2012 IEEE 53rd Annual Symposium on Foundations of Computer Science

Designing FPT algorithms for cut problems using randomized contractions

Rajesh Chitnis*, Marek Cygan†, MohammadTaghi Hajiaghayi*, Marcin Pilipczuk‡ and Michał Pilipczuk§
*Department of Computer Science, University of Maryland, College Park, USA. Email: {rchitnis,hajiagha}@cs.umd.edu
†IDSIA, University of Lugano, Switzerland. Email: marek@idsia.ch
‡Institute of Informatics, University of Warsaw, Poland. Email: malcin@mimuw.edu.pl
§Department of Informatics, University of Bergen, Norway. Email: michal.pilipczuk@ii.uib.no

***Abstract*—We introduce a new technique for designing fixed-parameter algorithms for cut problems, namely *randomized contractions*. With our framework:**

- **We obtain the first FPT algorithm for the parameterized version of the UNIQUE LABEL COVER problem, with single exponential dependency on the size of the cutset and the size of the alphabet. As a consequence, we extend the set of the polynomial time solvable instances of UNIQUE GAMES to those with at most $O(\sqrt{\log n})$ violated constraints.**
- **We obtain a new FPT algorithm for the STEINER CUT problem with exponential speed-up over the recent work of Kawarabayashi and Thorup (FOCS'11).**
- **We show how to combine considering 'cut' and 'uncut' constraints at the same time. 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 FPT algorithm with single exponential dependency on 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 (STACS'10), which yields algorithms with double exponential running time.**

An interesting aspect of our algorithms is that they can handle real weights; to the best of our knowledge, the technique of important separators does not work in the weighted version.

***Keywords*-fixed parameter tractability; unique label cover; graph cut problems;**

I. 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 of 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 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. [2]–[12]).

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 the 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 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 an interested reader to the books of Downey and Fellows [13] or Flum and Grohe [14].

Graph separation problems in the context of parameterized complexity were probably first considered in the seminal work of Marx [15]. 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. Perhaps the most fruitful consequence of his work 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. The technique has found a number of applications

The full version of this work can be found at [1]. The first, second and third author are 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). The second author was partially supported by the ERC grant NEWNET, reference 279352. The second and fourth author were partially supported by NCN grant N206567140 and Foundation for Polish Science. The third author is also with AT&T Labs–Research. The fifth author was partially supported by ERC grant "Rigorous Theory of Preprocessing", reference 267959.

0272-5428/12 $26.00 © 2012 IEEE
DOI 10.1109/FOCS.2012.29

[15]–[23], including proving fixed-parameter tractability of MULTICUT parameterized by the cutsize only, which was resolved by Marx and Razgon [22] and, independently, by Bousquet et al. [24], after resisting attacks as a long-standing open problem. The latest advances, by Chitnis et al. [18] and Kratsch et al. [25], try to find a proper understanding of this notion in directed graphs.

The important separators technique is based on greedy arguments, which unfortunately makes this approach difficult to extend. Consider, for instance, adding constraints of type 'uncut', i.e., we would like to find 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 existence of some paths. Similar obstacles arise when considering problems with weights or in directed setting.

Another approach has been recently presented by Marx, O'Sullivan and Razgon [26]. Intuitively, the authors prove that one can find a subset of vertices in which all the minimal cutsets of size at most k are contained, and which induces (for an appropriate meaning of 'induce') a graph of treewidth bounded by an exponential function of k. Thus, having expressed the problem in $\mathbf{MSO}_2$ logic, we can use Courcelle's lemma on the obtained tree decomposition. The technique can tackle more general constraints than important separators, such as 'uncut' constraints, and works elegantly for easy cut problems with strong parameterizations. A drawback is that algorithms constructed in this manner almost always have double exponential dependence on k of the running time, as we run a dynamic program on a tree decomposition of width exponential in k.

A. Our techniques

We introduce a new technique, called *randomized contractions*, of constructing fixed-parameter algorithms for graph cut problems. The technique is based on a WIN/WIN approach: 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. First we test, whether the graph admits a well-balanced separation. If this is the case, we run the algorithm recursively on one of the sides for all possible behaviors on the boundary, for each marking some optimal solution. Then we argue that unmarked parts can be conveniently reduced. However, if no separation is present, we find that after removing the solution the graph can split only into a bounded number of connected components of bounded size, and at most one connected component that can be arbitrarily large. Then, we randomly contract parts of the graph; the highly-connected structure ensures us that with high probability the optimal solution will be 'highlighted' by the contraction step, so that we can easily extract it. Intuitively, the event we aim for is that all the components of bounded size and a sufficiently large neighborhood of the solution are distinguished.

Finding a small separator and recursively reducing one of the sides is the core idea of many algorithms for parameterized problems. For example, a novel concept of finding and reducing *protrusions*, large subgraphs of constant treewidth and constant boundary, led to construction of linear kernels for a large number of problems on classes of graphs with topological constraints [27], [28]. Perhaps a better example is the recent fixed-parameter algorithm for the k-WAY CUT problem of Kawarabayashi and Thorup [29], which is in fact the original motivation of our technique. The authors observe that if they find a small, well-balanced separation in the graph, they may reduce one of the sides up to bounded size, in a very similar manner as we do. Basing on this, they analyze the graph in a minor-style manner and whenever they encounter a feasible separation, they can apply the reduction.

