Multicut is FPT[*]

Nicolas Bousquet [†1], Jean Daligault[1], and Stéphan Thomassé [‡2]

[1]LIRMM (Université Montpellier 2, CNRS), 161 Rue Ada, 34392 Montpellier, France
[2]LIP (ENS Lyon, Université Lyon, CNRS, INRIA, UCBL), 46 Allée d'Italie, 69364 Lyon Cedex 07, France

Abstract

Let $G = (V, E)$ be a graph on n vertices and R be a set of pairs of vertices in V called *requests*. A *multicut* is a subset F of E such that every request xy of R is separator by F, *i.e.*every xy-path of G intersects F. We show that there exists an $O(f(k)n^c)$ algorithm which decides if there exists a multicut of size at most k. In other words, the MULTICUT problem parameterized by the solution size k is Fixed-Parameter Tractable.

1 Introduction

The study of separators and flows is one of the most active field in combinatorial optimization. However, if the simplest case involving one source and one sink is algorithmically tractable, the problem becomes hard as soon as one deals with multiple terminals. For instance, given two requests (x_1, y_1) and (x_2, y_2) in a directed graph D, it is NP-complete to decide if there exist two disjoint directed paths, respectively from x_1 to y_1 and from x_2 to y_2 [18]. In fact, even deciding if two given vertices belong to a directed circuit is already hard. The picture changes when considering undirected graphs, in which case the result of Robertson and Seymour [32] asserts that given k requests $(x_1, y_1), (x_2, y_2), \dots, (x_k, y_k)$, one can decide in cubic time if there exists k disjoint paths connecting all pairs x_i, y_i. The catch is of course that cubic time refers to the instance size, which we generally denote by n. From their work, the complexity of the k-path problem is $\mathcal{O}(f(k) \cdot n^3)$, where f is by no mean polynomial, since the question is NP-complete when k is part of the input.

This result received considerable attention, both since this is the key tool for computing a given minor in a graph, but also because it has opened a breach in the classical NP-complete/P duality. Indeed, the difficulty of the k-path problem does not depend on the size of the instance, but rather on the number of paths we are looking for. In other words, the parameter containing the hardness of the problem is the number k of paths. In a more general way, a problem is *fixed parameter tractable* (FPT) with respect to the parameter k (e.g. solution size, treewidth ...) if for any instance of size n it can be solved in time $O(f(k)n^c)$ for some fixed c. The reader is invited to refer to now classical books by Downey and Fellows [14], Flum and Grohe [16] and Niedermeier [31].

The dual problem of finding disjoint paths from a source s to a sink t is the separator problem where one asks for a set of vertices or edges which deletion separates s from t. Menger's theorem, or more generally LP-duality, asserts that the maximum number of disjoint st-paths is equal to the minimum size of a separator. This property no longer holds when considering multiple requests,

[*]An extended abstract of this paper was published in STOC'11 [2].
[†]bousquet@lirmm.fr
[‡]stephan.thomasse@ens-lyon.fr

where the maximum number of disjoint paths connecting requests is only an obvious lower bound for the size of a multicut, *i.e.*a set of vertices or edges which deletion separates x_i from y_i for every request x_iy_i. Formally, we consider the following problem:

> MULTICUT:
> **Input**: A graph G, a set of requests R, an integer k.
> **Parameter**: k.
> **Output**: TRUE if there is a multicut of size at most k, otherwise FALSE.

MULTICUT and its variants have raised an extensive literature. These problems play an important role in network issues, such as routing and telecommunications (see [10]). For example, vertices of the graph could represent Urban Switch Centers in a telephone network, and (weighted) edges represent physical connections between vertices [4].

The MULTICUT problem is already hard when the input graph is a tree since VERTEX COVER can be viewed as MULTICUT in stars. Indeed consider a star on n branches where each branch corresponds to a vertex of the graph. There is a request between two branches if the corresponding vertices are adjacent in the graph. One can easily verify that a multicut of size at most k is a vertex cover of the original graph of size at most k (and conversely). Hence MULTICUT IN TREES is NP-complete and MaxSNP hard (since VERTEX COVER is), which implies that it admits no Polynomial-Time Approximation Scheme (PTAS) unless P=NP. Garg *et al.* [19] proved that MULTICUT IN TREES admits a 2-approximation algorithm, by showing that in trees the minimal size of a multicut is at most twice the maximal flow value, and using a primal-dual approach. Guo and Niedermeier [23] proved that MULTICUT IN TREES is FPT parameterized by the size of the solution. The existence of a polynomial kernel was asked in [1]. Bousquet, Daligault, Thomassé and Yeo proved in [3] that MULTICUT IN TREES admits a $\mathcal{O}(k^6)$ kernel. This upper bound was improved by Chen *et al.* in [6] into a $\mathcal{O}(k^3)$ kernel.

A classical variant of MULTICUT is the MULTIWAY CUT problem in which a set of terminals has to be pairwise separated.

> MULTIWAY CUT:
> **Input**: A graph $G = (V, E)$, a set of terminals $X \subseteq V$, an integer k.
> **Parameter**: k.
> **Output**: TRUE if there exist k edges whose deletion puts every vertex of X in distinct connected components, otherwise FALSE.

MULTIWAY CUT has been proved to be FPT parameterized by the size of the solution by Marx [26]. A faster $O^*(4^k)$ algorithm was proposed by Chen *et al.* [7], improved into a $\mathcal{O}^*(2^k)$ algorithm in [12]. The proof, based on important separators, will be detailed a little bit further. Recently Kratsch and Wahlström proved in [25] that MULTIWAY CUT admits a $\mathcal{O}(k^t)$ kernel where t denotes the number of terminals. It is still open to determine if MULTIWAY CUT admits a polynomial kernel parameterized by the size of the cutset only.

On general instances, Garg *et al.* gave an approximation algorithm for MULTICUT within a logarithmic factor in [17], proving that the minimum size of the multicut is within a factor $\mathcal{O}(log(\ell))$ of the maximum multiflow value in general graphs, where ℓ is the number of requests. However, MULTICUT has no constant factor approximation algorithm if Khot's Unique Games Conjecture holds [5].

Guo *et al.* showed in [22] that MULTICUT is FPT when parameterized by both the treewidth of the graph and the number of requests. Gottlob and Lee proved a stronger result in [20]: MULTICUT is FPT when parameterized by the treewidth of the input structure, *i.e.*the input graph whose edge set is augmented by the set of requests. Marx proved that MULTICUT is FPT parameterized by the size of the solution plus the number of requests [26]. A faster algorithm running in time $\mathcal{O}^*((8 \cdot \ell)^k)$ was given by Guillemot [21] (recall that ℓ is the number of requests). Marx *et al.* [27] obtained FPT results for more general types of constrained MULTICUT problems through treewidth reduction results. However their treewidth reduction techniques do not yield FPT algorithm of MULTICUT when parameterized by the size of the solution only. Finally, Marx and Razgon obtained a factor 2 Fixed-Parameter-Approximation for Multicut parameterized by the size of the solution in [29]. In this paper we prove that Multicut is FPT parameterized by the size of the solution. Independently, Marx and Razgon also provided an FPT algorithm with a rather different approach [30]. Nevertheless there remain several open problems on Multicut. In particular we can add constraints on the structure of the target Multicut: we can look for independent multicuts, or connected multicuts. When parameterized by both the size of a solution and the number of requests, the problem is FPT for finding independent multicuts [28]. Recently, Cygan *et al.* proved that both Multicut and Multiway Cut parameterized by the size of the solution do not admit polynomial kernels [11].

Problem 1. Does Multicut parameterized by both the size of the solution and the number of requests admits a polynomial kernel?

Recently, lots of research have been done for studying Multicut in directed graphs. The input graph is a directed graph and the requests are directed pairs. The objective is to eliminate all the directed paths between directed pairs of terminals. Marx and Razgon proved that Directed Multicut is W[1]-hard parameterized by the size of the solution in [30]. Even worse, Kratsch *et al.* proved that Directed Multicut is $W[1]$-hard parameterized by the size of the solution in directed acyclic graphs [24]. Nevertheless Directed Multicut in Directed Acyclic Graphs is FPT parameterized by the size of the solution plus the number of pairs of terminals [24]. In addition, Chitnis *et al.* proved in [9] that Directed Multicut with two pairs of terminals is FPT parameterized by the size of the solution. A crucial variant of Directed Multicut is FPT parameterized by the size of the solution in directed graphs: Subdivided Multicut. In the Subdivided Multicut problem, we are given an oriented graph and two ordered sets of terminals S, T and we want to find a subset of at most k edges which eliminates all the directed paths from s_i to t_j for every $j \geq i$. Subdivided Multicut is FPT parameterized by the size of the solution in Directed Acyclic Graphs [8]. The proof of this result is the core of the proof that Directed Feedback Vertex Set is FPT parameterized by the size of the solution [8]. More recently and using shadow removal techniques, several other variants of Directed Multicut such as Directed Multiway Cut were shown to be FPT [9].

The remaining of this paper is devoted to proving that Multicut is FPT.

Acknowledgments. This proof was a long process in which several researchers gave us advice or participated in some of the partial results. One of our important tools, Lemma 15, is the crucial result of the reduction of Multicut to bounded treewidth, and was obtained in collaboration with Christophe Paul and Anthony Perez [13]. The fact that a vertex cutset Y could be obtained by iterative compression, a tool of [29], was brought to our attention by Sylvain Guillemot. Finally, we would like to thank Valentin Garnero who worked on this project as a part of his graduate research experience in June 2010.

2 Preliminaries

A vertex which sends a request is called a *terminal*. The number of requests sent by a terminal is its *request degree*.

2.1 Detailed outline of the proof

Some parts of the proof are technically very involved. This section provides a detailed outline of the proof, underlying the structure of the main results and the reasons behind the main definitions. For formal definitions and statements, and for complete proofs, the reader is referred to the following sections.

A Vertex-Multicut. First of all, we can assume by iterative compression that a vertex-multicut Y of size $k+1$ is given, and that a solution must split Y. In other words, the solution has to be a Multiway Cut of Y. This is expressed in Lemma 21. This vertex-multicut Y gives a first layer of structure to an instance: we can focus on the Y-components, *i.e.* the connected components of the graph where vertices of Y have been removed.

Setting the number of edges of the solution per component. The number of Y-components is bounded in k, considering that all connected components of $G \setminus Y$ which are adjacent to a single given vertex $y \in Y$ form a single Y-component. So, we can branch to decide how many edges of the solution lie in each Y-component.

Half-requests. No request is contained inside a Y-component with two or more attachment vertices, so we can simulate a request (u, v) with several half-requests (u, y, v), where $y \in Y$ is an attachment vertex of both the component $C(u)$ of u and of the component $C(v)$ of v. Cutting a half-request (u, y, v) means cutting all paths between u and v which go through y.

