$ so that it behaves on $\widehat{\mathcal{I}}_b$ as predicted by the output $(\widehat{\mathtt{sol}}_{\widehat{\mathcal{P}}})_{\widehat{\mathcal{P}} \in \mathbb{P}(\widehat{\mathcal{I}}_b)}$. To this end, we observe how $X_{\mathcal{P}} \cap W$ behaves on $\widehat{\mathcal{I}}_b$ and define a triple $\widehat{\mathcal{P}} \in \mathbb{P}(\widehat{\mathcal{I}}_b)$ so that $X_{\mathcal{P}} \cap W$ is a solution to $(\widehat{\mathcal{I}}_b, \widehat{\mathcal{P}})$. The information stored in the triple $\widehat{\mathcal{P}}$ should be enough to argue that swapping $X_{\mathcal{P}} \cap W$ with $\widehat{\mathtt{sol}}_{\widehat{\mathcal{P}}}$ does not invalidate $X_{\mathcal{P}}$ as a solution to $(\mathcal{I}_b, \mathcal{P})$.

Let us now proceed with this strategy in full detail. We start by defining a triple $\widehat{\mathcal{P}} = (\widehat{X}_b, \widehat{E}_b, \widehat{\mathcal{R}}_b)$ that represents how $X_{\mathcal{P}} \cap W$ behaves on $\widehat{\mathcal{I}}_b$.

- As $\widehat{X}_b$ is meant to keep information on deleted border terminals, its definition is straightforward. We take $\widehat{X}_b = X_{\mathcal{P}} \cap \widehat{T}_b$; note that $X_b \cap W \subseteq \widehat{X}_b$ since $X_{\mathcal{P}}$

is a solution to $(\mathcal{I}_b, \mathcal{P})$.

- Recall that $\widehat{E}_b$ is meant to predict connectivity between border terminals outside the instance $\widehat{\mathcal{I}}_b$. Consequently, in the definition of $\widehat{E}_b$ we need to take into account both the graph $G - \widehat{V}$ after the deletion of $X_{\mathcal{P}}$ as well as the edges E_b. Formally, we define $\widehat{E}_b$ to be the following relation on $\widehat{T}_b \setminus \widehat{X}_b$: $(u, v) \in \widehat{E}_b$ iff u and v are in the same connected component of $G_{\mathcal{P}} \setminus ((\widehat{V} \setminus T_b) \cup X_{\mathcal{P}})$ (in particular, if $(u, v) \in E_b$).
- Recall that $\widehat{\mathcal{R}}_b$ is meant as a requirement on the final connectivity of the terminals and border terminals in the entire graph; to this end, we can use connectivity in $G_{\mathcal{P}} \setminus X_{\mathcal{P}}$. That is, we define $\widehat{\mathcal{R}}_b$ to be the following relation on $\widehat{T} \cup (\widehat{T}_b \setminus \widehat{X}_b)$: $(u, v) \in \mathcal{R}_b$ iff u and v are in the same connected component of $G_{\mathcal{P}} \setminus X_{\mathcal{P}}$. As $X_{\mathcal{P}}$ is a solution to $(\mathcal{I}_b, \mathcal{P})$, $\widehat{\mathcal{R}}_b|_{\widehat{T}} = \widehat{\mathcal{R}} = \mathcal{R}|_{\widehat{T}}$.

Note that $\widehat{E}_b \subseteq \widehat{\mathcal{R}}_b$, as both $\widehat{E}_b$ and $\widehat{\mathcal{R}}_b$ correspond to the relation of being in the same connected component, but in $\widehat{E}_b$ we consider a smaller graph than in $\widehat{\mathcal{R}}_b$. This justifies that $\widehat{\mathcal{P}} \in \mathbb{P}(\mathcal{I}_b)$.

The main idea of the definition of $\widehat{\mathcal{P}}$ is that $X_{\mathcal{P}} \cap W$ is a solution to $(\widehat{\mathcal{I}}_b, \widehat{\mathcal{P}})$. Let us now verify it formally. Clearly, $|X_{\mathcal{P}} \cap W| \leq k$. By the definition of $\widehat{X}_b$, we have $X_{\mathcal{P}} \cap W \cap \widehat{T}_b = \widehat{X}_b$. Consider two vertices $u, v \in \widehat{T} \cup (\widehat{T}_b \setminus \widehat{X}_b)$. We have $(u, v) \in \widehat{\mathcal{R}}_b$ iff there exists a path P between u and v in $G_{\mathcal{P}} \setminus X_{\mathcal{P}}$. By the definition of $\widehat{E}_b$, such a path P exists iff there exists a path $\widehat{P}$ connecting u and v in $\widehat{G}_{\widehat{\mathcal{P}}} \setminus X_{\mathcal{P}}$: each subpath of P with internal vertices in $V(G) \setminus W$ corresponds to an edge in $\widehat{E}_b$ and vice versa. Thus, u and v are in the same connected component of $\widehat{G}_{\widehat{\mathcal{P}}} \setminus X_{\mathcal{P}}$ iff $(u, v) \in \widehat{\mathcal{R}}_b$ and the claim is proven.

To wrap up, we have defined an element $\widehat{\mathcal{P}} \in \mathbb{P}(\widehat{\mathcal{I}}_b)$ that represents the behavior of $X_{\mathcal{P}}$ on $\widehat{\mathcal{I}}_b$. Our goal now is to show that if we swap $X_{\mathcal{P}} \cap W$ with $\widehat{\texttt{sol}}_{\widehat{\mathcal{P}}}$, that is, the prescribed solution to $(\widehat{\mathcal{I}}_b, \widehat{\mathcal{P}})$, we obtain another solution to $(\mathcal{I}_b, \mathcal{P})$ that fulfills our requirements.

Since $X_{\mathcal{P}} \cap W$ is a solution to $(\widehat{\mathcal{I}}_b, \widehat{\mathcal{P}})$, we infer that $\widehat{\texttt{sol}}_{\widehat{\mathcal{P}}} = \widehat{X}_{\widehat{\mathcal{P}}} \neq \bot$ and $|\widehat{X}_{\widehat{\mathcal{P}}}| \leq X_{\mathcal{P}} \cap W$. Let us define our new solution to $(\mathcal{I}_b, \mathcal{P})$ as $X'_{\mathcal{P}} = (X_{\mathcal{P}} \setminus W) \cup \widehat{X}_{\widehat{\mathcal{P}}}$. Clearly, $|X'_{\mathcal{P}}| \leq |X_{\mathcal{P}}|$. To finish the proof of the lemma, we need to formally show that indeed $X'_{\mathcal{P}}$ is a solution to $(\mathcal{I}_b, \mathcal{P})$.

It is straightforward to verify that $X'_{\mathcal{P}}$ satisfies the constaint imposed by X_b: As $\widehat{X}_b$ is defined as $X_{\mathcal{P}} \cap \widehat{T}_b$ and $X_{\mathcal{P}}$ is a solution to $(\mathcal{I}_b, \mathcal{P})$, we have $X'_{\mathcal{P}} \cap T_b = X_b$.

It remains to check the connectivity requirement imposed by $\mathcal{R}_b$. Let $u, v \in T \cup (T_b \setminus X_b)$. Our goal is to show that u and v lie in the same connected component of $G_{\mathcal{P}} \setminus X'_{\mathcal{P}}$ iff they lie in the same connected component of $G_{\mathcal{P}} \setminus X_{\mathcal{P}}$. We present the proof only in one direction, as the proofs in both directions are totally symmetric: we use only the facts that both $X_{\mathcal{P}} \cap W$ and $X'_{\mathcal{P}} \cap W$ are solutions to $(\widehat{\mathcal{I}}_b, \widehat{\mathcal{P}})$ and that $X_{\mathcal{P}} \setminus \widehat{V} = X'_{\mathcal{P}} \setminus \widehat{V}$. The last equality holds because $Z_W \subseteq \widehat{T}_b$, so $Z_W \cap X_{\mathcal{P}} = Z_W \cap \widehat{X}_{\widehat{\mathcal{P}}}$.