One of the tools we use in our algorithms is the color coding technique introduced by Alon et al. [30] to solve some special cases of the SUBGRAPH ISOMORPHISM problem. The main idea 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. Our technique is both similar and different to color coding. The similarity lies in the first application of color coding in our approach, i.e., finding a feasible separation to make the reduction step. However, when no good separation can be found, we use different ideas not only to highlight the solution, but also to expose the highly connected structure of the graph. 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 et al. [31]. 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 I.1. *Given a set U of size n, and integers $0 \leq a, b \leq n$, one can in $O(2^{O(\min(a,b)\log(a+b))} n \log n)$ time construct a family $\mathcal{F}$ of at most $O(2^{O(\min(a,b)\log(a+b))} \log n)$ 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 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.

B. Our results

We use the technique 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) 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 the 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 GROUP FEEDBACK VERTEX SET problem [32], and hence ODD CYCLE TRANSVERSAL and 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 [33] 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 $\epsilon, \delta > 0$, there exists an alphabet size $|\Sigma|(\epsilon, \delta)$, 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 et al. [34] presented a breakthrough subexponential time algorithm, which in $2^{O(|\Sigma| n^{\epsilon})}$ 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 [35] 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 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 I.2. *There is an $O(2^{O(k^2 \log s)} n^4 \log n)$ time algorithm solving* NODE UNIQUE LABEL COVER*, and consequently* EDGE 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, with currently best by Charikar et al. [36], working under the assumption $|OPT| \geq (1-\epsilon)|E(G)|$, 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 I.2 is probably necessary, since the problem parameterized by the cutsize only is $W[1]$-hard. Hence, 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 [13] or Flum and Grohe [14]. We consider this result an interesting counterposition of the parameterized status of GROUP FEEDBACK VERTEX SET [37], which is FPT even when the group size is not a parameter.

Theorem I.3. *The* EDGE UNIQUE LABEL COVER *problem, and consequently* NODE UNIQUE LABEL COVER*, is $W[1]$-hard when parameterized by k only.*

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 [29][1].

Theorem I.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] In [29] the authors solve the k-WAY CUT problem, however a straightforward generalization of their algorithm solves the STEINER CUT problem as well.

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 $G \setminus X$ if and only if $(u, v) \in \mathcal{R}$?

Fixed-parameter tractability of this problem can be derived from the framework of Marx, Razgon and O'Sullivan [26], complemented with a reduction of the number of equivalence classes of $\mathcal{R}$ in flavour of the reduction for MULTIWAY CUT of Razgon [38]. However, the dependence on k of the running time is double exponential. Using our framework we provide an algorithm working in $O(2^{O(k^2 \log k)} n^4 \log n)$ time, which can be constructed by combining *(i)* ideas from the edge-deletion variant, *(ii)* the general approach to node-deletion problems, *(iii)* the aforementioned reduction of the number of equivalence classes of $\mathcal{R}$ and *(iv)* simple reductions of terminals sharing large parts of neighborhoods.

Theorem I.5. *There is an $O(2^{O(k^2 \log k)} n^4 \log n)$ time algorithm solving* NODE MULTIWAY CUT-UNCUT.

In this extended abstract we describe the general idea of the technique in Section II, illustrating them on the edge-deletion version of the N-MWCU problem. Then, in Section III, we sketch the algorithm for the NODE UNIQUE LABEL COVER problem; we find this problem best-suited to present the challenges that arise when considering node-deletion variants, and means of overcoming them. Section IV is devoted to brief concluding remarks. The complete proofs of all results can be found in the full version of the paper available at [1].

II. ILLUSTRATION

In this section we present the outline of the technique, illustrating it with a running example of the EDGE MULTIWAY CUT-UNCUT problem, the edge-deletion variant of NODE MULTIWAY CUT-UNCUT.

EDGE MULTIWAY CUT-UNCUT (E-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 $X \subseteq E(G)$ of at most k edges such that for any $u, v \in T$, the vertices u and v belong to the same connected component of $G \setminus X$ if and only if $(u, v) \in \mathcal{R}$?

As the edge-deletion variant can be easily reduced to the node-deletion variant, the fixed-parameter tractability of E-MWCU follows from Theorem I.5. However, we find this particular problem best-suited to serve as an illustration of our technique, to complement the description of abstract and often intuitive concepts with a 'real-life' example. Throughout this section we sketch an algorithm with running time $O(2^{O(k^2 \log k)} n^4 \log n)$, resolving the E-MWCU problem. As we consider the edge-deletion version, we use edge cuts throughout this section. However, as our general framework can be also applied to node-deletion problems, we comment along the description where additional argumentation is needed.

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. Hence, we can assume that $\mathcal{R}$ has at most $k+1$ equivalence classes, as otherwise we may safely return a negative answer.

A. High-level intuition

The starting point of our approach is an observation that if two vertices of the graph v, w can be connected via $k+1$ edge-disjoint paths, then after removing at most k edges they remain in the same connected component. Therefore, in this situation we may apply a reduction rule that simplifies the graph basing on the knowledge that v and w cannot be separated by a small separator. For example, in the EDGE MULTIWAY CUT-UNCUT problem it is safe to simply identify v and w. Note that in this way we may obtain a multigraph.