The goal: 2-SAT. Half-requests give a simpler structure to the multicut problem: cutting a half-request (u, y, v) is equivalent to either separating u from y in $C(u)$, or separating v from y in $C(v)$. We will express this "or" through 2-SAT clauses once we manage to express in a simple way whether the solution separates u from y. The remaining of the proof is devoted to simplifying the structure of the instance until we can express with 2-SAT variables whether the solution separates u from y.

Focus on 2-components. We first reduce Y-components with three or more attachment vertices in Lemma 24. Now, Y-components have either two attachment vertices (2-components) or one attachment vertex (cherries). The complexity of the problem mostly lies in the existence of 2-components.

In order to give a better structure to 2-components we compute in Lemma 26 a particular path between its two attachment vertices: the backbone, which has the following property. The multicut must contain exactly one edge in the backbone. The set of multicuts is thus linearly partially ordered, according to the edge of the backbone they use. The goal is now to simplify the structure of the instance so that the multicuts that separate a vertex u from an attachment vertex y of $C(u)$ form an initial (or final) section of this linear order. Indeed, the fact that the solution belongs to an initial or final section of a linear order can be easily expressed with a 2-SAT variable.

Backbone Multicut. The instance as reduced up to this point fits the first intermediate variant, Component Multicut, defined in Section 4. We introduce in Section 5 the second intermediate problem Backbone Multicut. We need this more general problem than Component Multicut, so that we can enrich instances. Considering half-requests is of major importance in our proof, but the presence of 2-SAT clauses is necessary only for Lemma 32, and could possibly be avoided with a slightly different proof.

Lemonizing 2-components. Through Lemma 32 in Subsection 5.5 we reduce 2-components so that each vertex of the backbone becomes a cut-vertex of the 2-components. An example is drawn in Figure 7. This is the first highly technical part of the proof. The components now consist in a sequence of lemons with cherries attached to the backbone.

Linearly ordering the set of multicuts. In Subsection 5.6 we perform a complete linearisation of the set of multicuts. This can be seen as the core of our approach. We define a meaningful partial order $\preceq$ on multicuts, such that the set of multicuts can be partitioned into a number bounded in k of parts totally ordered by $\preceq$ by Dilworth's Theorem. This is Lemma 34.

Reducing 2-components to a backbone. The linear order $\preceq$ on the set of all multicuts allows us to move terminals of cherries to the backbone in Lemma 35. In a component with no cherries left, we manage to reduce the sequence of lemons to just the backbone in Theorem 37. This is the second highly technical part of the proof.

Reduction to 2-SAT. At this point, the 2-components are just paths. Only cherries attached to vertices of Y remain, and they have a bounded number of meaningful separators, thanks to the important separators technique. With an instance consisting of a subdivision of a graph with at most k edges, we easily express with 2-SAT variables in Theorem 39 whether a given vertex is separated from a given vertex of Y. We end up (after a heavy dose of branching throughout the proof) with a 2-SAT instance, which is polynomially solvable.

Complexity and Programmability. The overall algorithm is single exponential. It should be quite difficult to effectively implement the algorithm though, the whole proof being very involved. Finding a simpler proof leading to a simpler algorithm would be very interesting.

2.2 Reductions, branchings and invariants

Let us now turn to the proof of the fixed-parameter tractability of the general Multicut problem. In this chapter, we study Multicut variants with additional constraints on the deleted edges. In the original Multicut problem, we can delete a set of k edges without restrictions, but in some more constrained versions we must delete a prescribed number of edges on some particular paths. The total number of deleted edges is called *deletion allowance* of the multicut problem. We will make extensive use of the term *bounded* which always implicitly means bounded in terms of the deletion allowance. Also, when speaking of FPT time, we always mean $\mathcal{O}(f(d)n^c)$ where c is a fixed constant and d is the deletion allowance. In the algorithm we will perform reductions and branchings, and we use invariants to bound the total running time.

Reductions. These are computations where the output is a reduced instance which is equivalent to the original instance with respect to the existence of a solution. One of the most natural reductions concerns irrelevant requests, *i.e.* a request xy such that every k-multicut of $R \setminus xy$ actually cuts x from y, where R is the set of requests. If one can certify that a request xy is irrelevant, the reduction consists in replacing R by $R \setminus xy$. Another reduction is obtained if we can certify that, if there exists a k-multicut, then there exists a k-multicut which does not separate two given vertices u and v. In this case, we simply contract u and v. Reductions are easy to control, and we can perform reductions liberally provided that some invariant polynomial in n decreases. For example, request deletions can be performed at most n^2 times, and vertex contractions at most n times.

Branchings. In our algorithm, we often have to decide if the multicut we are looking for is of a particular type, where the number of types is bounded. We will then say that we branch over all the possible cases. This means that, to compute the result of the current instance, we run our algorithm on each case, in which we force the solution to be of each given type. The instance is positive if at least one of the cases is positive. To illustrate this, in the case of a graph G with two connected components G_1 and G_2, both containing requests, we would branch over $k-1$ instances, depending of the number of edges (between 1 and $k-1$) that we remove from G_1. This simple branching explains why we can focus on connected graphs.

Invariants. To prove that the total number of branches is bounded, we show that some invariant is modified at each branching step, and that the number of times that this invariant can be modified is bounded. We usually have several invariants ordered lexicographically. In other words, we have different invariants which we want to increase or decrease and each invariant can take a bounded number of values. These invariants are ordered, there is a primary invariant, a secondary invariant, etc. Each branching must *improve the invariant*, *i.e.* the first invariant (with respect to priority order) which is changed by the branching must be modified according to the preference, increase or decrease, that we specified for it. For instance, the primary invariant could be the number of edges in the multicut, which we want to decrease, and the secondary invariant could be the connectivity of G, which we want to increase. In this case, if we can decrease the number of edges in the solution we do so even if the connectivity of the graph decreases. Also, if a branching increases connectivity and leaves the number of multicut edges unchanged, we improve the invariant.

3 Cherries

3.1 Important separators

Let us first introduce the notion of important separators, first used by Marx in [26]. All the results of this section are classical ones. We nevertheless recall them for the sake of completeness and for giving some intuitions on the important separators technique which is crucial our proof. Many graph separation problems solved in the last few years use important separators, such as [2, 8, 9, 26, 30]. Let G be a (integer weighted) graph and x, y be two vertices of a graph. The

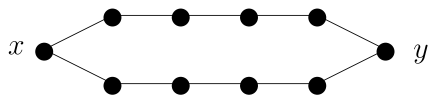

Figure 1: Every pair of edges with one edge in the upper path and one edge in the below path is a minimum xy-separator.

number of minimum separators between x and y can be arbitrarily large. Consider for instance the graph of Figure 1. Every pair of edges with one edge on the upper path and one on the below path is a minimum separator between x and y. So, if the two xy-paths have length n, then there are 2^n xy-separators of size 2. Nevertheless, all of them are not necessarily "important". Imagine for instance that our goal is to separate x from y in such a way the component of x is minimized. Then there is a unique such xy-separator of minimum size (instead of an exponential number). In Figure 1, it is the set of two edges adjacent to x. Marx proved that the number of (indivisible) important xy-separators of size at most k is at most 4^k. In addition, all these separators can be found in FPT time parameterized by k.

Let $G = (V, E)$ be a connected graph on n vertices with a particular vertex x called the *root*. In the following, we will only consider separators from the point of view of the root x. A separator can be considered as a set of edges, but also as a bipartition of the vertex set (the vertices which are in the connected component of x and the vertices which are not). In the following we consider separators as bipartitions. As we want to focus on one side of the bipartition, we define a *separator* as a subset of vertices S containing the root x. The *border* of S is the set of edges of G with exactly one endpoint in S. We denote the border of S by $\Delta(S)$, and the cardinality of $\Delta(S)$ by $\delta(S)$. By abuse of notations, the *size of a separator* S is the size of the border of S, *i.e.* $\delta(S)$. Recall that $\overline{S}$ denotes the complement of the set S, *i.e.* $V \setminus A$. Δ

Observation 2. *The function δ is submodular,* i.e.*for every pair of separators S, T we have $\delta(S) + \delta(T) \geq \delta(S \cap T) + \delta(S \cup T)$.*

Proof. Let us prove that every edge which contributes to the right hand side also contributes to the left hand side with at least the same multiplicity. Let S and T be two separators and let u, v be two vertices of the graph.

- If uv is in $\Delta(S \cap T)$ and in $\Delta(S \cup T)$, then without loss of generality, u is in $S \cap T$ and v in $\overline{S} \cap \overline{T}$. So uv is in both $\Delta(S)$ and $\Delta(T)$.
- If uv is in $\Delta(S \cap T)$ and not in $\Delta(S \cup T)$, then, up to symmetry, we have $u \in S \cap T$ and $v \in S \cap \overline{T}$. So uv is in $\Delta(T)$.
- If uv is in $\Delta(S \cup T)$ and not in $\Delta(S \cap T)$ then, without loss of generality, u is in $S \setminus (S \cap T)$ and v is in $\overline{S} \cap \overline{T}$. So in particular uv is in $\Delta(S)$.

□

Let x be a root. Let y be a vertex. Menger's Theorem ensures that the size of a minimum xy-separator can be determined in polynomial time (and a minimum separator can be computed in polynomial time). It is equal to the maximum number of edge-disjoint paths between x and y. The *connectivity* between x and y is the minimum size of a separator between x and y and it is denoted by $\lambda(x, y)$ (or when no confusion is possible by λ). A separator S is an *important separator* if every separator T such that $T \subsetneq S$ satisfies $\delta(T) > \delta(S)$. Let us first make an easy observation.

Observation 3. *Every separator S contains an important separator S' such that $\delta(S') \leq \delta(S)$.*

Proof. We prove it by induction on the number of vertices contained in S. Either S is an important separator and the conclusion holds. Otherwise there exists $S' \subsetneq S$ such that $\delta(S') \leq$

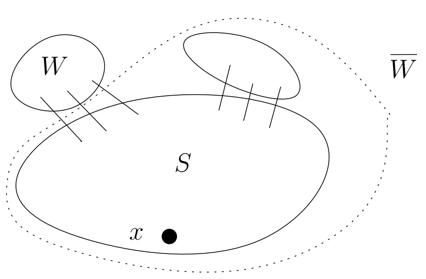

Figure 2: Illustration of Lemma 6.

$\delta(S)$. By induction there exists an important separator S'' such that $S'' \subset S'$. We also have $S'' \subseteq S$. ☐

The "idea" of important separators can be summarized as follows. If you want to find a separator with less vertices (in the component of the root x), then we have to pay a price: the size of the border of the separator must be strictly larger. Before stating the most important results of this section, let us first give some general properties of important separators.