Thus, assume that $u, v \in T \cup (T_b \setminus X_b)$ and u and v lie in the same connected component of $G_{\mathcal{P}} \setminus X_{\mathcal{P}}$. Let P be a path that connects u and v in $G_{\mathcal{P}} \setminus X_{\mathcal{P}}$. The remainder of the proof works as follows: we partition P into parts that live entirely in $G_{\mathcal{P}} \setminus \widehat{V}$ and in $G_{\mathcal{P}}[W]$ and argue that each such part of P has its counterpart in $G_{\mathcal{P}} \setminus X'_{\mathcal{P}}$.

Formally, let $u = v_0, v_1, v_2, \ldots, v_r = v$ be a sequence of vertices that lie on the path P and belong to $D := T \cup (T_b \setminus X_b) \cup Z_W$, in the order they appear on P. First note that since $X_{\mathcal{P}} \setminus \widehat{V} = X'_{\mathcal{P}} \setminus \widehat{V}$ and both $X_{\mathcal{P}} \cap W$ and $X'_{\mathcal{P}} \cap W$ are solutions to

$(\widehat{\mathcal{I}}_b, \widehat{\mathcal{P}})$, we have that $X_{\mathcal{P}} \cap D = X'_{\mathcal{P}} \cap D = X_b \cup \widehat{X}_b$. Thus, for each $0 \leq i \leq r$, we have $v_i \notin X'_{\mathcal{P}}$. To finish the proof of the lemma we need to show that for any $0 \leq i < r$, the vertices v_i and v_{i+1} lie in the same connected component of $G_{\mathcal{P}} \setminus X'_{\mathcal{P}}$.

Let P_i be the subpath of P between v_i and v_{i+1}. As $Z_W \subseteq D$, P_i is either a path in $G_{\mathcal{P}} \setminus \widehat{V}$ or a path in $G_{\mathcal{P}}[W]$. In the first case, since $X_{\mathcal{P}} \setminus \widehat{V} = X'_{\mathcal{P}} \setminus \widehat{V}$, we infer that the path P_i is present in $G_{\mathcal{P}} \setminus X'_{\mathcal{P}}$ and the claim is proven. In the second case, note that we have $(v_i, v_{i+1}) \in \widehat{\mathcal{R}}_b$. As $X'_{\mathcal{P}} \cap W$ is a solution to $(\widehat{\mathcal{I}}_b, \widehat{\mathcal{P}})$, we infer that v_i and v_{i+1} are connected via a path $\widehat{P}_i$ in $\widehat{G}_{\widehat{\mathcal{P}}} \setminus (X'_{\mathcal{P}} \cap W)$. However, by the definition of $\widehat{E}_b$, for any edge $w_1w_2 \in \widehat{E}_b$ on $\widehat{P}_i$, the vertices w_1 and w_2 are in the same connected component of $G_{\mathcal{P}} \setminus ((\widehat{V} \setminus T_b) \cup X_{\mathcal{P}})$. Since $X_{\mathcal{P}} \setminus \widehat{V} = X'_{\mathcal{P}} \setminus \widehat{V}$ and $X_{\mathcal{P}} \cap T_b = X'_{\mathcal{P}} \cap T_b$, we have that $X_{\mathcal{P}} \setminus (\widehat{V} \setminus T_b) = X'_{\mathcal{P}} \setminus (\widehat{V} \setminus T_b)$ and the claim is proven. This finishes the proof of the lemma. ∎

Note that in Lemma 59 we have $|U(\widehat{\mathcal{I}}_b) \cap \widehat{V}| \leq q$.

A recursive call due to an application of Lemma 59 allows us to reduce the number of nonterminal vertices in $\widehat{V}$ to at most $q = 2^{O(k \log k)}$. To make the recursion work in FPT time, we need to reduce the number of terminals as well. Fortunately, this is quite easy, due to the identifying operation and Lemma 57.

We are now ready to present the recursive step of the algorithm.

STEP 6.4. *Assume we are given a* BORDER N-MWCU *instance* $\mathcal{I}_b = (G, T, \mathcal{R}, k, T_b)$. *Invoke first the algorithm of Lemma* 19 *in a search for* (q,k)*-good node separation (with* $V^\infty = T$*). If it returns a good node separation* (Z, V_1, V_2)*, let* $j \in \{1,2\}$ *be such that* $|V_j \cap T_b| \leq k$ *and denote* $\widehat{Z} = Z$, $\widehat{V} = V_j$. *Otherwise, if it returns that no such good node separation exists in* G*, invoke the algorithm of Lemma* 20 *in a search for* (q,k)*-flower separation w.r.t.* T_b *(with* $V^\infty = T$ *again). If it returns that no such flower separation exists in* G*, pass the instance* $\mathcal{I}_b$ *to the next step. Otherwise, if it returns a flower separation* $(Z, (V_i)_{i=1}^{\ell})$*, denote* $\widehat{Z} = Z$ *and* $\widehat{V} = \bigcup_{i=1}^{\ell} V_i$.

In the case we have obtained $\widehat{Z}$ *and* $\widehat{V}$ *(from either Lemma* 19 *or Lemma* 20*), invoke the algorithm recursively for the* BORDER N-MWCU *instance* $\widehat{\mathcal{I}}_b$ *defined as in the statement of Lemma* 59 *for separator* $\widehat{Z}$ *and set* $\widehat{V}$*, obtaining an output* $(\mathtt{sol}_{\widehat{\mathcal{P}}})_{\mathcal{P} \in \mathbb{P}(\widehat{\mathcal{I}}_b)}$. *Compute the set* $U(\widehat{\mathcal{I}}_b)$. *Bypass (in an arbitrary order) all vertices of* $\widehat{V} \setminus (T \cup U(\widehat{\mathcal{I}}_b))$. *Recall that* $\widehat{T}_b \subseteq U(\widehat{\mathcal{I}}_b)$*, so no border terminal gets bypassed.*

After all vertices of $\widehat{V} \setminus U(\widehat{\mathcal{I}}_b)$ *are bypassed, perform the following operations on terminals of* $\widehat{V} \cap T$*:*

1. *As long as there exist two different* $u, v \in \widehat{V} \cap T$ *that satisfy* $uv \in E(G)$ *or* $|N_G(u) \cap N_G(v)| > k$ *do as follows: if* $(u,v) \in \mathcal{R}$*, identify* u *and* v*, and otherwise output* $\perp$ *for all* $\mathcal{P} \in \mathbb{P}(\mathcal{I}_b)$.
2. *If the above is not applicable, then, as long as there exist three pairwise distinct terminals* $u_1, u_2, u_3 \in T$ *of the same equivalence class of* $\mathcal{R}$ *that have the same neighborhood, delete* u_3 *from the graph (and delete all pairs containing* u_3 *from* $\mathcal{R}$*).*

Let $\mathcal{I}'_b$ *be the outcome instance.*

Finally, restart this step on the new instance $\mathcal{I}'_b$ *and obtain a family of solutions* $(\mathtt{sol}'_{\mathcal{P}})_{\mathcal{P} \in \mathbb{P}(\mathcal{I}_b)}$ *and return this family as an output to the instance* $\mathcal{I}_b$.

Let us first verify that the application of Lemma 59 is justified. Indeed, by the definitions of the good node separation and the flower separation, as well as the choice of $\widehat{V}$, we have in both cases $|\widehat{V} \cap T_b| \leq k$ and that $G[\widehat{V} \cup N_G(\widehat{V})]$ is connected. Moreover, note that the recursive call is applied to a graph with a strictly smaller

number of vertices than G: in the case of a good node separation, V_2 is removed from the graph, and in the case of a flower separation, recall that the definition of the flower separation requires $Z \cup \bigcup_{i=1}^{\ell} V_i$ to be a proper subset of $V(G)$.