Of course, this simple reduction cannot solve the problem completely in general, as the input graph can have, for instance, degrees bounded by a constant. However, we still would like to pursue the intuition that making the input graph well-connected leads to better understanding of its structure. To this end, we introduce the notion of *good separations*.

Definition II.1. Let G be a connected graph. A partition (V_1, V_2) of $V(G)$ is called a (q, k)-*good edge separation*, if *(i)* $|V_1|, |V_2| > q$; *(ii)* $|\delta(V_1, V_2)| \leq k$; *(iii)* $G[V_1]$ and $G[V_2]$ are connected.

In the first phase, 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 introducing 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 *high connectivity phase*, where we exploit this highly connected structure to identify the solution.

B. Recursive understanding

Lemma II.2. *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 sketch: Consider a family $\mathcal{F}$ obtained via Lemma I.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, 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 of (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 [39] 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 [40], 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. ■

Having found a good edge separation we can proceed to simplification of one of the sides. To this end, 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 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 E-MWCU problem.

The behavior on the border terminals, whose number will be bounded by $2k$, is defined by a pair of equivalence relations $\mathcal{P} = (\mathcal{R}_b, E_b)$. $\mathcal{R}_b$ is defined on $T \cup T_b$, but we require that $\mathcal{R}_b|_T = \mathcal{R}$, i.e., $\mathcal{R}_b$ extends $\mathcal{R}$. Informally, $\mathcal{R}_b$ expresses which border terminals are required to be in the same connected with which terminals after removing the solution. The second relation E_b is defined on T_b and we require it to be a subset of $\mathcal{R}_b$. Informally, E_b expresses, which pairs of border terminals are assumed to be reachable via paths outside the considered subgraph. The reader may view E_b as a *torso* operation, commonly used in the context of graph minors; thus, in the border problem we prepare ourselves for *all* the possible torso operations. For an instance of the border problem $\mathcal{I}_b$ by $\mathbb{P}(\mathcal{I}_b)$ we denote the set of possible pairs $\mathcal{P} = (\mathcal{R}_b, E_b)$; note that $|\mathbb{P}(\mathcal{I}_b)| \leq f(k) = 2^{O(k\log k)}$, as the number of equivalence classes of $\mathcal{R}$ is bounded by $k+1$. Formally, we say that a set $X \subseteq E(G)$ is a *solution* to $(\mathcal{I}_b, \mathcal{P})$ if $|X| \leq k$ and in the graph $G_{\mathcal{P}} := (V(G), E(G) \cup E_b)$ after removing X every two vertices $u, v \in T \cup T_b$ are in the same connected component if and only if they are equivalent with respect to $\mathcal{R}_b$. Note that the graph $G_{\mathcal{P}}$ is exactly G with torso operation performed on equivalence classes of E_b. We can now formally define the border problem:

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

BORDER E-MWCU generalizes E-MWCU: we ask for $T_b = \emptyset$ and take the output for the pair $(\mathcal{R}, \emptyset)$.

Now assume that (V_1, V_2) is a (q,k)-good separation of the graph G, for the input instance $\mathcal{I}_b = (G, T, \mathcal{R}, k, T_b)$ of BORDER E-MWCU, where $q = k \cdot f(k) + 1$ is the maximum number of edges that can appear on the output plus 1. As $|T_b| \leq 2k$, at least one of the sides contains at most k border terminals. Without loss of generality we assume that $|V_1 \cap T_b| \leq k$. Now consider an instance $\mathcal{I}_b^* = (G[V_1], T \cap V_1, \mathcal{R}|_{T\cap V_1}, k, (T_b \cap V_1) \cup (V_1 \cap V(\delta(V_1, V_2))))$ of BORDER E-MWCU. In other words, we trim the instance to the part V_1 and incorporate all the vertices incident to V_2 to the set of border terminals. Note that $\mathcal{I}_b^*$ is a correct instance, as $|(T_b \cap V_1) \cup (V_1 \cap V(\delta(V_1, V_2)))| \leq k + k = 2k$. We now recursively solve the instance $\mathcal{I}_b^*$, obtaining a set of at most $q-1$ edges that appear in any solution for any behavior on border terminals. Consider the remaining edges, i.e., those that do not appear in any solution to $\mathcal{I}_b^*$ for any behavior on the terminals. It is not hard to see that all these edges can be in fact contracted, as it is useless to incorporate them in the solution to $\mathcal{I}_b$ for any behavior on the border terminals. Intuitively, for every solution that uses some of these edges we can replace the part contained in $G[V_1]$ with the optimal solution for $\mathcal{I}_b^*$ that imposes the same behavior on border terminals of $\mathcal{I}_b^*$, computed by the recursive call. We omit here the formal proof. After applying the contractions, part $G[V_1]$ shrinks to size at most q, as it is still connected and

contains at most $q-1$ edges. The sets of terminals, border terminals and relations are defined naturally.

We remark that the operation applied to reduce parts of the graph determined to be useless is problem-dependent. In E-MWCU we use edge contraction; however, more complex problems or node-deletion versions require more careful simplification rules.

We are left with estimating the running time of the algorithm. By Lemma II.2 the time required to find a (q,k)-good edge separation is $O(2^{O(k \log f(k))} n^3 \log n) = O(2^{O(k^2 \log k)} n^3 \log n)$; hence, the total running time is $O(2^{O(k^2 \log k)} n^4 \log n)$. We note that if f, 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.