Lemma 4. *Important separators are closed under union. In other words, if S and T are important separators, then $S \cup T$ is an important separator.*

Proof. Let $S_1 \cup S_2$ be the union of two important separators. Let $S_3 \subsetneq S_1 \cup S_2$ be a separator such that $\delta(S_3)$ is minimized. Our goal is to prove that $\delta(S_3) > \delta(S_1 \cup S_2)$. Indeed in this case, since $\delta(S_3)$ minimizes the size of the border over the strict subsets of $S_1 \cup S_2$, it would mean that no $T \subsetneq S_1 \cup S_2$ satisfies $\delta(T) \leq \delta(S_1 \cup S_2)$, *i.e.* $S_1 \cup S_2$ is an important separator.
Without loss of generality, we can assume that S_1 is not included in S_3. Since S_1 is an important separator, we have $\delta(S_1 \cap S_3) > \delta(S_1)$. Indeed $S_1 \cap S_3 \subsetneq S_1$ and every strict subset T of S_1 satisfies $\delta(T) > \delta(S_1)$ since S_1 is an important separator. As Observation 2 ensures that $\delta(S_1 \cap S_3) + \delta(S_1 \cup S_3) \leq \delta(S_1) + \delta(S_3)$, we obtain $\delta(S_1 \cup S_3) < \delta(S_3)$.
Note that we have $S_1 \cup S_3 \subseteq S_1 \cup S_2$ (since both S_1 and S_3 are in this set). By definition, S_3 has minimum border among strict subsets of $S_1 \cup S_2$ and $\delta(S_3) > \delta(S_1 \cup S_2)$, hence the set $S_1 \cup S_3$ is not a strict subset of $S_1 \cup S_2$. So it is equal to $S_1 \cup S_2$. Finally $\delta(S_1 \cup S_2) = \delta(S_1 \cup S_3) < \delta(S_3)$, thus $S_1 \cup S_2$ is an important separator. ☐

Lemma 5. *If S_1, S_2 are distinct important separators, then $\delta(S_1 \cup S_2) < \max(\delta(S_1), \delta(S_2))$.*

Proof. As $S_1 \neq S_1 \cup S_2$ or $S_2 \neq S_1 \cup S_2$, we can assume without loss of generality that $S_1 \subsetneq S_1 \cup S_2$. By Lemma 4, $S_1 \cup S_2$ is an important separator and $S_1 \subsetneq S_1 \cup S_2$, thus we have $\delta(S_1 \cup S_2) < \delta(S_1) \leq \max(\delta(S_1), \delta(S_2))$. ☐

A separator S is an *indivisible separator* if no strict subset of $\Delta(S)$ is a separator. In other words, when we delete from the graph all the edges of $\Delta(S)$ but exactly one, the graph becomes a connected graph. Or yet differently an indivisible separator is a separator S such that $V \setminus S$ induces a connected subgraph. A separator is *divisible* if it is not indivisible. An *indivisible important xy-separator* S is an indivisible important separator rooted in x such that $y \notin S$.

Lemma 6. *If S is a divisible important separator and W is a connected component of $G \setminus S$, the separator $\overline{W}$ is an indivisible important separator with $\delta(\overline{W}) < \delta(S)$.*

Proof. The situation of Lemma 6 is illustrated on Figure 2. The separator $\overline{W}$ is indivisible since the set W induces a connected graph. So we just have to prove that $\overline{W}$ is an important separator. First note that we have $\Delta(\overline{W}) \subsetneq \Delta(S)$. Note that the non equality comes from the fact that S is divisible.

Consider an important separator $T \subseteq \overline{W}$ which minimizes $\delta(T)$. Our goal is to prove that $T = \overline{W}$. By Lemma 4, $S \cup T$ is an important separator. As $\delta(T) \leq \delta(S \cup T)$ by minimality of $\delta(T)$ (since $S \cup T \subseteq \overline{W}$), we have $T = S \cup T$ (since $S \cup T$ is an important separator). In particular, $S \subseteq T$. Every edge of $\Delta(\overline{W})$ has one endpoint in S and one endpoint in W. Since $S \subseteq T$ and since no vertex of W is in T, we have $\Delta(\overline{W}) \subseteq \Delta(T)$. Therefore, by minimality of $\delta(T)$, we have $\Delta(T) = \Delta(\overline{W})$. Finally we have $T = \overline{W}$. Thus, $\overline{W}$ is an important separator. $\square$

Corollary 7. *Every indivisible separator S contains an indivisible important separator S' with $\delta(S') \leq \delta(S)$.*

Proof. Let S'' be an (non necessarily indivisible) important separator contained in S such that $\delta(S'') \leq \delta(S)$. As $\overline{S}$ is connected, the set $\overline{S}$ is included in a connected component Y of $G \setminus S''$. By Lemma 6, $S' := \overline{Y}$ is an indivisible important separator with $\delta(S') < \delta(S'') \leq \delta(S)$. Moreover, $S' \subseteq S$ as $\overline{S} \subseteq Y$. $\square$

In the following we are looking for xy-separators: we do not care about separating x from other vertices than y, we only care about indivisible xy-separators since if the separator is divisible we can do "the same" with less edges. And Corollary 7 ensures that we can study indivisible important xy-separators. Marx first proved that the number of indivisible important xy-separators of size at most k is bounded by a function of k. In the literature the "indivisible" is often omitted and such indivisible important xy-separators are often called important xy-separators.

Lemma 8. *Let G be a graph and x, y be two vertices. There is a unique important xy-separator of size $\lambda(x, y)$ (which can be computed in polynomial time). This separator is indivisible. In addition, every important xy-separator is contained in S.*

Proof. Let S be an important xy-separator of size $\lambda(x, y)$. Note that such a separator exists by Observation 3. Assume by contradiction that an important xy-separator T contains a vertex which is not in S. Lemma 4 ensures that $S \cup T$ is an important separator. Since $S \subsetneq S \cup T$ and since $S \cup T$ is an important separator, we have $\delta(S \cup T) < \delta(S)$. In addition $y \notin S \cup T$, so there is a xy-separator of size less than $\lambda(x, y)$, a contradiction.
So there is a unique minimum important xy-separator. In addition this separator is indivisible otherwise it would not be minimal. The proof of the complexity part can be found in [26] for instance. $\square$

In the following the unique important xy-separator of size $\lambda(x, y)$ is called *the minimum important xy-separator*. Let $G = (V, E)$ be a graph. The graph $G' = (V', E')$ obtained by *identifying* two vertices u and v is the graph where u, v are deleted and are replaced by a unique vertex w, and w is adjacent to every vertex w' which is a neighbor of u or v and the multiplicity of this edge is the multiplicity of uw' plus the multiplicity of vw'. *Contracting* an edge e consists in identifying the endpoints of e. Before stating the main theorem of this section, let us first make an observation.

Observation 9. Let G be a graph. Let S be the minimum important xy-separator and If G' be the graph in all the vertices of $\overline{S}$ are identified with y. Every indivisible important xy-separator of G is an indivisible important xy-separator of G'.

Proof. Lemma 8 ensures that every important xy-separators are contained in S. Thus all the vertices of $\overline{S}$ can be identified with y. Indeed for every indivisible important xy-separator T, no vertex of $\overline{S}$ is in T and then the border of T is not affected by this contraction. $\square$

Note also that in the graph G' there is a unique xy-separator of size $\lambda(x, y)$ (which is $V \setminus y$). The first upper bound on the number of indivisible important xy-separators is due to Marx. It was improved by Chen *et al.*

Theorem 10 (Marx [26], Chen *et al.* [7])**.** *Let G be a graph and x, y be two vertices. There are at most 4^k indivisible important xy-separators of size at most k which can be enumerated in FPT time (more precisely in $2^{\mathcal{O}(k)}$).*

Proof. First note that if $\lambda(x, y) > k$, then there is no indivisible important xy-separator. So in the following we assume that $k \geq \lambda(x, y)$. By Lemma 8, there exists a unique minimum indivisible important xy-separator S. Identify all the vertices in $\overline{S}$ with y (since no vertex of $\overline{S}$ is in an indivisible important xy-separator by Lemma 8 this operation is safe). We still denote by y the identified vertex. Let us denote by G' the graph obtained after this contraction. Observation 9 ensures that after this contraction:

- There is a unique minimum xy-separator which is the set of edges adjacent to y.
- The indivisible important xy-separators of the graph G are exactly the indivisible xy-separators of the graph G'.

In the remaining of the proof we only work on the graph G'. Let us prove that every indivisible important xy-separator can be found using a branching algorithm. Let e be an edge of $\Delta(S)$.

Claim 11. *Every indivisible important xy-separator of G' containing e in its border is an indivisible important xy-separator of $G' - e$ (*i.e. *the graph obtained by a deletion of e from E).*

Proof. Let T be an indivisible important xy-separator such that $e \in \Delta(T)$. Assume by contradiction that $T \setminus e$ is not the border of an important separator in $G' - e$. Then there exists $T' \subsetneq T$ such that $\delta(T') \leq \delta(T)$ in $G' \setminus e$. In the whole graph G' (*i.e.* when we add e), the border of T' increases by at most one (since only the edge e can be added in the border of T'). And the border of T increases by exactly one. So we still have $\delta(T') \leq \delta(T)$ in G, contradicting the fact that T is an important xy-separator. Moreover $T \setminus e$ is clearly still indivisible in $G' - e$. $\square$

Claim 12. *Every indivisible important xy-separator of G' not containing e in its border is an indivisible important xy-separator of G' where the endpoints of e are contracted.*

Proof. Let us denote by G'_e the graph G' in which the edge e is contracted. Let T be an indivisible important xy-separator of G' such that $e \notin \Delta(T)$. So both endpoints of e are in the same connected component in $G'[E \setminus \Delta(T)]$ (since e is not in $\Delta(T)$). Assume by contradiction that T is not indivisible in G'_e. So there exists $T' \subsetneq T$ such that $\delta(T') \leq \delta(T)$. Though the size of $\delta(T')$ does not increase in the graph G'. So we still have $\delta(T') \leq \delta(T)$ and $T' \subsetneq T$, a contradiction. $\square$ Consider the following algorithm. If $\lambda(x, y) > k$ return the empty set. Otherwise, compute the unique minimum indivisible important xy-separator S (which can be done in polynomial time by Lemma 8). Identify all the vertices of $\overline{S}$ with y. Let e be an edge adjacent to y in the contracted graph. Claim 11 and 12 ensures that if we branch by deleting the edge e or by contracting the edge e, then we find all the indivisible important xy-separators.