We have that after the bypassing operations, $\widehat{V}$ contains at most q vertices that are not terminals (at most k border terminals and at most $q-k$ vertices which are neither terminals nor border terminals). Let us now bound the number of terminal vertices once Step 6.4 is applied. Note that after Step 6.4 is applied, for any $v \in T \cap \widehat{V}$, we have $N_G(v) \subseteq (\widehat{V} \setminus T) \cup Z$ and $|(\widehat{V} \setminus T) \cup Z| \leq (q+k)$. Due to the first rule in Step 6.4, for any set $A \subseteq (\widehat{V} \setminus T) \cup Z$ of size $k+1$, at most one terminal of $T \cap \widehat{V}$ is adjacent to all vertices of A. Due to the second rule in Step 6.4, for any set $B \subseteq (\widehat{V} \setminus T) \cup Z$ of size at most k and for each equivalence class of $\mathcal{R}$, there are at most two terminals of this equivalence class with neighborhood exactly B. We infer that

$$|T \cap \widehat{V}| \leq (q+k)^{k+1} + 2\ell \sum_{i=1}^{k} (q+k)^i =: q'.$$

Note that $q' = 2^{O(k^2 \log k)}$.

The following lemma verifies the correctness of Step 6.4.

LEMMA 60. *Assume we are given a* BORDER N-MWCU *instance* $\mathcal{I}_b = (G, T, \mathcal{R}, k, T_b)$ *on which Step* 6.4 *is applied, and let* $\mathcal{I}'_b$ *be an instance after Step* 6.4 *is applied. Then any correct output to the instance* $\mathcal{I}'_b$ *is a correct output to the instance* $\mathcal{I}_b$ *as well. Moreover, if Step* 6.4 *outputs* $\perp$ *for all* $\mathcal{P} \in \mathbb{P}(\mathcal{I}_b)$*, then this is a correct output to* $\mathcal{I}_b$.

Proof. The lemma is a straightforward corollary of Lemma 59, the properties of the bypassing operation described in Lemma 55, and Lemmas 57 and 58. Lemma 59 ensures us that each vertex not in $U(\widehat{\mathcal{I}}_b)$ is omitted by some optimal solution for every $\mathcal{P} \in \mathbb{P}(\mathcal{I}_b)$, which enables us to use Lemma 55. Finally, if for any terminals $u, v \in T$, we have $uv \in E(G)$ or $|N_G(u) \cap N_G(v)| > k$, then u and v are in the same connected component of $G \setminus X$ for any set X of at most k nonterminals and if $(u, v) \notin \mathcal{R}$, for any $\mathcal{P} \in \mathbb{P}(\mathcal{I}_b)$, there is no solution to $(\mathcal{I}_b, \mathcal{P})$. ∎

We are left with the analysis of the time complexity of Step 6.4. The applications of Lemmas 19 and 20 use $O(2^{O(\min(q,k)\log(q+k))} n^3 \log n) = O(2^{O(k^2 \log k)} n^3 \log n)$ time. Let $n' = |\widehat{V}|$; the recursive step is applied to a graph with at most $n' + k$ vertices and, after bypassing, there are at most $\min(n-1, n-n'+q+q')$ vertices left. Moreover, each bypassing operation takes $O(n^2)$ time, and the computation of $U(\widehat{\mathcal{I}}_b)$ takes $O(2^{O(k \log k)} n)$ time. Application of Lemma 57 takes $O(kn^2)$ time per operation, which can be implemented by having a counter for each pair of terminals and increasing those counters accordingly by considering every pair of terminals of $N_G(x)$, for each $x \in V$. Since when a counter reaches value $k+1$ for vertices u, v, we know that $|N_G(u) \cap N_G(v)| > k$, the total time consumed is bounded by $O(kn^2)$. Application of Lemma 58 takes $O(n^2 \log n)$ time per one operation, since we can sort terminals from one equivalence class according to their sets of neighbors. Thus all applications of Lemmas 57 and 58 take $O(n^3(k + \log n))$ time in total. The value of k does not change in this step. Therefore, we have the following recursive formula for time complexity as a function of the number of vertices of G:

(3)
$$T(n) \leq \max_{q+1 \leq n' \leq n-q-1} \Big(O(2^{O(k^2 \log k)} n^3 \log n) + T(n'+k) + T(\min(n-1, n-n'+q+q')) \Big).$$

Note that the function $p(t) = t^4 \log t$ is convex, so it is easy to see that the maximum

is attained either when $n' = q + q' - 1$ or when $n' = n - q - 1$. A straightforward inductive check of both of the ends proves that we have indeed the claimed bound on the complexity, i.e., $T(n) = O(2^{O(k^2 \log k)} n^4 \log n)$.

We conclude this section with a note that Lemma 21 asserts that if Step 6.4 is not applicable, then for any set $Z \subseteq V(G) \setminus T$ of size at most k, the graph $G \setminus Z$ contains at most $t := (2q+1)(2^k - 1) + 2k + 1$ connected components containing a nonterminal, out of which at most one has more than q vertices not from T.

6.4. Brute-force approach. If the graph output by Step 6.4 has a small number of vertices outside T, the algorithm may apply a straightforward brute-force approach to the Border N-MWCU problem. In this section we describe this method formally.

Lemma 61. *A correct output to a* Border N-MWCU *instance* $\mathcal{I}_b = (G, T, \mathcal{R}, k, T_b)$ *can be computed in* $O(2^{O(k \log k)} n^2 n_{\neg T}^k)$ *time, where* $n_{\neg T} = |V(G) \setminus T|$.

Proof. Simply, for each $\mathcal{P} \in \mathbb{P}(\mathcal{I}_b)$ (at most $2^{O(k \log k)}$ choices) for each deletion set $X \subseteq V(G) \setminus T$ with $|X| \leq k$ (at most $(k+1) n_{\neg T}^k$ choices) we verify in $O(n^2)$ time if X is a solution to $(\mathcal{I}_b, \mathcal{P})$. ∎

Step 6.5. *If* $|V(G) \setminus T| \leq qt + k$, *apply Lemma* 61 *to find a correct output to a* Border N-MWCU *instance* $\mathcal{I}_b = (G, T, \mathcal{R}, k, T_b)$.

Recall that $q, t \leq 2^{O(k \log k)}$. Thus, if Step 6.5 is applicable, its running time is $O(2^{O(k^2 \log k)} n^2)$.

6.5. High connectivity phase. Assume we have a Border N-MWCU instance $\mathcal{I}_b = (G, T, \mathcal{R}, k, T_b)$ where Steps 6.4 and 6.5 are not applicable. In this section we show that high connectivity of G makes the problem much easier. To this end, fix $\mathcal{P} = (X_b, E_b, \mathcal{R}_b) \in \mathbb{P}(\mathcal{I}_b)$. We focus on finding the solution $\mathtt{sol}_{\mathcal{P}}$; iterating through all the possible $\mathcal{P}$ gives additional $2^{O(k \log k)}$ overhead to the running time. Recall that $G_{\mathcal{P}} = G \cup E_b$.

Note that if $|V(G) \setminus T|$ is too large for Step 6.5 to be applicable, for any set $Z \subseteq V(G) \setminus T$ of size at most k, the bound on the number of connected components from Lemma 21 implies that there exists exactly one connected component of $G \setminus Z$ with more than q vertices outside T; denote its vertex set by $\mathtt{big}(Z)$.

We now use Lemma 1 to get some more structure of the graph G.

Definition 62. *Let* $Z \subseteq V(G) \setminus T$ *be a set of size at most* k *and let* $S \subseteq V(G) \setminus T$. *We say that* S interrogates Z *if the following holds:*

1. $S \cap Z = \emptyset$;
2. *for any connected component* C *of* $G \setminus Z$ *with at most* q *vertices outside* T, *all vertices of* C *belong to* $S \cup T$.

Lemma 63. *Let* $\mathcal{F}$ *be a family obtained by the algorithm of Lemma* 1 *for universe* $U = V(G) \setminus T$ *and constants* $a = qt$ *and* $b = k$. *Then, for any* $Z \subseteq V(G) \setminus T$ *with* $|Z| \leq k$, *there exists a set* $S \in \mathcal{F}$ *that interrogates* Z.