C. High connectivity phase

We are left with the more involved part of our approach, namely, what to do when no (q,k)-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. We remark that we have this structure only in graph G, not $G_{\mathcal{P}}$. Fortunately, adding the edges of E_b will not be a problem.

Lemma II.3. *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.*

From now on, we always assume that $|V(C_i)| \leq q$ for $i = 1, 2, \ldots, \ell$, while C_0 can have unbounded size. In fact, as $|V(G)| > q(k+1)$ we find that $|V(C_0)| > q$. We refer to components C_i for $i = 1, 2, \ldots, \ell$ as to *small components*, while C_0 is called the *big component*. 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 small components — in the node-deletion setting we need additional tools here.

Fix some behavior on the border terminals $\mathcal{P} = (\mathcal{R}_b, E_b) \in \mathbb{P}(\mathcal{I}_b)$; we iterate through all of them, which gives $2^{O(k \log k)}$ 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 minimum. Let $C_0, \ldots, C_\ell$ be components of $G \setminus X$, as in Lemma II.3, where $|V(C_i)| \leq q$ for $i = 1, 2, \ldots, \ell$. 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 $A_2 = \bigcup_{u \in V(X) \cap V(C_0)} E(T_0^u)$. As X is minimum, we have that $|V(X) \cap V(C_0)| \leq k$ and hence $|A_2| \leq qk$.

Now, we would like to apply edge contractions once more, in order to condensate the high-connectivity between large connected subgraphs of G to single vertices. Let $\mathcal{F}$ be the family obtained from Lemma I.1 for the universe $E(G)$ and constants $a = (2q-1)k$ and $b = k$. By Lemma I.1, there exists $S_0 \in \mathcal{F}$ such that all the edges from $A_1 \cup A_2$ belong to S_0, whereas all the edges from X are not in S_0. We branch into $|\mathcal{F}|$ subcases labeled by the sets $S \in \mathcal{F}$. In each branch we either find a candidate for an optimal solution among edges that do not belong to S, or conclude that no such exists. We argue that if a (minimum) solution X exists, then some optimal solution will be found in the branch where S_0 is chosen.

For the sake of analysis, we present the routine performed in every branch assuming that we consider S_0. First, we can contract all the edges of S_0, as we seek a solution that is disjoint with S_0. Let H_0 be the graph obtained in this operation. Observe that each small component C_i, $i = 1, 2, \ldots, \ell$, forms exactly the subgraph contracted to a single vertex c_i. Moreover, all the vertices c_i can be incident only to edges from X, which are preserved due to the properties of S_0 and X being minimum. The set of terminals, border terminals and relations in H_0 are defined naturally as projections of the corresponding objects in G; if we encounter any mismatch, e.g., two terminals that are non-equivalent with respect to $\mathcal{R}$ are contracted onto the same vertex, we can safely terminate the branch. Observe that the branch when S_0 is considered is not terminated. Moreover, X survives the contractions and remains a feasible solution.

For a vertex $u \in V(H_0)$ we define its *weight* to be the number of vertices of G that were contracted onto it. A vertex $u \in V(H_0)$ is called *heavy* if it has weight at least $q+1$. Observe that all the vertices of $V(X) \cap V(C_0)$, that is, endpoints of edges from the solution that lie in C_0, are contracted onto heavy vertices. Now we may make use of the high-connectivity structure of the graph. As G does not admit any (q,k)-good edge separation, by Menger's theorem between each pair of big vertices in H_0 we can find $k+1$ edge-disjoint paths. This means that after removing the solution, all the big vertices are still in the same connected component. Hence, it is safe to identify them into one vertex b, which will be denoted the *core vertex*. Let H be the graph obtained after identification. The set of terminals and border terminals are defined naturally as before; again we can provide a negative answer if we encounter any mismatch during identifications. Moreover, all the edges from X are still present in H, as they do not connect big vertices, and X remains a feasible solution.

We remark that we can assume that the optimal deletion set is nonempty, as this particular possibility can be checked

in polynomial time. Hence, there exists at least one heavy vertex or otherwise we may terminate the branch. Therefore, the core vertex b is well-defined. See Figure 1 for an illustration.

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, edge-disjoint and b separates them. We claim that for every component B_i, the solution X contains either all edges of $E(B_i)$ or none of them. This follows directly from the fact that all the small components are contracted into single vertices, while all endpoints of edges from X that are contained in C_0 are contracted onto b.

We conclude the algorithm by showing how to find an optimal solution inside the graph H in $O(kn^2)$ time. First, if b is not a border terminal nor a terminal, we iterate through all the possible equivalence class of $\mathcal{R}_b$, with which the vertex b is in the same connected component (plus one possibility, with none of them). Observe that we can assume that the number of equivalence classes of $\mathcal{R}_b$ is bounded by $k+1$, so we have at most $k+2$ possibilities. Let D be the guessed subset of vertices from $T \cup T_b$ that are reachable from b after removing the solution. Observe that now we know exactly how the solution needs to look like. It simply needs to contain the entire edge sets of components that contain terminals or border terminals that do not belong to D, while all the other components may be left disjoint with the solution. The last claim is asserted by $E_b \subseteq \mathcal{R}_b$: the additional edges inserted in $G_{\mathcal{P}}$ cannot force us to include into the solution also an edge set of a component, that contains only terminals and border terminals from D. Having constructed a candidate for the solution, we can check if it is satisfies all the constraints in $G_{\mathcal{P}}$ in $O(n^2)$ time. It follows from the presented construction that when S_0 is considered, some solution of optimal size is found.