Let us prove that in both cases, the invariant $2k - \lambda$ decreases. In the first case, k decreases by one and the connectivity decreases by exactly one. Indeed Menger's theorem ensures that there are $\lambda(x, y)$ edge-disjoint xy-paths in G, so there remain at least $\lambda - 1$ paths when e is deleted, *i.e.* the connectivity decreases by at most one (and actually exactly one). In the second case, k is not modified and λ strictly increases. Indeed since vertices of $\overline{S}$ have been contracted with y and since S is the unique minimum important xy-separator. The contraction of the edge e ensures that the connectivity strictly increases. Otherwise there would be an xy-separator T of size λ with both vertices of e in $\overline{T}$, and then $T \subsetneq S$ and $\delta(T) \leq \delta(S)$, a contradiction. So the invariant decreases. So the height of the branching tree is at most $2k$. And the width of the branching tree is 2 since we branch over two possible choices at each step. So there are at most 4^k branches and each leaf gives at most an indivisible important xy-separator, so there are at most 4^k indivisible important xy-separators of size at most k.

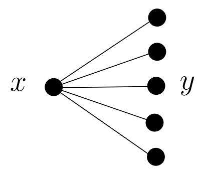

Figure 3: Every pair of edges containing the edge xy is an important xy-separator, but there is a unique indivisible important xy-separator which is the edge xy.

Since computing the unique minimum indivisible important xy-separator can be done in polynomial time, the resulting algorithm is FPT. $\square$

Theorem 10 ensures that the number of indivisible important xy-separators is bounded. We have already seen that the "important" is necessary in Figure 1. But the "indivisible" assumption is also necessary. In Figure 3, any pair of edges containing the edge xy is an important xy-separators while the number of indivisible important xy-separators is one. In addition Theorem 10 is almost tight since there exist graphs with $\mathcal{O}(4^k/poly(k))$ indivisible important YX-separators of size k.

3.2 Irrelevant requests

Using the important separator technique, let us show that we can assume that every vertex of the graph appears in a bounded number of requests. We denote by C^y_k, $C^y_{<k}$ and $C^y_{\leq k}$ the set of indivisible important xy-separators of size respectively exactly k, less than k and at most k. We denote by C_k, $C_{<k}$ and $C_{\leq k}$ respectively the union over all vertices y of G of C^y_k, $C^y_{<k}$ and $C^y_{\leq k}$ respectively.

A collection of sets is called a *Δ-system* if every two distinct sets have the same intersection. Erdős and Rado [15] proved that there exists a function er such that a collection of $er(k,r)$ sets of size at most k contains a Δ-system consisting of r sets. The bound proved in following result will be immediately improved to a single-exponential bound with Theorem 14, whose proof is conceptually more complicated.

Theorem 13. *Every set K of at least $er(4^k, k')$ vertices contains a subset K' of size k' such that every important separator S with $\delta(S) \leq k$ satisfies either $S \cap K' = \emptyset$ or $|K' \setminus S| \leq k$. In other words, every important separator with border at most k isolates either all the elements of K', or at most k elements of K'. The set K' can be computed in FPT time in k and k'.*

Proof. Let us consider the collection $\mathcal{C}$ of sets $C^y_{\leq k}$, for all $y \in K$. By Theorem 10, the collection $\mathcal{C}$ has size bounded in terms of k and k' and can be computed in FPT time.

Consider the following hypergraph H. The vertex set is the set of all the indivisible important separators. A set Y is a hyperedge if there exists a vertex y such that Y is precisely the set of indivisible xy-separators. The sets $C^y_{\leq k}$ have size at most 4^k and the set K has size $er(4^k, k')$, so the hypergraph H admits a Δ-system of size k', *i.e.* a subset K' of k' vertices of K such that for all $y, y' \in K'$, the set $C^{y'}_{\leq k} \cap C^y_{\leq k}$ is equal to some fixed set C of $C_{\leq k}$. This set K' is computable in FPT time. Every indivisible important separator S in C satisfies $S \cap K' = \emptyset$, *i.e.* the separators in C isolate K'. Moreover, if a separator S in $C_{\leq k}$ does not belong to C, then S belongs to at most one $C^y_{\leq k}$ for $y \in K'$. Indeed otherwise y, y' would be in the same connected component, and then S would be both an indivisible important xy-separator and an indivisible important xy'-separator, so by definition of Δ-system it would be a separator for the whose set K'. So finally S isolates at most one vertex of K'.

We have proved so far that the conclusion of Theorem 13 holds if S is an indivisible important separator with border or size at most k, with the stronger conclusion that S isolates at most one vertex of K' when it does not completely separates K'. To conclude, let us observe that if S is divisible and Z is a component of $G \setminus S$, then by Lemma 6 the separator $\overline{Z}$ belongs to $C_{\leq k}$. Hence either $\overline{Z}$ isolates K', or $\overline{Z}$ isolates at most one vertex of K'. The number of components of $G \setminus S$ is at most k, which concludes the proof of Theorem 13. $\square$

The same result holds with a better (single-exponential) bound, as expressed in the following result. We have kept the simpler Theorem 13 along with its proof for readers more concerned with the simplicity of the argument than by the efficiency of the algorithm.

Theorem 14. *Every set K of at least $\alpha(k,k') = k'^{\binom{k+2}{2}-1}$ vertices contains a subset K' of size k' such that every important separator S with $\delta(S) \leq k$ satisfies either $S \cap K' = \emptyset$ or $|K' \setminus S| \leq k$. The set K' can be computed in FPT (single exponential) time.*

Proof. Observe that the result trivially holds when $k' \leq k$. So we can assume that $k' > k$. We prove the result by induction on k.

For $k = 1$, $\alpha(1,k') = k'^2$. The complements of important separators with a border of size 1 form a collection of disjoint sets of vertices, which induces a partition of K. There is either a class K' of this partition containing at least $\sqrt{|K|} \geq k'$ elements, or a set K' of size at least $\sqrt{|K|} \geq k'$ whose elements are chosen in different classes. In both cases K' satisfies the induction hypothesis.

Assume now that $k > 1$. We distinguish two cases. Assume that there exists an indivisible important separator S with $\delta(S) \leq k$ and $|K \setminus S| \geq k'^{\binom{k+1}{2}-1}$. By induction, we extract from $K \setminus S$ a subset K' of size k' such that every important separator T with $\delta(T) \leq k-1$ satisfies either $T \cap K' = \emptyset$ or $|K' \setminus T| \leq k-1$. Consider an important separator S' with $\delta(S') = k$. If $S' = S$, then S' isolates K' by definition of K', hence we assume that S' is distinct from S. Observe that $K' \setminus S'$ is equal to $K' \setminus (S \cup S')$. As $S \cup S'$ is an important separator with border at most $k-1$ (by submodularity we have $\delta(S \cup S') \leq \delta(S) + \delta(S') - \delta(S \cap S')$), the set $K' \setminus S'$ is either K' or has size at most $k-1$. So the conclusion of Theorem 14 holds.

Conversely, assume that all indivisible important separators S with $\delta(S) \leq k$ satisfy $|K \setminus S| < k'^{\binom{k+1}{2}-1}$. Consider an auxiliary graph H with vertex set K and where vv' is an edge when there exists an indivisible important separator S with $\delta(S) \leq k$ such that $\{v,v'\} \cap S = \emptyset$. The degree in H of a vertex v is less than $d := k'^{\binom{k+1}{2}-1} \cdot 4^k$. Indeed, there are at most 4^k indivisible important xv-separators with a border of size at most k, and we have assumed that all indivisible important separators S with $\delta(S) \leq k$ satisfy $|K \setminus S| < k'^{\binom{k+1}{2}-1}$. Note that d is less than $k'^{\binom{k+1}{2}-1} \cdot 4^{k'}$ as $k' \geq k$. There is a stable set K' in H of size at least $\frac{|K|}{d}$, and $\frac{|K|}{d} \geq k'$ since $\binom{k+2}{2} - \binom{k+1}{2} - k = 1$ (note that we evaluate 4^k as k^k for simplicity but a better function would work). Every indivisible important separator S with $\delta(S) \leq k$ isolates at most one vertex of K' as K' is stable in H. Thus, every important separator S with $\delta(S) \leq k$ isolates at most k vertices of K'. The induction hypothesis holds in both cases, which concludes the proof of Theorem 14. $\square$

Theorem 15. *Every set K with at least $h(\ell) := \ell \cdot 4^\ell + 1$ vertices of G contains a vertex y such that every separator S with $\delta(S) + |S \cap K| \leq \ell$ is such that $y \notin S$. Moreover, y can be found in FPT time.*

In other words, the vertex y verifies the following: whenever the deletion of a set of a edges isolates x from all but b elements of K, with $a + b \leq \ell$, then the vertex y is also isolated from x.

Proof. Let G' be the graph obtained from G by adding a new vertex z with neighborhood K. In G', the set C of indivisible important zx-separators with border at most ℓ has size at most 4^ℓ by Theorem 10. Note that there are at most ℓ vertices of K which are not in a zx-separator of size at most ℓ. The size of K is at least $\ell \cdot 2^{4^\ell} + 1$, so there exists a subset T of K of size at least $\ell + 1$ such that for every important separator S' in C, we have $T \cap S' = T$. Thus, every indivisible separator S of size at least $\ell + 1$ satisfies $T \cap S = T$. Indeed, every indivisible separator S contains an important separator S' such that $S' \cap T = T$, thus $S \cap T = T$.

We pick a vertex y in T. Let us prove that y satisfies the conclusion of Theorem 15.

In the graph G, consider a set A of a edges which isolates x from all the elements of K apart from a subset B of size b with $a + b \leq \ell$. Let F be the set of edges $A \cup \{zb \ : \ b \in B\}$ of G'. Note that F is a zx-edge separator. We denote by X the component of x in $G' \setminus F$. Since $V(G') \setminus X$ is an indivisible zx-separator with border at most ℓ, it contains by Corollary 7 an indivisible important zx-separator U with border at most ℓ. In other words, U belongs to C.

Let us first observe that the set T cannot be disjoint from U. Indeed T has size $\ell+1$ and each vertex of T is adjacent to z, thus the border of U would exceed ℓ. Hence T is included in U, and the set of edges A isolates T from x in G, and in particular separates y from x. This concludes the proof of Theorem 15. □

This connectivity result allows us to bound the *request degree* of a vertex v, *i.e.* the number of requests with v as an endpoint.