Proof. Fix $Z \subseteq V(G) \setminus T$ with $|Z| \leq k$. Let A be the union of vertex sets of all connected components of $G \setminus Z$ that have at most q vertices outside T; by Lemma 21, $|A \setminus T| \leq qt$. By Lemma 1, as $|A \setminus T| \leq qt$ and $|Z| \leq k$, there exists a set $S \in \mathcal{F}$ that contains $A \setminus T$ and is disjoint with Z. By the construction of the set A, S interrogates Z and the lemma is proven. ∎

Note that, as $q, t = 2^{O(k \log k)}$, the family $\mathcal{F}$ of Lemma 63 is of size $2^{O(k^2 \log k)} \log n$ and can be computed in $O(2^{O(k^2 \log k)} n \log n)$ time. Therefore we may branch, guessing a set S that interrogates a solution $\mathtt{sol}_{\mathcal{P}} = X_{\mathcal{P}}$ we are looking for. Formally, we

perform computations in each branch and return the minimum size solution from those obtained in the branches.

STEP 6.6. *Compute the family $\mathcal{F}$ from Lemma 63 and branch into $|\mathcal{F}|$ subcases, indexed by sets $S \in \mathcal{F}$. In a branch S we seek for a set $X_{\mathcal{P}}$ with minimum possible $|X_{\mathcal{P}}|$ that not only is a solution to $(\mathcal{I}_b, \mathcal{P})$ but also is interrogated by S.*

Lemma 63 verifies the correctness of the branching of Step 6.6; as discussed, the step is applied in $O(2^{O(k^2 \log k)} n \log n)$ time and leads to $O(2^{O(k^2 \log k)} \log n)$ subcases.

The following observation is crucial for the final step.

LEMMA 64. *Let $X_{\mathcal{P}}$ be a minimum size set that is a solution to $(\mathcal{I}_b, \mathcal{P})$ interrogated by S. Then there exists a set $T^{\mathtt{big}} \subseteq T \cup (T_b \setminus X_b)$ that is empty or contains all vertices of exactly one equivalence class of $\mathcal{R}_b$, such that $X_{\mathcal{P}} = X_b \cup N_G(S(T^{\mathtt{big}}))$, where $S(T^{\mathtt{big}})$ is the union of vertex sets of all connected components of $G[S \cup T \cup (T_b \setminus X_b)]$ that contain a vertex of $(T \cup (T_b \setminus X_b)) \setminus T^{\mathtt{big}}$.*

Proof. Consider the graph $G_{\mathcal{P}} \setminus X_{\mathcal{P}}$ and let $\mathtt{big}_{\mathcal{P}}(X_{\mathcal{P}})$ be the vertex set of the connected component of $G_{\mathcal{P}} \setminus X_{\mathcal{P}}$ that contain $\mathtt{big}(X_{\mathcal{P}})$ (recall that $G_{\mathcal{P}}$ is the graph G with additional edges E_p; thus $\mathtt{big}_{\mathcal{P}}(X_{\mathcal{P}})$ may be significantly larger than $\mathtt{big}(X_{\mathcal{P}})$). As $X_{\mathcal{P}}$ is a solution to $(\mathcal{I}_b, \mathcal{P})$, we have $X_{\mathcal{P}} \cap T_b = X_b$. Define $T^{\mathtt{big}} = (T \cup (T_b \setminus X_b)) \cap \mathtt{big}_{\mathcal{P}}(X_{\mathcal{P}})$; as $X_{\mathcal{P}}$ is a solution to $(\mathcal{I}_b, \mathcal{P})$, $T^{\mathtt{big}}$ is empty or contains vertices of exactly one equivalence class or $\mathcal{R}_b$.

Now let C be the vertex set of a connected component of $G \setminus X_{\mathcal{P}}$ that contains a vertex $v \in (T \cup (T_b \setminus X_b)) \setminus T^{\mathtt{big}}$. Clearly, $v \notin \mathtt{big}_{\mathcal{P}}(X_{\mathcal{P}})$. As S interrogates $X_{\mathcal{P}}$, $\mathtt{big}_{\mathcal{P}}(X_{\mathcal{P}})$ contains $\mathtt{big}(X_{\mathcal{P}})$, and $X_{\mathcal{P}} \cap (T \cup T_b) = X_b \subseteq T_b$, we infer that C is the vertex set of a connected component of $G[S \cup T \cup (T_b \setminus X_b)]$ as well. As $v \in C$, C is a connected component of $G[S(T^{\mathtt{big}})]$. Since the choice of C was arbitrary, we infer that $N_G(S(T^{\mathtt{big}})) \subseteq X_{\mathcal{P}}$. Denote $X'_{\mathcal{P}} = X_b \cup N_G(S(T^{\mathtt{big}})) \subseteq X_{\mathcal{P}}$. To finish the proof of the lemma we need to show that $X'_{\mathcal{P}}$ is a solution to $(\mathcal{I}_b, \mathcal{P})$ as well.

Clearly, $X'_{\mathcal{P}} \cap (T \cup T_b) = X_b$, as $N_G(S(T^{\mathtt{big}})) \cap (T \cup (T_b \setminus X_b)) = \emptyset$ by the definition of $S(T^{\mathtt{big}})$. Moreover, as $X'_{\mathcal{P}} \subseteq X_{\mathcal{P}}$ and $X_{\mathcal{P}}$ is a solution to $(\mathcal{I}_b, \mathcal{P})$, if $(u, v) \in \mathcal{R}_b$, then u and v are in the same connected component of $G_{\mathcal{P}} \setminus X'_{\mathcal{P}}$. We now show that for any $(u, v) \notin \mathcal{R}_b$ the vertices u and v are in different connected components of $G_{\mathcal{P}} \setminus X'_{\mathcal{P}}$. Assume the contrary, and let $u, v \in T \cup (T_b \setminus X_b)$ be such that $(u, v) \notin \mathcal{R}_b$, u and v are in the same connected component of $G_{\mathcal{P}} \setminus X'_{\mathcal{P}}$, and the distance between u and v in $G_{\mathcal{P}} \setminus X'_{\mathcal{P}}$ is the minimum possible. Let P be a shortest path between u and v in $G_{\mathcal{P}} \setminus X'_{\mathcal{P}}$.

As $X_{\mathcal{P}}$ is a solution to $(\mathcal{I}_b, X_b)$, u and v are in different connected components of $G_{\mathcal{P}} \setminus X_{\mathcal{P}}$; without loss of generality assume $v \notin \mathtt{big}_{\mathcal{P}}(X_{\mathcal{P}})$ and let C be the vertex set of the connected component of $G \setminus X_{\mathcal{P}}$ that contains v. Clearly, since $(u, v) \notin \mathcal{R}_b$, we have $u \notin C$. Moreover, $v \in (T \cup (T_b \setminus X_b)) \setminus T^{\mathtt{big}}$ and C is a connected component of $G[S(T^{\mathtt{big}})]$. Therefore $N_G(C) \subseteq X'_{\mathcal{P}}$. Since $u \notin C$, the path P needs to go via an edge $v_1 u_1 \in E_b$, where $v_1 \in C$ but $u_1 \notin C$. Note that then $u_1, v_1 \in T_b$. As $v_1 \in C$ and $X_{\mathcal{P}}$ is a solution to $(\mathcal{I}_b, \mathcal{P})$, we have $(v, v_1) \in \mathcal{R}_b$. As $E_b \subseteq \mathcal{R}_b$, we have that $(v, u_1) \in \mathcal{R}_b$. As $(u, v) \notin \mathcal{R}_b$, we infer that $(u_1, u) \notin \mathcal{R}_b$, but u_1 and u are connected via a proper subpath of P in $G_{\mathcal{P}} \setminus X'_{\mathcal{P}}$, a contradiction to the choice of u, v, and P. This finishes the proof of the lemma. ∎

Lemma 64 justifies the final step.

STEP 6.7. *In each branch, where by S we denote the corresponding guess, for each set $T^{\mathtt{big}}$ that is empty or contains all vertices of one equivalence class of $\mathcal{R}_b$, check if $X_b \cup N_G(S(T^{\mathtt{big}}))$ is a solution to $(\mathcal{I}_b, \mathcal{P})$ that is interrogated by S. For given $\mathcal{P}$, output*

the smallest solution to $(\mathcal{I}_b, \mathcal{P})$ *found, or* $\perp$ *if no solution is found for any choice of* S *and* $T^{\mathtt{big}}$.