We remark that the last part of the algorithm, i.e., solving the problem inside the graph H, is highly problem-dependent. In our simple example of EDGE MULTIWAY CUT-UNCUT we were able to derive a simple routine with running time polynomial in k and n, but for more involved problems we can still need time exponential in k. All the other parts of the approach are to some extent generic and can be applied to various problems. We also would like to note that contractions of big trees on the side of C_0 were not necessary in this particular problem: one could iterate through a smaller family $\mathcal{F}$ assuming only contractions of small components into single vertices. However, we chose to present the algorithm in this way, as the idea of condensating the big component into the core vertex significantly strengthens the obtained structure of the graph, and is an essential ingredient in all the other our algorithms.

III. FPT ALGORITHM FOR NODE UNIQUE LABEL COVER

In this section we present a brief sketch of the algorithm for NODE UNIQUE LABEL COVER, with the emphasis on the difficulties that arise due to considering a node-deletion variant, as well as means that we use to overcome them. We note that the edge-deletion variant can be reduced to the node-deletion variant via a simple polynomial parameterized transformation.

Recall that in NODE UNIQUE LABEL COVER we are given an instance $(G, \Sigma, k, (\psi_{e,v})_{e \in E(G), v \in e})$ and we are to construct a set X of size at most k and a function $\Psi : V(G) \setminus X \to \Sigma$ that satisfies all constraints $\psi_{e,v}$ in $G \setminus X$. The relations $\psi_{e,u}$ are called *edge constraints*, function Ψ is called a *labeling* and set X is the *deletion set*. We can naturally extend the notion of labelings to subsets: a labeling of a subset $S \subseteq V(G)$ is a function from S to the alphabet that respects all the constraints in $G[S]$. Given a set S one can check in $O(s^2 n^2)$ time whether S admits a consistent labeling via a simple breadth-first search on each connected component.

In case of the edge-deletion problems we could simply contract edges determined to be useless. Since NODE UNIQUE LABEL COVER is a vertex-deletion type of problem, when we decide that a vertex is not going to be a part of a solution, then we *bypass* this vertex. When bypassing a vertex v, we remove it from the graph and for each pair of neighbors $u_1, u_2 \in N_G(v)$ the constraint $\psi_{u_1 u_2}$ needs to be restricted to simultaneously satisfy also $\psi_{vu_2,v} \circ \psi_{vu_1,u_1}$. For this reason, we need to consider a slightly more general version of the NODE UNIQUE LABEL COVER problem, where edge constraints are partial permutations and each vertex has a list of available labels $\phi_v \subseteq \Sigma$. The additional *vertex constraints*, as we call lists ϕ_v, are due to the fact that a solution might remove all but one vertex of $N(v)$, and then even the restricted edge constraints between vertices of $N(v)$ are not enough.

A. Good cuts in the node-deletion variant

We now adjust the notion of good separations also to the node-deletion problems.

Definition III.1. Let G be a connected graph. A triple (Z, V_1, V_2) of subsets of $V(G)$ is called a (q,k)-*good node separation*, if $|Z| \leq k$, V_1 and V_2 are vertex sets of two different connected components of $G \setminus Z$ and $|V_1|, |V_2| > q$.

Unfortunately, we need not only the aforementioned natural extension of the good separation, but also we need to capture a situation where a large number of small components is attached to a common, small interface. This leads us the definition of a (q,k)-flower cut; we omit its technical formal definition in this extended abstract. Similarly to edge-deletion variant, one can find both types of separations in time $O(2^{O(\min(q,k)\log(q+k))} n^3 \log n)$ in the graph, or conclude that no such exists. If a graph does not admit a (q,k)-good node separation or a (q,k)-flower separation, we can prove a similar result to Lemma II.3 that encapsulates the high-connectivity behavior in G.