Corollary 16. *In a* MULTICUT *instance with deletion allowance k, the maximum request degree can be reduced to at most $h(k+1) = \mathcal{O}(k \cdot 4^k)$ in FPT time.*

Proof. Consider a vertex x, and denote by K the set of vertices forming a request with x. Assume that $|K| \geq h(k+1)$. In this proof we consider separators rooted in x. By Theorem 15, there is a vertex y of K such that every subset S containing x and verifying $\delta(S) + |S \cap K| \leq k+1$ is such that $y \notin S$. We simply remove the request xy from the set of requests. Indeed, let F be a multicut of size at most k of this reduced instance. Let S be the component of x in $G \setminus F$. As F is a multicut, no element of $K \setminus y$ belongs to S (so $K \cap S| \leq 1$). Moreover $\delta(S) \leq k$ since at most k edges are deleted. Thus $\delta(S) + |S \cap K|$ is at most $k+1$, which implies that $y \notin S$. In other words, even if we do not require to cut x from y, a multicut of the reduced instance must cut the request xy. Therefore removing the request xy from R is correct. When such a reduction can no longer be performed, each vertex has request degree smaller than $h(k+1)$. □

3.3 Cherry reduction

An *x-cherry*, or simply *cherry* is a connected induced subgraph C of G with a particular vertex x called *attachment vertex* of C such that there is no edge from $C \setminus x$ to $G \setminus C$ and no request has its two terminals in $C \setminus x$. Note that we can always assume that the restriction of a multicut to an x-cherry C is the border of an important separator of C, where x is the root. Indeed, one can only improve the number of separated pairs of terminals by taking less vertices in the component of x (since there is no internal request in a cherry). If $u \in C \setminus x$, a request $uv \in R$ is *irrelevant* if for every multicut F of at most k edges of $R \setminus uv$ and such that $F \cap C$ is the border of an important separator in C, F actually separates u from v.

Theorem 17. *Let C be an x-cherry of an instance with deletion allowance k. We can find in FPT time a set $K(C)$ of at most $b(k) := h(k+1)\alpha(k, h(2k+1)) = k^{\mathcal{O}(k^3)}$ terminals in $C \setminus x$, such that if F is a set of at most k edges which separates all requests with one endpoint in $K(C)$ and such that $F \cap C$ is the border of an important separator, then F actually cuts all requests with an endpoint in $C \setminus x$.*

Proof. By Corollary 16, we can assume that all terminals have request degree at most $h(k+1)$. Let L be the set of terminals in $C \setminus x$. We assume that $|L| > b(k)$. Our goal is to show that there exists an irrelevant request with one endpoint in L. Let us consider the bipartite request graph B formed by the set of requests with one endpoint in L. The graph B is bipartite since $C \setminus x$ has no internal requests. Recall that if a bipartite graph with vertex bipartition (X, Y) has maximum degree d and minimum degree one, there exists a matching with at least $|X|/d$ edges. Indeed, in this case the edges can be partitioned into d matchings and the graph contains at least $|X|$ edges.

The request graph B thus contains a matching M of size at least $\alpha(k, h(2k+1))$ such that each request in M has one endpoint in L and the other endpoint outside $C \setminus x$. Let $K := V(M) \cap V(C \setminus x)$. We first only consider the cherry C where x is the root. Since the size of K is at least $\alpha(k, h(2k+1))$, the set K contains by Theorem 14 a subset K' of size $h(2k+1)$ such that every important separator S with border at most k verifies $S \cap K' = \emptyset$ or $|K' \setminus S| \leq k$. Let M' be the set of edges of M

having an endpoint in K'. We denote by L' the set of vertices $M' \setminus K'$, *i.e.* the endpoints of edges in M' which do not belong to $C \setminus x$. Now let us consider the graph $G' := G \setminus (C \setminus x)$ with root x. The set L' has size at least $h(2k+1)$, thus by Theorem 15 there is a vertex y in L' such that, whenever we delete k edges in G' such that at most k vertices of L' belong to the component of x, then y does not belong to the component of x. The vertex y being an element of L', we consider the request $zy \in M'$, where z belongs to $V(C \setminus x)$.

We claim that the request zy is irrelevant. Indeed, let F be a multicut of $R \setminus zy$ with at most k edges such that $F_C = F \cap C$ is the border of an important separator. Let S be the component of x in $C \setminus F_C$. The set S is an important separator and has a border of size at most k, hence, either S completely isolates x from K' or S isolates at most k vertices of K' from x. If K' is isolated from x, then in particular x is disconnected from y, hence the request zy is separator by F. So we assume that a subset K'' containing all but at most k vertices of K' is included in S. Hence, denoting by L'' the other endpoints of the edges of M' intersecting K'', this means that F must disconnect x from L''. Therefore, the set F of at most k edges disconnects x from at most $k+1$ elements of L' (the k elements of L'' and possibly y), so by definition of y, the set F disconnects x from y. In particular zy is separator by F. Thus the request zy is indeed irrelevant. All the computations so far are FPT.

We repeat this process, removing irrelevant requests until the size of L does not exceed $b(k)$. We then set $K(C) := L$, and the conclusion of Theorem 17 holds. □

Let C be a cherry of a graph G with deletion allowance k. A subset $\mathcal{L}$ of the edges of C is *active* when we can assume that a multicut uses only edges of $\mathcal{L}$ in C, or more formally: if a multicut F of size at most k exists, then there exists a multicut F' of size at most $|F|$ such that $F' \setminus C = F \setminus C$ and $F' \cap C \subseteq \mathcal{L}$. When the set $\mathcal{L}$ is clear from the context, we say by extension that edges of $\mathcal{L}$ themselves are *active*.

Lemma 18. *Let C be an x-cherry of a graph G with deletion allowance k, and let K be the set of all terminals of $C \setminus x$. Let $\mathcal{L}(C)$ be the union of all borders of separators of $C^y_{\leq k}$, where $y \in K$. Then $\mathcal{L}(C)$ is active, and has size at most $4^{k+1} \cdot |K|$.*

Proof. Assume that F is a multicut of size at most k. Let S be the component of x in $C \setminus F$. Let T be an important separator with $T \subseteq S$ and $\delta(T) \leq \delta(S)$. If a component U of $\overline{T}$ does not intersect K (*i.e.* no terminal is in the component), then $F \setminus \Delta(U)$ is still a multicut. Hence, we can assume that all components U of $\overline{T}$ intersect K, in which case $\Delta(U) \in C^y_{\leq k}$ for some y in K, hence $\Delta(T)$ is included in $\mathcal{L}(C)$. The set $F' = (F \setminus C) \cup \Delta(T)$ is a multicut, and the size bound for $\mathcal{L}(C)$ follows from Theorem 10: $C^y_{\leq k}$ contains at most 4^k indivisible important separators of size at most k, and $\mathcal{L}(C)$ is a union of $|K|$ such sets. □

Theorem 19. *Let $H_1, H_2, \ldots, H_p$ be x-cherries of a graph G with deletion allowance k such that $H_1 \setminus x, H_2 \setminus x, \ldots, H_p \setminus x$ are pairwise disjoint. Assume that for every i, $U_i := H_1 \cup \cdots \cup H_i$ is a cherry. Then every set U_i has a bounded active set $\mathcal{L}_i$ such that $\mathcal{L}_j \cap U_i \subseteq \mathcal{L}_i$ whenever $i \leq j$.*

Proof. By Theorem 17, we can reduce the set of terminals in U_1 to a bounded set K_1. The set $\mathcal{L}_1 = \mathcal{L}(U_1)$ is bounded and active by Lemma 18. The requests of $C_1 \setminus K_1$ are irrelevant in U_2 since they are irrelevant in U_1, hence we can assume that Theorem 17 applied to U_2 yields a set of terminals $K_2 \subseteq K_1 \cup C_2$. Let $\mathcal{L}_2$ be the active edges associated to K_2. Note that if an edge $e \in \mathcal{L}_2$ is in U_1, then the edge e must belong to a set $C^y_{\leq k}$ for some $y \in K_2 \cap U_1$. Since $K_2 \subseteq K_1 \cup C_2$, we have $y \in K_1$, and so $e \in \mathcal{L}_1$, which is the property we are looking for. We extract K_3 from $K_2 \cup C_3$, and iterate this process to form the sequence $\mathcal{L}_i$. □

4 Reducing Multicut to Component Multicut

Let $G = (V, E)$ be a connected graph, and R be a set of requests. A *vertex-multicut* Y is a subset of V such that every xy-path of G where $xy \in R$ contains a vertex of Y. Let A be a connected component of $G \setminus Y$. We call *Y-component*, or *component*, the union of A and its set of neighbors in Y. Let C be a Y-component, the vertices of $C \cap Y$ are the *attachment vertices* of C.

4.1 Component Multicut

Our first intermediate problem is formally expressed below. Informally, the Y-components have at most two attachment vertices. Each Y-component with two attachment vertices has a distinguished path called *backbone*, and the multicut restricted to a component must consist of exactly one edge of the backbone plus a fixed number of other edges. Finally, the vertex-multicut Y must be split by the solution.

Component Multicut:
Input: A connected graph $G = (V, E)$, a vertex-multicut Y, a set of requests, and q integers integers $f_1, \dots, f_q$ such that:

1. There are q Y-components $G_1, \dots, G_q$ with two attachment vertices x_i and y_i. The other Y-components have only one attachment vertex.
2. Every G_i has an x_iy_i-path denoted P_i called the *backbone* of G_i, such that the deletion of an edge of P_i decreases the edge connectivity in G_i between x_i and y_i.
3. The integers $f_1, \dots, f_q$ are such that $f_1 + \cdots + f_q \leq k - q$.

Parameter: k.
Output: TRUE if there exists a multicut F such that:

1. every path P_i contains exactly one edge of F,
2. every component G_i contains exactly $1 + f_i$ edges of F,
3. the solution F *splits* Y, *i.e.* each connected component of $G \setminus F$ contains at most one vertex of Y.

Otherwise, the output is FALSE.

The edges of G which do not belong to the backbones are called *free edges*. The backbone P_i, in which only one edge is deleted, is the crucial structure of G_i. Indeed, the whole proof consists of modifying each Y-component G_i step by step to finally completely reduce it to the backbone P_i. Here, f_i is the number of free edges that we can delete in G_i. Observe that $k - q - f_1 - \cdots - f_q$ edges can be deleted in Y-components with one attachment vertex. Our first reduction is the following:

Theorem 20. *Multicut can be reduced to Component Multicut in FPT time.*

The remaining of Section 4 is devoted to the proof of Theorem 20. We first construct a vertex-multicut Y through iterative compression. Then, we prove that we can reduce to Y-components with one or two attachment vertices. Finally, we show that we can assume that every component with two attachment vertices has a path in which exactly one edge is chosen in the solution. This is our backbone.

4.2 The Vertex-Multicut Y

This subsection is devoted to proving by iterative compression that Multicut is equivalent to the following problem, as was first noted in [29]:

Restricted Multicut:
Input: A graph G, a set of requests R, an integer k, a vertex multicut Y of size at most $k + 1$.
Parameter: k.
Output: TRUE if there is a multicut of size at most k which splits Y, otherwise FALSE.

Lemma 21. *MULTICUT is FPT-equivalent to RESTRICTED MULTICUT.*

Lemma 21 follows from the following two Lemmas:

Lemma 22. *MULTICUT can be solved in time $\mathcal{O}(f(k)n^c)$ if the MULTICUT variant where a vertex multicut of size at most $k+1$ is additionally given in the input can be solved in time $\mathcal{O}(f(k)n^{c-1})$.*

Proof. By induction on n, we solve MULTICUT in time $f(k)(n-1)^c$ on $G - v$, where $v \in V(G)$. If the output is FALSE, we return FALSE, otherwise the output is a multicut F of size at most k. Let X be a vertex cover of F of size at most k (*i.e.* X contains one endpoint of each edge in F). The set $X \cup \{v\}$ is a vertex-multicut of the original instance, so we solve MULTICUT in time $f(k)n^{c-1} + f(k)(n-1)^c$ which is at most $f(k)n^c$. □

So we can assume that the input of MULTICUT contains a vertex-multicut Y of size at most $k+1$.