Note that $\mathcal{R}$ has at most $\ell = k^2 + k$ equivalence classes. As $|T_b| \leq 2k$, there are at most $1 + 3k + k^2$ choices of the set $T^{\mathtt{big}}$. For each $T^{\mathtt{big}}$, computing $X_b \cup N_G(S(T^{\mathtt{big}}))$ and verifying if it is a solution to $(\mathcal{I}_b, \mathcal{P})$ interrogated by S takes $O(n^2)$ time. Therefore Step 6.7 takes $O(2^{O(k^2 \log k)} n^2 \log n)$ time for all subcases.

This finishes the description of the fixed-parameter algorithm for N-MWCU.

7. Lower bound for big alphabet size. In this section we prove that the dependence on s in the algorithm from Theorem 2 is probably essential, even for the edge-deletion case and in the classical setting, when every vertex has a full list of possible labels and the partial permutations on edges are required to be permutations. We define formally the problem as follows.

EDGE UNIQUE LABEL COVER (k) **Parameter:** k
Input: An undirected graph G, a finite alphabet Σ of size s, an integer k, and for each vertex $v \in V(G)$ a set $\phi_v \subseteq \Sigma$ and for each edge $e \in E(G)$ and each its endpoint v a partial permutation $\psi_{e,v}$ of Σ, such that if $e = uv$, then $\psi_{e,u} = \psi_{e,v}^{-1}$.
Question: Does there exist a set $F \subseteq E(G)$ of size at most k and a function $\Psi : V(G) \to \Sigma$ such that for any $v \in V(G)$ we have $\Psi(v) \in \phi_v$ and for any $uv \in E(G) \setminus F$ we have $(\Psi(u), \Psi(v)) \in \psi_{uv,u}$?

THEOREM 65. EDGE UNIQUE LABEL COVER *(k) is* $W[1]$*-hard, even in the restricted case, when* $\phi_v = \Sigma$ *for all* $v \in V(G)$ *and* $\psi_{uv,u}, \psi_{uv,v}$ *are permutations for all* $uv \in E(G)$.

Before we proceed to the proof, we state that this restricted case is not easier than the general one.

LEMMA 66. *There exists a polynomial time algorithm that, given an instance*

$$I = (G, \Sigma, k, (\phi_v)_{v \in V(G)}, (\psi_{e,v})_{e \in E(G), v \in e})$$

of EDGE UNIQUE LABEL COVER*, outputs an equivalent instance*

$$I' = (G', \Sigma', k', (\phi'_v)_{v \in V(G)}, (\psi'_{e,v})_{e \in E(G), v \in e}),$$

where $k' = k(k+2)$, $|\Sigma'| = |\Sigma| + k + 2$, $\phi'_v = \Sigma'$ *for all* $v \in V(G)$ *and* $\psi'_{e,v}$ *is a permutation for all* $e \in E(G), v \in e$.

Proof. The graph G' we are going to construct will be a multigraph, possibly with loops. Note that we can easily get rid of multiple edges and loops by subdividing every edge and loop and for each subdivision preserving the constraint on one of the obtained edges while setting the constraint on the other edge to be identity.

We start with setting $k' = k(k+2)$ and $\Sigma' = \Sigma \cup \Gamma$, where $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_{k+2}\}$ is the set of $k+2$ new symbols that do not belong to Σ. Now we construct the multigraph G' as follows. First, $V(G') = V(G)$. For every vertex $v \in V(G)$ we take an arbitrary permutation π_v of Σ' such that ϕ_v is exactly the set of labels that π_v stabilizes; note that this is possible due to $k + 2 \geq 2$. We create $k' + 1$ loops in v with π_v as the constraint. Then, for every edge $uv \in E(G)$ denote by $X_{uv,u}$ the set of labels from Σ that do not have an image in $\psi_{uv,u}$, and similarly denote by $X_{uv,v}$ the set of labels from Σ that do not have an image in $\psi_{uv,v}$. Let $\{\psi^i_{uv,u}\}_{i=1,\ldots,k+2}$ be an arbitrary family of permutations of Σ', such that

- each $\psi^i_{uv,u}$ extends $\psi_{uv,u}$;
- each label $\alpha \in X_{uv,u} \cup \Gamma$ is mapped to pairwise different labels in $\psi^i_{uv,u}$ for $i = 1, 2, \ldots, k+2$;
- each label $\beta \in X_{uv,v} \cup \Gamma$ is mapped to pairwise different labels in $\psi^i_{uv,v} = \left(\psi^i_{uv,u}\right)^{-1}$ for $i = 1, 2, \ldots, k+2$.

Observe that as $|\Gamma| = k+2$, one can find such family $\{\psi^i_{uv,u}\}_{i=1,\ldots,k+2}$ by enumerating $X_{uv,u} \cup \Gamma$ and $X_{uv,v} \cup \Gamma$ in arbitrary orders, fixing one bijection between them and shifting it cyclicly $k+1$ times. Between u and v we insert the set of $k+2$ edges $P_{uv} = \{uv^i\}_{i=1,2,\ldots,k+2}$, imposing the constraints $(\psi^i_{uv,u}, \psi^i_{uv,v})$ on uv^i. Finally, we set $\phi'_v = \Sigma'$ for all $v \in V(G)$. This concludes the construction. We are left with a formal proof of the equivalence.

Assume first that there exists a set of edges $F \subseteq E(G)$, $|F| \leq k$, such that $G \setminus F$ admits a labeling Ψ respecting constraints in the input instance I. Let $F' = \{e^i : e \in F\}$; note that $|F'| = (k+2)|F| \leq k'$. A direct check shows that Ψ is also a correct labeling in $G' \setminus F'$, which proves that F' is a solution to the instance I'.

Now assume that there exists a set of edges $F' \subseteq E(G')$, $|F'| \leq k'$, such that $G' \setminus F'$ admits a labeling Ψ' respecting constraints in the output instance I'. Note that for each $v \in V(G)$ we have that $\Psi'(v) \in \phi_v$, as otherwise the set F' would need to contain $k'+1$ loops at v. Let $F \subseteq E(G)$ be the set of edges uv of G such that $(\Psi'(u), \Psi'(v)) \notin \psi_{uv,u}$. Clearly, F is a solution in the instance I as Ψ' is a correct labeling of $G \setminus F$. It remains to prove that $|F| \leq k$. Assume otherwise, i.e., $|F| \geq k+1$.

Consider an edge $uv \in E(G)$ such that $(\Psi'(u), \Psi'(v)) \notin \psi_{uv,u}$. We claim that $|F' \cap P_{uv}| \geq k+1$. If $\Psi'(u)$ belongs to the domain of $\psi_{uv,u}$, then all the constraints $\psi^i_{uv,u}$ map $\Psi'(u)$ to a label different than $\Psi'(v)$. Hence $P_{uv} \subseteq F'$ and the claim holds. Otherwise, $\Psi'(v)$ is mapped to $k+2$ different images in constraints $\psi^i_{uv,u}$, which means that at least $k+1$ of them must be different than $\Psi'(v)$. The corresponding edges have to be contained in F' and the claim holds in this case as well. As $|F| \geq k+1$, we have that $|F'| \geq (k+1)^2 = k'+1$, which is a contradiction. ∎

We are now ready to prove Theorem 65.

Proof of Theorem 65. By Lemma 66, we may consider the general problem definition, where we allow lists in vertices and partial permutations as constraints imposed on edges.

We provide a parameterized reduction from the Multicolored Clique problem, which is known to be W[1]-hard [24].

Multicolored Clique	**Parameter:** k
Input: An undirected graph H with vertices partitioned into k parts $V_0, V_1, \ldots, V_{k-1}$, such that H does not contain edges connecting vertices from the same part V_i, for $i = 0, 1, \ldots, k-1$.	
Question: Is there a clique C in G of size k?	