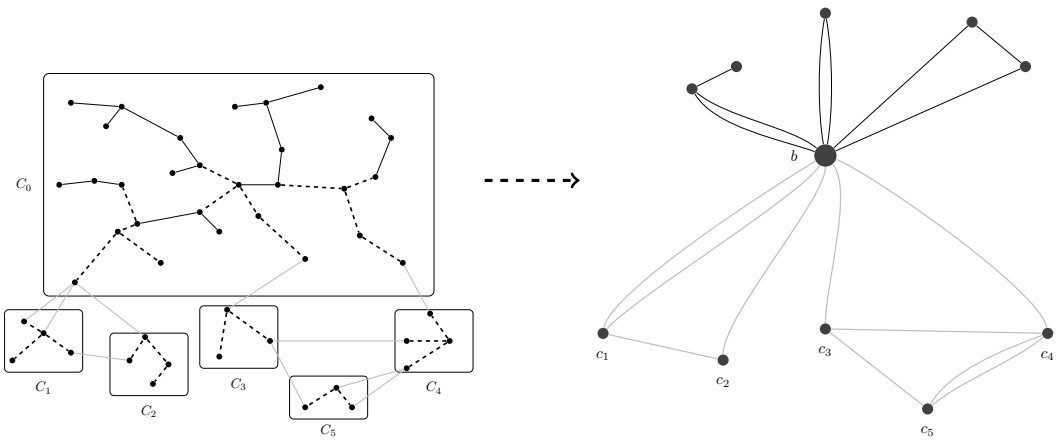

Figure 1. Obtaining the graph H from the graph G. Light grey edges are the edges of X that are preserved, while dashed edges belong to $A_1 \cup A_2$ and are contracted. Note that trees T_0^u inside T_0 are not necessarily disjoint, but to make the presentation clearer they are disjoint in the figure. Moreover, on the left side of the picture we have drawn only edges belonging to X and trees T_i. However, there may be many more edges that, in particular, can influence the shape of the graph H after contraction and identification.

Lemma III.2. *If a connected graph G with 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 $Z \subseteq V(G)$ of size at most k, the graph $G \setminus Z$ contains at most $(2q+2)(2^k-1)+|T_b|+1$ connected components, out of which at most one has more than q vertices.*

Note that the number of connected components could not be bounded if we only used the first type of separations. This problem is the sole purpose of introducing flower separations as well. Fortunately, they admit enough structure to make the recursive understanding step still work.

B. Border problem and recursive understanding

The definition of the border problem is very natural. In the border problem the graph is also equipped with a set T_b of at most $4k$ border terminals. The behavior on the border terminals is expressed by a function $\mathcal{P} : T_b \to \Sigma \cup \{\text{☠}\}$, which simply fixes the values of the labeling. We say that a solution (X, Ψ) is *consistent with* $\mathcal{P}$ if $X \cap T_b = \mathcal{P}^{-1}(\text{☠})$ and $\Psi|_{T_b \setminus X} = \mathcal{P}|_{T_b \setminus X}$.

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 $\texttt{sol}_\mathcal{P} = (X_\mathcal{P}, \Psi_\mathcal{P})$ to the instance $\mathcal{I}$ that is consistent with $\mathcal{P}$ and $|X_\mathcal{P}|$ is minimum possible, or outputs $\texttt{sol}_\mathcal{P} = \perp$ if such a pair does not exist.

This definition allows a standard recursive understanding routine. Take $q = k(s+1)^{4k} + 2k$ and suppose we are given a $(q, 2k)$-good node separation (Z, V_1, V_2) of G. At least one set V_1, V_2 has at most $2k$ terminals, let it be V_1. We recursively apply the algorithm to the instance trimmed to the graph induced by the set $N[V_1] \subseteq V_1 \cup Z$, where we also incorporate $N(V_1) \subseteq Z$ to the border terminals. Let us mark all the border terminals and all the vertices that appeared in any output solution. One can easily show that all the unmarked vertices can be safely bypassed, as their usage can be always replaced with using some marked ones. As we chose q to be large enough, we bypass at least one vertex and we obtain a recursive equation that gives the claimed complexity bound. Given a (q,k)-flower separation with respect to T_b we can make a recursive understanding step in the same manner. Once neither a $(q,2k)$-good node separation nor a (q,k)-flower separation with respect to T_b is present, the graph admits the structure expressed by Lemma III.2 and we may proceed to the high connectivity phase. Note that we also excluded good node separations with larger cutsets, as it will be useful in the future.

C. High connectivity phase

Recall that we are to compute the optimal solution for every behavior $\mathcal{P}$ on border terminals. We iterate through all possible $\mathcal{P}$ and perform computation for each separately; note that this gives $2^{O(k \log s)}$ overhead to the running time. From now on we can assume that $\mathcal{P}$ is fixed.

Let us examine the structure of the instance after removing an optimal deletion set X, given by Lemma III.2. We have at most $d = (q+1)(2^k-1) + 4k$ small components, each containing at most q vertices, and one component of unbounded size. Note that we can assume that this big component, denote its vertex set by $\texttt{big}(X)$, contains more than q vertices, as otherwise the whole graph has size at most $q(d+1)$ and a brute-force search runs within the claimed complexity bound. As feasibility of an empty deletion set

can be checked in polynomial time, we assume that X is nonempty. Let Ψ be a labeling witnessing that X is a feasible deletion set.

We now would like to use a similar approach to the one presented in the illustration, that is, to condensate the high-connectivity behavior in G. Let us take A_1 to be the union of vertex sets of small connected components in $G \setminus X$; note that $|A_1| \leq qd$. For every $u \in X \cap N(\mathtt{big}(X))$ we construct a set $A^u \subseteq \mathtt{big}(X)$ such that $|A^u| = q+1$, $G[A^u]$ is connected and A^u contains a neighbor of u; note that this is possible due to $|\mathtt{big}(X)| > q$. Let us now take $A_2 = \bigcup_{u \in X \cap N(\mathtt{big}(X))} A^u$; note that $|A_2| \leq k(q+1)$. Therefore, if using Lemma I.1 we generate the random family $\mathcal{F}$ for universe $U = V(G)$ and constants $a = qd + k(q+1)$, $b = k$, then we know that there is a set $S_0 \in \mathcal{F}$ such that $A_1 \cup A_2 \subseteq S_0$ and $X \cap S_0 = \emptyset$. We guess the right S_0 by branching into $|\mathcal{F}|$ subcases, labeled by $S \in \mathcal{F}$. Formally, each branch produces a candidate for an optimal solution and the algorithm picks the smallest possible; we argue that the branch with S_0 guessed correctly produces an optimal solution. From now on, we have fixed a set S and we seek a deletion set X that is disjoint with S, but S contains all the vertex sets of small components after removing X, as well as, for every $u \in X$ adjacent to the big component, a vertex set inducing a connected subgraph of size at least $q+1$ adjacent to u.