Lemma 23. *We can assume that the solution F splits Y.*

Proof. To a solution F is associated the partition of $G \setminus F$ into connected components. In particular, this induces a partition of Y. We branch over all possible partitions of Y. In a given branch, we simply contract the elements of Y belonging to a same part of the partition. □

This concludes the proof of Lemma 21.

During the following reduction proof, the size of the set Y will never decrease. Since one needs $k+1$ edges to separate $k+2$ vertices, the size of Y cannot exceed $k+1$, otherwise we return FALSE. Hence the primary invariant is the size of Y, and we immediately conclude if we can increase $|Y|$.

4.3 Reducing attachment vertices

Our second invariant, which we intend to maximize, is the number of Y-components with at least two attachment vertices. This number cannot exceed k, since a solution must split Y. Our third invariant is the sum of the edge connectivity between all pairs of vertices of Y, which we want to increase. This invariant is bounded by $k\binom{|Y|}{2}$ since the connectivity between two elements of Y is at most k. Note that this third invariant never decreases when we contract vertices.

Lemma 24. *If C is a Y-component with at least three attachment vertices, we improve the invariant.*

Proof. Let x, y, z be attachment vertices of C. Let λ be the edge-connectivity between x and y in C. Let $P_1, \ldots, P_\lambda$ be a set of edge-disjoint xy-paths. A *critical edge* is an edge which belongs to some xy-edge separator of size λ. Note that every critical edge belongs to some path P_i. A *slice* of C is a connected component of C minus the critical edges. Given a vertex v of C, the *slice of* v, denoted by $SL(v)$, is the slice of C containing v. Let $B(z)$ be the *border* of $SL(z)$, *i.e.* the set of vertices of $SL(z)$ which are incident to a critical edge. Note that $B(z)$ intersects every path P_i on at most two vertices, namely the leftmost vertex of P_i belonging to $SL(z)$ and the rightmost vertex of P_i belonging to $SL(z)$. In particular, $B(z)$ has b vertices, where $b \leq 2\lambda$.

We branch over $b+1$ choices to decide whether one of the b vertices of $B(z)$ belongs to a component of $G \setminus F$ (where F is the solution) which does not contain a vertex of Y. When this is the case, the vertex is added to Y, which increases the primary invariant. In the last branch, all the vertices of $B(z)$ are connected to a vertex of Y in $G \setminus F$. We branch again over all mappings f from $B(z)$ into Y. In each branch, the vertex $v \in B(z)$ is connected to $f(v) \in Y$ in $G \setminus F$. Hence we can contract every vertex $v \in B(z)$ with the vertex $f(v) \in Y$. This gives a new graph G'. We denote by S' the subgraph $SL(z)$ in G'. Observe that S' is a Y-component of G'.

If x and y belong to S', then the edge connectivity between x and y has increased. Indeed, there is now a path P between x and y inside S', in particular P has no critical edge. Thus the connectivity between x and y has increased, so the invariant has improved. We assume without loss of generality that x does not belong to S'.

If S' contains an element of Y distinct from z (w.l.o.g. distinct from x by the previous paragraph), then S' is a Y-component with at least two attachment vertices. Moreover, there exists a path P in $C \setminus S'$ from x to $B(z)$. Hence we have created an extra Y-component with at least two attachment vertices in G', which improves the second invariant.

In the last case, z is the only vertex of Y which belongs to S'. Therefore, $B(z)$ is entirely contracted to z. In particular z is now incident to a critical edge e. So there exists an xy-separator A with $\delta(A) = \lambda$ and $e \in \Delta(A)$. Without loss of generality, we assume that $z \notin A$ (otherwise we consider the yx-separator $\overline{A}$). We denote by B the vertices of $\overline{A}$ with a neighbor in A. In particular, B contains z, has size at most λ, and every xy-path in C contains a vertex of B. Let us denote by L the set $A \cup B$ and by R the set $\overline{A}$. Note that $L \cap R = B$. We now branch to decide in which components of $G \setminus F$ the elements of B are partitioned. If an element of B is not connected to Y in $G \setminus F$, we improve the invariant. If each element of B is contracted to a vertex of Y, both L and R in the contracted graph are Y-components with at least two attachment vertices (respectively $\{x, z\}$ and $\{y, z\}$). We again improve the invariant. □

4.4 Backbones

We now assume that every component has at most two attachment vertices. Let $G_1, \dots, G_q$ be the components of G with two attachment vertices. We denote by λ_i the edge connectivity of G_i between its two attachment vertices x_i and y_i. Recall that the third invariant is the sum of the λ_i for $i = 1, \dots, q$.

Lemma 25. *We can assume that x_i and y_i have degree λ_i in G_i.*

Proof. Let A be the unique important x_iy_i-separator with $\delta(A) = \lambda_i$ in the graph G_i rooted in x_i. Let B be the set of vertices of A with a neighbor in $\overline{A}$. We now branch to decide how the components of $G \setminus F$ partition B. If a vertex of B is not connected to a vertex of Y in $G \setminus F$, we can add it to Y and improve the invariant. If a vertex of B is contracted to a vertex $y_j \in Y$, we increase λ_j. Hence all elements of B are contracted to x_i. Therefore A becomes an x_i-cherry, hence $A \setminus x_i$ is removed from G_i. The degree of x_i inside G_i is now exactly λ_i. We apply the same argument to reduce the degree of y_i to λ_i. □

We now branch over all partitions of k into $k_0 + k_1 + \cdots + k_q = k$, where k_i is the number of edges of the solution chosen in G_i when $i > 0$, and k_0 is the total number of edges chosen in the y-cherries for $y \in Y$.

Lemma 26. *Every component G_i can be deleted or has a backbone.*

Proof. If $k_i \geq 2\lambda_i$, then the whole component G_i can be disconnected from the rest of the graph by removing the edges in G_i incident to x_i and y_i (which is the best possible since no request has both endpoints in a same component). This reduces the second invariant, the number of components with at least two attachment vertices. So we can assume that $k_i \leq 2\lambda_i - 1$. Let $P_1, P_2, \dots, P_{\lambda_i}$ be edge-disjoint x_iy_i-paths.

Our algorithm now branches $2\lambda_i$ times, where the branches are called B_j and B'_j for $j = 1, \dots, \lambda_i$. In the branch B_j, we assume that there is only one edge of the solution selected in P_j, and that this edge is critical, *i.e.* belongs to an x_iy_i-separator of size λ_i. In the branch B'_j, we assume that all the edges of the solution selected in P_j are not critical. Let us show that every solution F belongs to one of these branches. If F does not belong to any branch B'_j, this means that F uses at least one critical edge in each P_j. But since $k_i \leq 2\lambda_i - 1$, some path P_j only intersects F on one edge, which is therefore critical. Hence F is a solution in the branch B_j. Thus this branching process is valid. In the branch B_j, we contract all non critical edges of P_j, therefore P_j is the backbone we are looking for. In the branch B'_j, we contract all critical edges of P_j, hence the connectivity λ_i increases. We thus improve the invariant. □

This concludes the proof of Theorem 20. To sum up, the invariants in this reduction from Multicut to Component Multicut are (in decreasing order of importance):

- $|Y|$, to maximize, bounded by $k+1$.
- The number of components with at least two attachment vertices, to maximize, bounded by k.
- The sum of the connectivity of all pairs of vertices of Y, to maximize, bounded by k^3.

The tree which represents the branchings made by the algorithm has depth $\mathcal{O}(k^5)$ (the product of the bounds on the invariants). The degree of its nodes is bounded by $\mathcal{O}^*(k^{\mathcal{O}(k)})$. Indeed, when reducing the components with three or more attachment vertices in Lemma 24, or when ensuring that the degree of attachment vertices in a given component is exactly their connectivity in Lemma 25, we improve the invariant at a cost of $\mathcal{O}^*(k^{\mathcal{O}(k)})$, and these are the bottlenecks. Finally, the amount of work performed at each node vanishes in front of the number of branches, which gives:

Observation 27. The reduction from Multicut to Component Multicut is executed in time $\mathcal{O}^*(k^{\mathcal{O}(k^6)})$.

Reductions which are only performed once do not impact the overall running time: reducing Multicut to Restricted Multicut is achieved in time $\mathcal{O}^*(k^{\mathcal{O}(k)})$ in Lemma 21; also, there are $\mathcal{O}(k^2)$ components with at most 2 attachment vertices (all y-cherries are considered as a single component for $y \in Y$), hence branching to decide how many edges of the multicut per component costs $k^{\mathcal{O}(k^2)}$. These terms vanish in front of the $\mathcal{O}^*(k^{\mathcal{O}(k^6)})$ term.

5 Backbone Multicut is FPT

5.1 Backbone Multicut

We introduce here the problem Backbone Multicut, which is a generalization of Component Multicut. Our goal is to show that Backbone Multicut is solvable in FPT time, which implies that Component Multicut is FPT, which in turns implies that Multicut is FPT thanks to Theorem 20.

Backbone Multicut, formally defined below, differs from Component Multicut in two ways. Backbone Multicut contains half-requests of the type (u, y, v), where $u, v \in V$ and $y \in Y$. Cutting the half-request (u, y, v) means cutting all paths from u to v going through y. Also, an instance of Backbone Multicut can express simple properties on the edge of a backbone selected in the multicut, which allows us to enrich instances.

Backbone Multicut:
Input: A connected graph $G = (V, E)$, a set R of half-requests, a set Y of at most $k+1$ vertices, a set B of q variables, a set $\mathcal{C}$ of clauses, and q non negative integers $f_1, \dots, f_q$ such that:

1. G has q Y-components called G_i with two attachment vertices $x_i, y_i \in Y$, where $i = 1, \dots, q$. Moreover, G_i has a backbone Q_i (a prescribed x_iy_i-path) and the x_iy_i-connectivity in G_i is λ_i.
2. The set R contains *half-requests*, *i.e.* sets of triples (u, y, v), informally meaning that vertex u sends a request to vertex v via y, where $y \in Y$. Also, Y is a u, v-separator for every half-request $(u, y, v) \in R$.
3. The set B contains q integer-valued variables $c_1, \dots, c_q$. Each variable c_i corresponds to the deletion of one edge in the backbone Q_i. Formally, if the edges of Q_i are $e_1, \dots, e_{\ell_i}$, ordered from x_i to y_i, the variable c_i can take all possible values from 1 to ℓ_i, and $c_i = r$ means that we delete the edge e_r in Q_i.