Observe that by the assumption on the structure of H, the clique C has to contain exactly one vertex from each part V_i. Moreover, by adding independent vertices we can assume that each part V_i is of the same size n. In each part V_i fix an arbitrary ordering of vertices $v^i_0, v^i_1, \ldots, v^i_{n-1}$.

Now, we are going to construct an instance $(G, \Sigma, k', (\phi_v)_{v \in V(G)}, (\psi_{e,v})_{e \in E(G), v \in e})$ that is a YES instance of Edge Unique Label Cover iff H contains a clique of size k. As the construction will be performed in polynomial time and $k' = k^2$, this gives the promised parameterized reduction.

We take $\Sigma = \{0,1,2,\ldots,n\} \times \{0,1,2,\ldots,n\}$ and let $\Lambda = \{0,1,\ldots,n-1\} \times \{0,1,\ldots,n-1\} \subseteq \Sigma$. For every part V_i we create a cycle C_i of length kn. Denote the vertices of C_i by $u_0^i, u_1^i, \ldots, u_{kn-2}^i, u_{kn-1}^i$ in the order of their appearance on the cycle. For every vertex u_p^i let $\mathrm{next}(u_p^i)$ be the next vertex on the cycle C_i, i.e., u_{p+1}^i if $p < kn-1$ and u_0^i if $p = kn-1$. Let $e(u_p^i)$ be the edge connecting u_p^i with $\mathrm{next}(u_p^i)$.

On every edge of the cycle C_i we impose a constraint given by the permutation $\pi_0((a,b)) = (a-1,b)$, where the numbers behave cyclicly modulo $n+1$. More precisely, the constraint on the edge $e(u_p^i)$ states that the label of $\mathrm{next}(u_p^i)$ has the first coordinate decremented by 1 modulo $n+1$ comparing to the label of u_p^i. Now, for every $i \neq j$, $0 \leq i,j < k$, we create an edge $u_{j\cdot n}^i u_{i\cdot n}^j$ with constraint given by the partial permutation $\sigma_{i,j} = \{((p,q),(q,p)) \mid v_p^i v_q^j \in E(H)\}$. In other words, from the domain of the permutation $\sigma((a,b)) = (b,a)$ we remove out all the pairs that contain $n+1$ and all the pairs that correspond to nonedges between V_i and V_j. Finally, we set $\phi_v = \Lambda$ for every $v \in V(G)$ and $k' = k^2$. This concludes the construction.

Let us first assume that C is a clique of size k in H and let $\{v_{c_i}^i\} = V(C) \cap V_i$. We construct

- a set of edges $F = \{e(u_{jn+c_i}^i) \mid 0 \leq i,j < k\}$;
- a labeling $\Psi(u_p^i) = ((c_i - p) \mod n, c_{q/n})$, where u_q^i is the closest next vertex on the cycle that has lower index being a multiplicity of n, i.e., $q/n = \lceil p/n \rceil \mod n$.

Obviously, $|F| = k'$. Let us check that Ψ is a correct labeling of $G \setminus F$. Clearly, $\Psi(v) \in \Lambda = \phi_v$ for any $v \in V(G)$. Consider any edge $e(u_p^i) \notin F$. As $p \mod n \neq c_i$, we have that $\Psi(u_p^i) = (x,y)$ for some $x > 0$ and $\Psi(\mathrm{next}(u_p^i)) = (x-1,y)$; hence, these constraints are satisfied. Now consider any edge of the form $u_{j\cdot n}^i u_{i\cdot n}^j$ for $i \neq j$, $0 \leq i,j < k$. By the construction of Ψ we have that $\Psi(u_{j\cdot n}^i) = (c_i, c_j)$ and $\Psi(u_{i\cdot n}^j) = (c_j, c_i)$. Recall that C is a clique, so $v_{c_i}^i v_{c_j}^j \in E(H)$. Hence, (c_i, c_j) lies in the domain of σ_{ij} and the constraint imposed on this edge is satisfied as well.

Let us now assume that there is a set of edges $F \subseteq E(G)$, $|F| \leq k'$, such that there exists a correct labeling Ψ of $G \setminus F$. First, we claim that for every n consecutive edges of every cycle C_i, F has to contain at least one of these edge. Otherwise there would be $n+1$ consecutive vertices $u_p^i, u_{p+1}^i, \ldots, u_{p+n}^i$ such that edges $u_{p+i}^i u_{p+i+1}^i$ do not belong to F for $i = 0,1,\ldots,n-1$ (indices behave cyclicly). It follows that if $\Psi(u_p^i) = (\ell, d)$ for some $\ell < n$, then we would have $\Psi(u_{p+\ell+1}^i) = (n,d)$, but n is a forbidden value in a label for every vertex. As every cycle C_i has length kn, it has to contain at least k edges from F. As $k' = k^2$, it has to contain exactly k edges from F. We can use again the claim to infer that between every two subsequent edges from F there must be exactly $n-1$ edges not from F, as otherwise there would be n consecutive edges not belonging to F. Moreover, the same argumentation yields that the vertices of each interval on the cycle between the two subsequent edges from F have to be labeled with $(n-1,d),(n-2,d),\ldots,(0,d)$, in this order, for some d depending on the interval, but constant within. Hence, for every cycle C_i we can find an integer $c_i \in \{0,1,\ldots,n-1\}$, such that $F = \{e(u_{jn+c_i}^i) \mid 1 \leq i,j < k\}$ and $\Psi(u_{jn}^i) = (c_i, d_j^i)$ for all $j = 0,1,\ldots,k-1$ and some numbers d_j^i.

We are going to prove that vertices $v_{c_i}^i$ for $i = 0,1\ldots,k-1$ induce a clique in H. Take parts V_i, V_j for $i \neq j$ and examine the edge $u_{j\cdot n}^i u_{i\cdot n}^j$ with constraint σ_{ij}. As $\Psi(u_{j\cdot n}^i) = (c_i, d_j^i)$, $\Psi(u_{i\cdot n}^j) = (c_j, d_i^j)$, and σ_{ij} swaps the elements of the pair, we find

that $d^i_j = c_j$ and $d^j_i = c_i$. Moreover, (c_i, c_j) is in the domain of σ_{ij} iff $v^i_{c_i} v^j_{c_j} \in E(H)$. Therefore, $v^i_{c_i}$ and $v^j_{c_j}$ are adjacent for all $i \neq j$ and we are done. ∎

8. Weights. We would like to note that using our technique we can solve a more general problem, where the graph is edge-weighted (or vertex-weighted, in the vertex-deletion setting), and the goal is, instead of minimizing the cardinality of the cutset, to find a cutset of size at most k, having minimum sum of weights of the edges (or vertices) it contains. For example for the problem considered in section 2, the formal definition is as follows.

Weighted Edge Unique Label Cover (W-E-ULC) **Parameter:** $k + s$
Input: An undirected (multi)graph G together with a weight function $\omega : E(G) \to \mathbb{R}_+$, a finite alphabet Σ of size s, an integer k, and for each edge $e \in E(G)$ and each of its endpoints v a permutation $\psi_{e,v}$ of Σ, such that if $e = uv$, then $\psi_{e,u} = \psi^{-1}_{e,v}$.
Question: What is the minimum weight of a set $X \subseteq E(G)$ of size at most k such that there exists a function $\Psi : V(G) \to \Sigma$ satisfying that for any $uv \in E(G) \setminus X$ we have $(\Psi(u), \Psi(v)) \in \psi_{uv,u}$?

Note that now we have to reformulate the bordered problem definition as well, because solutions to the bordered problem need to have a prescribed cardinality in order to make them comparable. Let us see it on the example of W-E-ULC.

By $\mathbb{P}(\mathcal{I}_b)$ we define the set of all pairs $\mathcal{P} = (\Psi_b, k_b)$, such that Ψ_b is a function from T_b to Σ and $0 \leq k_b \leq k$. We say that a set $X \subseteq E(G)$ is a solution to $(\mathcal{I}_b, \mathcal{P})$ if $|X| \leq k_b$, there exists a function $\Psi : V(G) \to \Sigma$ extending Ψ_b such that for any $uv \in E(G) \setminus X$ we have $(\Psi(u), \Psi(v)) \in \psi_{uv,u}$, and the sum of weights of edges in X is the minimum possible (comparing to all other sets X' satisfying the remaining constraints). The border problem is defined as follows.