We refer to vertex sets of connected components of $G[S]$ as *stains*; note that these sets will induce connected subgraphs even after removing the solution. A stain is *big* if it is of size at least $q+1$, otherwise it is *small*. Note that each big stain has to be entirely contained in $\mathtt{big}(X)$. Moreover, for each stain there must exist a consistent labeling of it, or otherwise we may immediately terminate the branch. We would like to mimic the identification step from the illustration for big stains. Consider two big stains C_1, C_2. As the graph does not admit a $(q, 2k)$-good node separation, by Menger's theorem we can distinguish $2k+1$ paths $P^0, P^1, \ldots, P^{2k}$ from C_1 to C_2 that are internally vertex-disjoint. Having fixed $\Psi|_{C_1}$ (recall that we assume that all the vertices of S do not belong to X), for every path P^i we can find a labeling Ψ^i of C_2 that is Ψ propagated via the path P^i to C_2 assuming that P^i is not hit by the deletion set, or conclude that P^i must be hit. As at most k paths P^i are hit, majority of the paths are not hit. It follows that for these paths $\Psi^i = \Psi|_{C_2}$. Therefore, having fixed a labeling on one big stain, we can also fix the labeling on all the big stains, even if we do not know the deletion set: for each big stain we just compute all the labelings Ψ^i and take the one that appears most of the time. Let $S^{\mathtt{big}}$ be the union of all big stains. We branch into at most s branches, in each fixing a labeling of one vertex from $S^{\mathtt{big}}$, thus fixing the labeling of the whole $S^{\mathtt{big}}$.

By our assumptions on S, we know that $X \cap N_G(\mathtt{big}(X)) \subseteq N_G(S^{\mathtt{big}})$. Intuitively, $S^{\mathtt{big}}$ together with its neighbourhood play the role of the core vertex b, while the components of $G \setminus N_G[S^{\mathtt{big}}]$ can be solved more or less independently. In fact, it still holds that for each such component, let C be its vertex set, either C is disjoint with X and entirely contained in $\mathtt{big}(X)$, or all the small stains in C are separated from each other and from $\mathtt{big}(X)$ by X, while the deletion set X contains $C \setminus S$ as well as the whole $N(C)$. This corresponds to the observation in the edge-deletion variant, that no component of the graph after removing the core has a nontrivial intersection with the deletion set.

We omit the description of the remaining part of the algorithm in this extended abstract, as it consists of a few further simple observations and a technical and involved but quite standard application of the bounded search tree technique.

IV. Conclusions

In this paper we presented a new technique of designing parameterized algorithms for cut problems. The natural next step is to try to find further applications of this framework. On the other hand, the algorithms obtained using our framework run in time complexity $2^{O(k^2 \log k)} \cdot \mathrm{poly}(n)$. Is the dependence on k optimal (for example, under Exponential Time Hypothesis)? Or is it possible to design algorithms with running time $2^{O(k \log k)} \cdot \mathrm{poly}(n)$ or even $2^{O(k)} \cdot \mathrm{poly}(n)$ for at least some of the considered problems?

References

[1] R. Chitnis, M. Cygan, M. Hajiaghayi, M. Pilipczuk, and M. Pilipczuk, "Designing FPT algorithms for cut problems using randomized contractions," *CoRR*, vol. abs/1207.4079, 2012.

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

[3] V. Bafna, P. Berman, and T. Fujito, "A 2-approximation algorithm for the undirected feedback vertex set problem," *SIAM J. Discrete Math.*, vol. 12, no. 3, pp. 289–297, 1999.

[4] C. Chekuri, S. Guha, and J. Naor, "The Steiner k-Cut Problem," *SIAM J. Discrete Math.*, vol. 20, no. 1, pp. 261–271, 2006.

[5] G. Even, J. Naor, B. Schieber, and M. Sudan, "Approximating minimum feedback sets and multi-cuts in directed graphs," in *Proc. of IPCO'95*, 1995, pp. 14–28.

[6] G. Even, J. S. Naor, and L. Zosin, "An 8-approximation algorithm for the subset feedback vertex set problem," in *Proc. of FOCS'99*, 1996, pp. 310–319.

[7] N. Garg, V. V. Vazirani, and M. Yannakakis, "Multiway cuts in directed and node weighted graphs," in *Proc. of ICALP'94*, 1994, pp. 487–498.

[8] ——, "Approximate max-flow min-(multi)cut theorems and their applications," *SIAM J. Comput.*, vol. 25, no. 2, pp. 235–251, 1996.

[9] D. R. Karger, P. N. Klein, C. Stein, M. Thorup, and N. E. Young, "Rounding algorithms for a geometric embedding of minimum multiway cut," in *Proc. of STOC'99*, 1999, pp. 668–678.