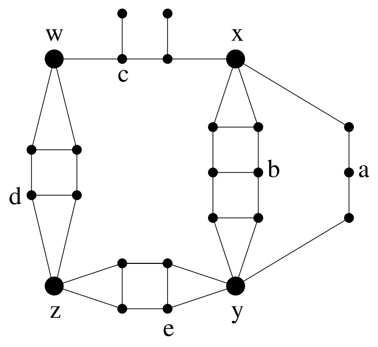

Figure 4: Fat vertices w, x, y, z belong to Y. The request ab can be simulated by two half-requests: (a, x, b) and (a, y, b). The request cd can be simulated by the single half-request (c, w, d), as paths between c and d which do not go through w have to go through x and y, and hence will be automatically cut when Y is split. By the same argument, the request ce is irrelevant and can be deleted.

4. The clauses in $\mathcal{C}$ have four possible types: $(c_i \leq a \Rightarrow c_j \leq b)$, or $(c_i \leq a \Rightarrow c_j \geq b)$, or $(c_i \geq a \Rightarrow c_j \geq b)$, or $(c_i \geq a \Rightarrow c_j \leq b)$.
5. The integers $f_1, \ldots, f_q$ sum to a value at most k. Each integer f_i corresponds to the number of free edges (*i.e.* edges of G which are not in a backbone) of the solution which are chosen in G_i.

Parameter: k.
Output: TRUE if:

1. There exists an assignment of the variables of B which satisfies $\mathcal{C}$.[1]
2. There exists a subset F of at most k free edges of G, which contains f_i free edges in G_i for $i = 1, \ldots, q$.
3. The union F' of the set F together with the backbone edges corresponding to the variables of B splits Y and intersects every half-request of R, *i.e.* for every half-request $(u, y, v) \in R$ every path between u and v containing y intersects F'.

Otherwise, the output is FALSE.

Note that the deletion allowance of BACKBONE MULTICUT is $k + q$. COMPONENT MULTICUT directly translates into BACKBONE MULTICUT with an empty set of clauses, and where each request is simulated by one or two half-requests (see Figure 4).

This section is devoted to the proof of the following result, which will conclude the proof of the fixed-parameter tractability of MULTICUT:

Theorem 28. *BACKBONE MULTICUT can be solved in FPT time.*

5.2 Invariants

Our primary invariant is the sum of the numbers of free edges f_i for $i = 1, \ldots, q$, which starts with value at most k and is non-negative. A branch in which we can decrease this primary invariant will be considered solved. The secondary invariant is the sum of the $\lambda_i - 1$, called the *free connectivity*, which we try to increase. Observe that this invariant is bounded above by k. For the last invariant, recall that the *slice* $SL(v)$ of a vertex v in a component G_i is the connected component containing v of G_i minus the *critical* edges of G_i, *i.e.* edges of λ_i-separators. Observe that since all edges of a backbone are critical, the slices of distinct vertices in a backbone do not intersect. See Figure 5.

[1]Formally, if c_i is assigned value r, then variables $c_i \leq a$ for $a \geq r$ and variables $c_i \geq a$ for $a \leq r$ are true and variables $c_i \geq a$ for $a \geq r + 1$ and variables $c_i \leq a$ for $a \leq r - 1$ are false.

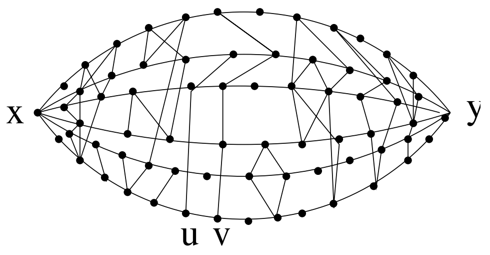

Figure 5: A component in a Backbone Multicut instance with two attachment vertices $x, y \in Y$. The bottom xy-path is the backbone P_1, and the other "horizontal" xy-paths are the prescribed paths $P_2, \dots, P_6$, from bottom to top. Each edge of the backbone does indeed belong to a λ-separator (*i.e.* an xy-separator consisting of six edges). In other words, there exists no path between two distinct vertices of the backbone which consists only of *diagonal edges*, *i.e.* edges which do not belong to the paths $P_1, \dots, P_6$. Thus the slices of two distinct backbone vertices are disjoint. The tag of vertex u, as defined in Subsection 5.4, is $\{1, 4, 5\}$ and the tag of vertex v is $\{1, 3, 4, 5, 6\}$. The slice connectivity of u is $sc(u) = 6 - 3 = 3$, and $sc(v) = 6 - 5 = 1$. The slice connectivity of this component is 5, as there exists a vertex of the backbone P_1 with no neighbor outside P_1.

The *slice connectivity* of a vertex v in Q_i is the x_iy_i-edge-connectivity of $G_i \setminus SL(v)$ (where $SL(v)$ is considered as a vertex set). We denote it by $sc(v)$. For example, if the set of neighbors of v intersects every x_iy_i-path in $G_i \setminus Q_i$, then we have $sc(v) = 0$. Conversely, if $v \in Q_i$ has only neighbors in Q_i, then $sc(v) = \lambda_i - 1$. The *slice connectivity* sc_i of G_i is the maximum of $sc(v)$, where $v \in Q_i$. The third invariant is the sum sc of the sc_i, for $i = 1, \dots, q$, and we try to minimize this invariant. Observe that sc is always at most k.

Our goal is to show that we can always improve the invariant, or conclude that $\lambda_i = 1$ for all i.

In the following, we consider a component G_i with $\lambda_i > 1$, say G_1.

To avoid cumbersome indices, we assume that the attachment vertices of G_1 are x and y, and that their edge-connectivity is denoted by λ instead of λ_1. Moreover, we denote by P_1 the backbone of G_1, and we assume that $P_1, P_2, \dots, P_\lambda$ is a set of edge-disjoint xy-paths in G_1. We visualize x to the left and y to the right (see Figure 5). Hence when we say that a vertex $u \in P_i$ is to the left of some vertex $v \in P_i$, we mean that u is between x and v on P_i.

5.3 Contracting edges

In our proof, we contract edges of the backbone and also free edges which are not critical. We always preserve the fact that the edges of the backbone are critical.

When contracting an edge of the backbone P_1, we need to modify several parameters. Assume that the edges of P_1 are $e_1, \dots, e_\ell$. The variable c_1 represents the edge of P_1 which belongs to the multicut. Now assume that the edge $e_i = v_iv_{i+1}$ is contracted. All the indices of the edges which are at least $i + 1$ are decreased by one. All the constraints associated to the other backbones are not affected by the transformation. However, each time a clause contains a literal $c_1 \geq j$, where $j > i$, this literal must be replaced by $c_1 \geq j - 1$. Similarly, each occurrence of $c_1 \leq j'$ for $j' \geq i$ must be replaced by $c_1 \leq j' - 1$. If a set of edges is contracted, we perform the contractions one by one.

The collection of paths $P_2, \dots, P_\lambda$ can be affected during our contractions since it can happen that a path P_i with $i \geq 2$ contains both endpoints of a contracted edge uv. In such a case, we remove from P_i the loop formed by the contraction, *i.e.* the subpath of P_i between u and v. We thus preserve our path collection.

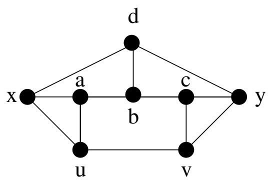

Figure 6: This component has two attachment vertices x and y. Its backbone is $P_1 = xuvy$, and we have $P_2 = xabcy$ and $P_3 = xdy$. Both u and v have tag $t(u) = t(v) = \{1, 2\}$, since all edges of P_2 are edges of λ-separator. After contracting u and v, edges ab and bc are no longer edges of λ-separator, and the slice of the resulting vertex $u = v$ becomes $\{u = v, a, b, c, d\}$, and its tag becomes $\{1, 2, 3\}$, which strictly contains $t(u) \cup t(v)$.

5.4 Choosing a stable edge

Let v be a vertex of P_1. The *tag* of v is the subset $t(v) := \{i \mid P_i \cap SL(v) \neq \emptyset\}$, *i.e.* the set of indices of the paths intersecting the slice of v. Note that $t(v)$ contains 1. Observe also that the slice connectivity of G_1 is the maximum of $\lambda - |t(v)|$, where v belongs to P_1. See Figure 5.

By extension, the *tag* of an edge v_iv_{i+1} of the backbone P_1 is the ordered pair $(t(v_i), t(v_{i+1}))$. When speaking of an XY*-edge*, we implicitly mean that its tag is (X, Y). In particular, the edge of P_1 which is selected in the solution has a given tag. We branch over the possible choices for the tag XY of the deleted edge of P_1. Let us assume that the chosen edge has tag XY.

Lemma 29. *If $X \neq Y$, we improve the invariant.*

Proof. Since only one edge is cut in the backbone, we can contract all the edges of P_1 with tags different from XY. Observe that when contracting some UV-edge of P_1, the tag of the resulting vertex contains $U \cup V$ since the slice of the resulting vertex contains the union of both slices (it can actually be larger, see Figure 6). After contraction, all the edges of P_1 between two consecutive occurrences of XY-edges are contracted, hence the tag of every vertex of P_1 now contains $X \cup Y$. In particular, the slice connectivity of G_1 decreases while the free connectivity is unchanged. Thus the invariant has improved. □

Therefore we may assume that we choose an XX-edge in the solution. Let us contract all the edges of P_1 which are not XX-edges. By doing so, the tag of every vertex of P_1 contains X. After this contraction, the instance is modified, hence we have to branch again over the choice of the tag of the edge chosen in the solution. Any choice different from XX increases the slice connectivity. Hence we can still assume that the tag of the chosen edge is XX.

The slice connectivity of G_1 is $\lambda - |X|$. An XX-edge uv of the backbone is *unstable* if, when contracting uv, the tag of the vertex $u = v$ increases (*i.e.* strictly contains X). Otherwise uv is *stable*. We branch on the fact that the chosen XX-edge is stable or unstable.