Border W-E-ULC
Input: An W-E-ULC instance $\mathcal{I} = (G, \omega, \Sigma, k, (\psi_{e,v})_{e \in E(G), v \in e})$ with G being connected, and a set $T_b \subseteq V(G)$ of size at most $4k$; denote $\mathcal{I}_b = (\mathcal{I}, T_b)$.
Output: For each $\mathcal{P} \in \mathbb{P}(\mathcal{I}_b)$ output a solution $\texttt{sol}_{\mathcal{P}} = X_{\mathcal{P}}$ to $(\mathcal{I}_b, \mathcal{P})$ or output $\texttt{sol}_{\mathcal{P}} = \perp$ if such a solution does not exist.

Since, while finding a good separation, our algorithm does not perform any greedy choices, we almost leave the algorithm unchanged. Similarly, the recursive understanding step in the node-deletion problems is not affected significantly by this change. However, when solving a weighted problem, we need to be more careful in the final, high connectivity phase, as the existence of weights limits our possibilities of being greedy. In the following paragraphs we argue that the high connectivity phases of the algorithms presented in this paper can be adjusted to the weighted variants without greater effort.

Node Unique Label Cover. In the case of the Node Unique Label Cover problem, the high connectivity phase remains almost unchanged; however, we need to argue that all greedy steps used in this part of the algorithm are justified also in the weighted case. As in the case of the other problems, we start with guessing an interrogating set that (i) is disjoint with the solution Z we are looking for, (ii) contains all vertices of all small connected components of $G \setminus Z$, and (iii) contains a large connected set adjacent to each vertex of Z that is adjacent to the large connected component of $G \setminus Z$. The algorithm performs now two simple greedy steps: it checks

whether $Z = \emptyset$ is a solution and looks for not forsaken vertices without neighbors in S. Both steps can be easily justified in the weighted case, as we assume nonnegative weights and we require only $|X| \leq k_b$ in the border problem definition. The crucial observation—that there are only at most s reasonable labelings of the big stains (big connected components) of S—does not interfere with weights. In the final bounded search tree algorithm we argue that there is a limited number of vertices, out of which we need to delete at least one (the neighborhood branching rule), or that there are only a limited number of ways a small stain can be handled (the small stains rule). Both argumentations are oblivious to weights; note that this is also true in the second part of Lemma 39, where we argue about a greedy choice of a labeling in the case when the chosen labeling of the big stains can be consistently extended to a connected component of $G \setminus N[\Psi]$.

Steiner Cut. In the case of the Steiner Cut problem, we need to slightly change the final dynamic programming routine. Recall that in the high connectivity phase for this problem we first guess a set of edges S that (i) is disjoint with the solution Z we are looking for, (ii) contains a spanning tree of each small connected component of $G \setminus Z$, and (iii) contains a large spanning tree with an endpoint of an edge of the solution Z, for each such endpoint contained in the large connected component of $G \setminus Z$. Then we obtain a graph H by contracting the edges of S and identifying the images of the large trees of S (assumed in point (iii)) into the core vertex b. For each connected component B'_i of $H \setminus b$ we have two choices: we delete either all edges or no edges from $B'_i \cup \{b\}$. The choices between different components B'_i are independent, and we find the optimal solution via a simple dynamic programming routine. In the weighted case we need to add to the dynamic programming table one more dimension responsible for storing the cardinality of the constructed cutset, and the value in the table T is the minimum weight of a cutset of the prescribed cardinality.

N-MWCU. The simplicity of the high connectivity phase of the N-MWCU algorithm allows us to solve the weighted variant with almost no changes. Recall that in this phase we first guess a set S that (i) is disjoint with the solution Z we are looking for and (ii) covers all nonterminal vertices of small connected components of $G \setminus Z$. Then we argue that any inclusionwise minimal solution chooses at most one equivalence relation of $\mathcal{R}_b$ to be the set of terminals contained in the big connected component of $G \setminus Z$ and takes as the solution the neighborhood of all connected components of $G[S \cup T]$ that contain a terminal not contained in the selected equivalence class. The same argumentation holds in the case of nonnegative weights; note that in the border problem we require $|X| \leq k_b$ (instead of maybe more natural $|X| = k_b$), and thus we may consider only inclusionwise minimal solutions.

REFERENCES

[1] A. Agarwal, N. Alon, and M. Charikar, *Improved approximation for directed cut problems*, in Proceedings of STOC'07, 2007, pp. 671–680.

[2] N. Alon, R. Yuster, and U. Zwick, *Color-coding*, J. ACM, 42 (1995), pp. 844–856.

[3] S. Arora, B. Barak, and D. Steurer, *Subexponential algorithms for unique games and related problems*, J. ACM, 62 (2015), 42, doi:10.1145/2775105.

[4] V. Bafna, P. Berman, and T. Fujito, *A* 2-approximation algorithm for the undirected feedback vertex set problem, SIAM J. Discrete Math., 12 (1999), pp. 289–297.

[5] H. L. Bodlaender, F. V. Fomin, D. Lokshtanov, E. Penninkx, S. Saurabh, and D. M. Thilikos, *(Meta) kernelization*, in Proceedings of FOCS'09, 2009, pp. 629–638.

[6] N. Bousquet, J. Daligault, and S. Thomassé, *Multicut is FPT*, in Proceedings of STOC'11, 2011, pp. 459–468.

[7] K. Bringmann, D. Hermelin, M. Mnich, and E. J. van Leeuwen, *Parameterized complexity dichotomy for Steiner multicut*, in Proceedings of the 32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, 2015, Garching, Germany, E. W. Mayr and N. Ollinger, eds., LIPIcs 30, Schloss Dagstuhl, Leibniz-Zentrum fuer Informatik, 2015, pp. 157–170, doi:10.4230/LIPIcs.STACS.2015.157.

[8] M. Charikar, K. Makarychev, and Y. Makarychev, *Near-optimal algorithms for maximum constraint satisfaction problems*, ACM Trans. Algorithms, 5 (2009), doi:10.1145/1541885.1541893.

[9] C. Chekuri, S. Guha, and J. Naor, *The Steiner k-cut problem*, SIAM J. Discrete Math., 20 (2006), pp. 261–271.

[10] J. Chen, Y. Liu, and S. Lu, *An improved parameterized algorithm for the minimum node multiway cut problem*, Algorithmica, 55 (2009), pp. 1–13.

[11] 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 (2008).

[12] R. H. Chitnis, M. Cygan, M. Hajiaghayi, M. Pilipczuk, and M. Pilipczuk, *Designing FPT algorithms for cut problems using randomized contractions*, in proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, 2012, pp. 460–469, doi:10.1109/FOCS.2012.29.

[13] R. H. Chitnis, M. Cygan, M. T. Hajiaghayi, and D. Marx, *Directed subset feedback vertex set is fixed-parameter tractable*, ACM Trans. Algorithms, 11 (2015), 28, doi:10.1145/2700209.

[14] R. H. Chitnis, M. Hajiaghayi, and D. Marx, *Fixed-parameter tractability of directed multiway cut parameterized by the size of the cutset*, SIAM J. Comput., 42 (2013), pp. 1674–1696, doi:10.1137/12086217X.

[15] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, *Parameterized Algorithms*, Springer, New York, 2015, doi:10.1007/978-3-319-21275-3.

[16] M. Cygan, D. Lokshtanov, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, *Minimum bisection is fixed parameter tractable*, in Proceedings of the Symposium on Theory of Computing, STOC 2014, New York, 2014, D. B. Shmoys, ed., ACM, 2014, pp. 323–332, doi:10.1145/2591796.2591852.

[17] M. Cygan, M. Pilipczuk, and M. Pilipczuk, *On group feedback vertex set parameterized by the size of the cutset*, in Proceedings of WG'12, 2012, pp. 194–205.