[10] J. Naor and Y. Rabani, "Tree packing and approximating k-cuts," in *Proc. of SODA'01*, 2001, pp. 26–27.

[11] R. Ravi and A. S. II, "Approximating k-cuts via network strength," in *Proc. of SODA'02*, 2002, pp. 621–622.

[12] P. D. Seymour, "Packing directed circuits fractionally," *Combinatorica*, vol. 15, no. 2, pp. 281–288, 1995.

[13] R. G. Downey and M. R. Fellows, *Parameterized Complexity*. Springer, 1999. [Online]. Available: citeseer.ist.psu.edu/downey98parameterized.html

[14] J. Flum and M. Grohe, *Parameterized Complexity Theory*, 1st ed., ser. Texts in Theoretical Computer Science. An EATCS Series. Springer, March 2006. [Online]. Available: http://www.worldcat.org/isbn/3540299521

[15] D. Marx, "Parameterized graph separation problems," *Theor. Comput. Sci.*, vol. 351, no. 3, pp. 394–406, 2006.

[16] J. Chen, Y. Liu, and S. Lu, "An improved parameterized algorithm for the minimum node multiway cut problem," *Algorithmica*, vol. 55, no. 1, pp. 1–13, 2009.

[17] J. Chen, Y. Liu, S. Lu, B. O'Sullivan, and I. Razgon, "A fixed-parameter algorithm for the directed feedback vertex set problem," in *Proc. of STOC'08*, 2008, pp. 177–186.

[18] R. H. Chitnis, M. Hajiaghayi, and D. Marx, "Fixed-parameter tractability of directed multiway cut parameterized by the size of the cutset," in *Proc. of SODA'12*, 2012, pp. 1713–1725.

[19] M. Cygan, M. Pilipczuk, M. Pilipczuk, and J. O. Wojtaszczyk, "Subset feedback vertex set is fixed-parameter tractable," in *Proc. of ICALP'11*, 2011, pp. 449–461.

[20] S. Guillemot, "FPT algorithms for path-transversal and cycle-transversal problems," *Discrete Optimization*, vol. 8, no. 1, pp. 61–71, 2011.

[21] D. Lokshtanov and D. Marx, "Clustering with local restrictions," in *Proc. of ICALP'11*, 2011, pp. 785–797.

[22] D. Marx and I. Razgon, "Fixed-parameter tractability of multicut parameterized by the size of the cutset," in *Proc. of STOC'11*, 2011, pp. 469–478.

[23] I. Razgon and B. O'Sullivan, "Almost 2-SAT is fixed-parameter tractable," *J. Comput. Syst. Sci.*, vol. 75, no. 8, pp. 435–450, 2009.

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

[25] S. Kratsch, M. Pilipczuk, M. Pilipczuk, and M. Wahlström, "Fixed-parameter tractability of multicut in directed acyclic graphs," *CoRR*, vol. abs/1202.5749, 2012.

[26] D. Marx, B. O'Sullivan, and I. Razgon, "Finding small separators in linear time via treewidth reduction," *CoRR*, vol. abs/1110.4765, 2012.

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

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

[29] K. Kawarabayashi and M. Thorup, "The minimum k-way cut of bounded size is fixed-parameter tractable," in *Proc. of FOCS'11*, 2011, pp. 160–169.

[30] N. Alon, R. Yuster, and U. Zwick, "Color-coding: a new method for finding simple paths, cycles and other small subgraphs within large graphs," in *Proc. of STOC'94*, 1994, pp. 326–335.

[31] M. Naor, L. J. Schulman, and A. Srinivasan, "Splitters and near-optimal derandomization," in *Proc. of FOCS'95*, 1995, pp. 182–191.

[32] S. Guillemot, "Parameterized complexity and approximability of the longest compatible sequence problem," *Discrete Optimization*, vol. 8, no. 1, pp. 50–60, 2011.

[33] S. Khot, "On the power of unique 2-prover 1-round games," in *Proc. of STOC'02*, 2002, pp. 767–775.

[34] S. Arora, B. Barak, and D. Steurer, "Subexponential algorithms for unique games and related problems," in *Proc. of FOCS'10*, 2010, pp. 563–572.

[35] S. Khot, "On the unique games conjecture (invited survey)," in *IEEE Conference on Computational Complexity*, 2010, pp. 99–121.

[36] M. Charikar, K. Makarychev, and Y. Makarychev, "Near-optimal algorithms for unique games," in *Proc. of STOC'06*, 2006, pp. 205–214.

[37] M. Cygan, M. Pilipczuk, and M. Pilipczuk, "On group feedback vertex set parameterized by the size of the cutset," *CoRR*, vol. abs/1112.6255, 2011.

[38] I. Razgon, "Large isolating cuts shrink the multiway cut," *CoRR*, vol. abs/1104.5361, 2011.

[39] H. Nagamochi and T. Ibaraki, "A Linear-Time Algorithm for Finding a Sparse k-Connected Spanning Subgraph of a k-Connected Graph," *Algorithmica*, vol. 7, no. 5&6, pp. 583–596, 1992.

[40] D. R. Karger, "Minimum cuts in near-linear time," *J. ACM*, vol. 47, no. 1, pp. 46–76, 2000.