[18] M. Cygan, M. Pilipczuk, M. Pilipczuk, and J. O. Wojtaszczyk, *On multiway cut parameterized above lower bounds*, ACM Trans. Comput. Theory, 5 (2013), 3, doi:10.1145/2462896.2462899.

[19] M. Cygan, M. Pilipczuk, M. Pilipczuk, and J. O. Wojtaszczyk, *Subset feedback vertex set is fixed-parameter tractable*, SIAM J. Discrete Math., 27 (2013), pp. 290–309.

[20] R. G. Downey and M. R. Fellows, *Parameterized Complexity*, Springer, New York, 1999.

[21] R. G. Downey and M. R. Fellows, *Fundamentals of Parameterized Complexity*, Texts in Comput. Sci., Springer, New York, 2013, doi:10.1007/978-1-4471-5559-1.

[22] G. Even, J. Naor, B. Schieber, and M. Sudan, *Approximating minimum feedback sets and multicuts in directed graphs*, Algorithmica, 20 (1998), pp. 151–174.

[23] G. Even, J. Naor, and L. Zosin, *An* 8*-approximation algorithm for the subset feedback vertex set problem*, SIAM J. Comput., 30 (2000), pp. 1231–1252.

[24] M. R. Fellows, D. Hermelin, F. A. Rosamond, and S. Vialette, *On the parameterized complexity of multiple-interval graph problems*, Theoret. Comput. Sci., 410 (2009), pp. 53–61.

[25] J. Flum and M. Grohe, *Parameterized Complexity Theory*, Texts in Theoret. Comput. Sci. EATCS Ser., Springer, New York 2006, http://www.worldcat.org/isbn/3540299521.

[26] F. V. Fomin, D. Lokshtanov, S. Saurabh, and D. M. Thilikos, *Linear kernels for (connected) dominating set on H-minor-free graphs*, in Proceedings of SODA'12, 2012, pp. 82–93.

[27] N. Garg, V. V. Vazirani, and M. Yannakakis, *Approximate max-flow min-(multi)cut theorems and their applications*, SIAM J. Comput., 25 (1996), pp. 235–251.

[28] N. Garg, V. V. Vazirani, and M. Yannakakis, *Primal-dual approximation algorithms for integral flow and multicut in trees*, Algorithmica, 18 (1997), pp. 3–20, doi:10.1007/bf02523685.

[29] N. Garg, V. V. Vazirani, and M. Yannakakis, *Multiway cuts in node weighted graphs*, J. Algorithms, 50 (2004), pp. 49–61.

[30] F. Granot and R. Hassin, *Multi-terminal maximum flows in node capacitated networks*, Discrete Appl. Math, 13 (1986), pp. 157–163.

[31] M. Grohe and D. Marx, *Structure theorem and isomorphism test for graphs with excluded topological subgraphs*, SIAM J. Comput., 44 (2015), pp. 114–159, doi:10.1137/120892234.
[32] S. Guillemot, *FPT algorithms for path-transversal and cycle-transversal problems*, Discrete Optim., 8 (2011), pp. 61–71.
[33] S. Guillemot, *Parameterized complexity and approximability of the longest compatible sequence problem*, Discrete Optim., 8 (2011), pp. 50–60.
[34] Y. Iwata, M. Wahlström, and Y. Yoshida, *Half-Integrality, LP-Branching and FPT Algorithms*, arXiv:1310.2841, 2013.
[35] D. R. Karger, *Minimum cuts in near-linear time*, J. ACM, 47 (2000), pp. 46–76.
[36] D. R. Karger, P. N. Klein, C. Stein, M. Thorup, and N. E. Young, *Rounding algorithms for a geometric embedding of minimum multiway cut*, Math. Oper. Res., 29 (2004), pp. 436–461.
[37] K. Kawarabayashi and M. Thorup, *The minimum k-way cut of bounded size is fixed-parameter tractable*, in Proceedings of FOCS'11, 2011, pp. 160–169.
[38] S. Khot, *On the power of unique 2-prover 1-round games*, in Proceedings of STOC'02, 2002, pp. 767–775.
[39] S. Khot, *On the unique games conjecture (invited survey)*, in Proceedings of the IEEE Conference on Computational Complexity, 2010, pp. 99–121.
[40] S. Kratsch, M. Pilipczuk, M. Pilipczuk, and M. Wahlström, *Fixed-parameter tractability of multicut in directed acyclic graphs*, SIAM J. Discrete Math., 29 (2015), pp. 122–144, doi:10.1137/120904202.
[41] S. Kratsch and M. Sorge, *On kernelization and approximation for the vector connectivity problem*, in Proceedings of the 10th International Symposium on Parameterized and Exact Computation, IPEC 2015, Patras, Greece, T. Husfeldt and I. A. Kanj, eds., of LIPIcs 43, Schloss Dagstuhl Leibniz-Zentrum fuer Informatik, 2015, pp. 377–388, doi:10.4230/LIPIcs.IPEC.2015.377.
[42] D. Lokshtanov and D. Marx, *Clustering with local restrictions*, Inform and Comput., 222 (2013), pp. 278–292.
[43] D. Lokshtanov, N. S. Narayanaswamy, V. Raman, M. S. Ramanujan, and S. Saurabh, *Faster parameterized algorithms using linear programming*, ACM Trans. Algorithms, 11 (2014), pp. 15:1–15:31, doi:10.1145/2566616.
[44] D. Marx, *Parameterized graph separation problems*, Theoret. Comput. Sci., 351 (2006), pp. 394–406.
[45] D. Marx, B. O'Sullivan, and I. Razgon, *Finding small separators in linear time via treewidth reduction*, ACM Trans. Algorithms, 9 (2013), 30, doi:10.1145/2500119.
[46] D. Marx and I. Razgon, *Fixed-parameter tractability of multicut parameterized by the size of the cutset*, SIAM J. Comput., 43 (2014), pp. 355–388, doi:10.1137/110855247.
[47] H. Nagamochi and T. Ibaraki, *A linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph*, Algorithmica, 7 (1992), pp. 583–596.
[48] J. Naor and Y. Rabani, *Tree packing and approximating k-cuts*, in Proceedings of SODA'01, 2001, pp. 26–27.
[49] M. Naor, L. J. Schulman, and A. Srinivasan, *Splitters and near-optimal derandomization*, in Proceedings of FOCS'95, 1995, pp. 182–191.
[50] N. S. Narayanaswamy, V. Raman, M. S. Ramanujan, and S. Saurabh, *LP can be a cure for parameterized problems*, in Proceedings of the 29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012, Paris, France, C. Dürr and T. Wilke, eds., LIPIcs 14, Schloss Dagstuhl, Leibniz-Zentrum fuer Informatik, 2012, pp. 338–349, doi:10.4230/LIPIcs.STACS.2012.338.
[51] M. Pilipczuk and M. Wahlström, *Directed multicut is W[1]-hard, even for four terminal pairs*, in Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, R. Krauthgamer, ed., SIAM, 2016, pp. 1167–1178, doi:10.1137/1.9781611974331.ch81.
[52] R. Ravi and A. Sinha, *Approximating k-cuts via network strength as a lagrangean relaxation*, European J. Oper. Res., 186 (2008), pp. 77–90.
[53] I. Razgon, *Large Isolating Cuts Shrink the Multiway Cut*, CoRR abs/1104.5361, 2011.
[54] I. Razgon and B. O'Sullivan, *Almost 2-SAT is fixed-parameter tractable*, J. Comput. System Sci., 75 (2009), pp. 435–450.
[55] B. A. Reed, K. Smith, and A. Vetta, *Finding odd cycle transversals*, Oper. Res. Lett., 32 (2004), pp. 299–301, doi:10.1016/j.orl.2003.10.009.
[56] P. D. Seymour, *Packing directed circuits fractionally*, Combinatorica, 15 (1995), pp. 281–288.
[57] M. Wahlström, *Half-integrality, LP-branching and FPT algorithms*, in Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, OR, C. Chekuri, ed., SIAM, 2014, pp. 1762–1781, doi:10.1137/1.9781611973402.128.