Lemma 30. *If the chosen XX-edge is unstable, we improve the invariant.*

Proof. We enumerate the set of all unstable edges from left to right along P_1, and partition them according to their index into the odd indices and the even indices. We branch according to the index of the chosen unstable edge. Assume for instance that the chosen unstable edge has odd index. We contract all the edges of P_1 except from the odd unstable edges. We claim that the tag of every vertex now strictly contains X. Indeed, all edges of P_1 between two consecutive odd unstable edges are contracted, in particular some even unstable edge. Thus, since this even edge is unstable, the tag now strictly contains X. Hence the slice connectivity decreases. □

5.5 Contracting slices

In this part, we assume that the chosen edge of P_1 is a stable XX-edge. A vertex v of P_1 is *full* if v belongs to every P_i, where $i \in X$ (see Figure 7). Our goal in this subsection is to show that we

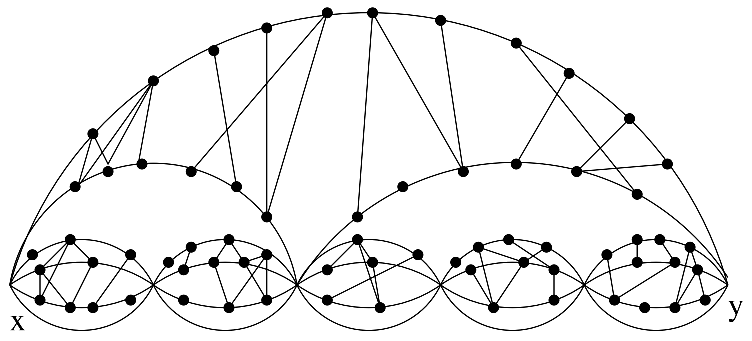

Figure 7: All backbone vertices of this component are full, where $X = \{1,2,3,4\}$, with the backbone P_1 at the bottom.

can reduce to the case where $X = \{1,\dots,\lambda\}$. By the previous section, a branching which increases the tag of the chosen edge would improve the invariant. So we assume that in all our branchings, the chosen edge is still a stable XX-edge.

Lemma 31. *We can assume that all backbone vertices are full.*

Proof. We can first assume that there are at most k vertices with tag X between two full vertices. Indeed, let us enumerate $w_1, w_2, \dots$ the vertices with tag X from left to right along the backbone P_1. A solution F contains at most k free edges and the slices of the vertices of the backbone are disjoint, so at most k slices of vertices w_i contain an edge of F. Hence, if we partition the set of all slices $SL(w_i)$ into $k+1$ classes according to their index i modulo $k+1$, the solution F will not intersect one of these classes. We branch on these $k+1$ choices. Assume for instance that F does not contain an edge in all $SL(w_i)$ where i divides $k+1$. Therefore, we can safely contract each of such slices $SL(w_i)$ onto w_i. This makes w_i a full vertex.

Let us now enumerate the full vertices $z_1, z_2, \dots$ from left to right. Let uv be a stable XX-edge. There exists a full vertex z_i to the left of u (with possibly $z_i = u$) and a full vertex z_{i+1} to the right of v. Since the number of vertices with tag X between z_i and z_{i+1} is at most k, the number of XX-edges between z_i and z_{i+1} is at most $k+1$. The *rank* of uv is the index of uv in the enumeration of the edges between z_i and z_{i+1} from left to right. Every edge of P_1 has some rank between 1 and $k+1$. In particular, we can branch over the rank of the selected stable XX-edge. Assume for instance that the rank of the chosen edge is 1. We then contract all edges which are not stable XX-edges with rank 1. This leaves only full vertices on P_1 since by construction there is a full vertex between two edges of the same rank. $\square$

Note that after performing the reduction of Lemma 31, if v_iv_{i+1} is a stable XX-edge, then for every vertex $w \in P_j$ with $j \in X$ which lies between v_i and v_{i+1} in P_j, every wY-path contains v_i or v_{i+1}. In particular, if $X = \{1,\dots,\lambda\}$ then v_i is an xy-separator-vertex in G_1, and v_{i+1} as well.

Lemma 32. *We can assume that $X = \{1,\dots,\lambda\}$. In other words, we can reduce to the case where every vertex of P_1 is a cut-vertex of G_1.*

Proof. Assume that X is not equal to $\{1,\dots,\lambda\}$. We show that we can partition the component G_1 into two components G_1^1 and G_1^2. This partition leaves the free-connectivity unchanged, but decreases the slice connectivity. A vertex v_i of the backbone P_1 is *left clean* if the edge $v_{i-1}v_i$ of P_1 is a stable XX-edge, but the edge v_iv_{i+1} of P_1 is not. The vertex v_i is *right clean* if the edge v_iv_{i+1} is a stable XX-edge, but the edge $v_{i-1}v_i$ is not. Finally, v_i is *clean* if both $v_{i-1}v_i$ and v_iv_{i+1} are stable XX-edges. When enumerating all left clean and right clean vertices from left to right, we obtain the sequence of distinct vertices $r_1, l_1, r_2, l_2, \dots, r_p, l_p$ where the r_i are the right clean vertices and the l_i are the left clean vertices. Observe that x and y do not appear in the sequence since their tag is $\{1,\dots,\lambda\}$. Let us consider a pair r_i, l_i. We say that a vertex v of G_1

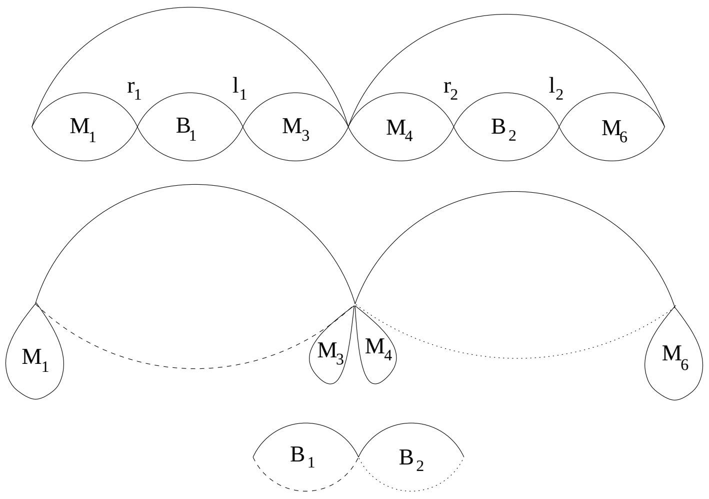

Figure 8: At the top, a component G_1 before transformation in the proof of Lemma 32. The new components G_1^1 and G_1^2 are depicted at the bottom and in the center respectively. The backbone edge in G_1^2 between w_i and w_{i+1} (dashed, mixed or dotted) is correlated to the subpath of the backbone contained in B_i in G_1^1 (similarly dashed, mixed or dotted). The vertices w_1 and r_1 are identified with vertex x, and the vertices w_4 and l_3 are identified with y, so that the xy-component G_1 is replaced by two xy-components G_1^1 and G_1^2. [3]

is *between* r_i and l_i if every path from v to x or y intersects $\{r_i, l_i\}$. Let B_i be the set of vertices which are between r_i and l_i. Let B be the union of B_i for $i = 1, \dots, p$.

Let G_1^1 be a copy of the subgraph induced by B on G_1. Observe that G_1^1 has p connected components, since $l_i \neq r_{i+1}$. We contract in G_1^1 the vertices l_i and r_{i+1}, for all $i = 1, \dots, p-1$, hence making G_1^1 connected. Finally, we contract the vertex r_1 of G_1^1 with x, and we contract the vertex l_p of G_1^1 with y, so that G_1^1 is a Y-component which has x and y as attachment vertices. The backbone P_1^1 of G_1^1 simply consists of the edges of the original backbone.

To construct G_1^2, we remove from G_1 all the vertices of B which are not left clean or right clean vertices. Hence no stable XX-edge is left in G_1^2. We contract backbone edges of G_1^2 as follows. All the backbone vertices between x and r_1 are contracted to a vertex $w_1 := x$, more generally all the vertices between l_i and r_{i+1} are contracted to a new vertex called w_{i+1}, and finally all the vertices between l_p and y are contracted to $w_{p+1} := y$. We add in G_1^2 the path $w_1, w_2, \dots, w_{p+1}$ which is the backbone P_1^2 of G_1^2. We correlate the edges of the backbone of G_1^1 and G_1^2 by adding clauses implying that the chosen edge of P_1^2 is $w_i w_{i+1}$ if and only if the chosen edge of P_1^1 is between r_i and l_i. We finally branch to split the number of free edges f_1 chosen in G_1 into $f_1^1 + f_1^2 = f_1$, the respective free edges deleted in G_1^1 and G_1^2. Let us call G' the graph G in which G_1 is replaced by G_1^1 and G_1^2. Note that the free edges of G_1 are partitioned into the free edges of G_1^1 and of G_1^2. Observe that the free-connectivity of G and G' are equal. However, the slice connectivity has decreased in G', since its value is 0 in G_1^1 and strictly less than $\lambda - |X|$ in G_1^2. Indeed, for $i = 1, \dots, p-1$, either the edge $l_i l_{i+1}$ is unstable or the tag of l_i or l_{i+1} strictly contains X. Hence, contracting all vertices between l_i and r_{i+1} strictly increases the tag of the resulting vertex in G_1^2. So the invariant improves. Figure 8 gives an example of this transformation.

Remains to prove that there exists a multicut in G' if and only if there exists a multicut in G which uses a stable XX-edge. This comes from the following observation. Let $e = v_j v_{j+1}$ be a stable XX-edge of P_1 between r_i and l_i (note that by definition there is no stable XX-edges between l_i and r_{i+1}). Let G_e be the graph obtained from G by deleting e, contracting x with all the vertices of P_1 to the left of v_j, and contracting y with all the vertices of P_1 to the right of v_{j+1}. Let G'_e be the graph obtained from G' by deleting e in P_1^1, deleting the edge $w_i w_{i+1}$ correlated to

[3]The names M_i are not used in the proof of Lemma 32, their purpose is just to identify what vertices which do not belong to a set B_i become in G_1^2.

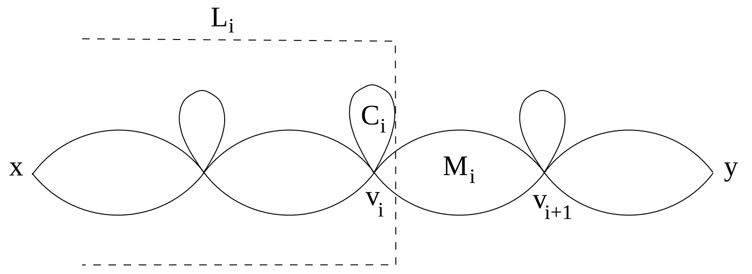

Figure 9: The left subgraph L_i of v_i. The set C_i is the v_i-cherry, and M_i is the lemon of the backbone edge $v_i v_{i+1}$.

e in P_1^2, contracting x with all the vertices of P_1^1 to the left of v_j and all the vertices of P_1^2 to the left of w_i, and contracting y with all the vertices of P_1^1 to the right of v_{j+1} and all the vertices of P_1^2 to the right of w_{i+1}. The key fact is that G_e is equal to G'_e. Hence the multicuts in G and G' selecting the edge e are in one to one correspondence. $\square$

The proof of Lemma 32 produces a new component, hence a new edge to be chosen in a backbone. This increases the deletion allowance by 1, but the number of free edges has not increased. The slice connectivity decreases, so the the invariant improves.

5.6 Reducing the lemons

Thanks to the results of the previous section, we can assume that each vertex of the backbone P_1 of G_1 intersects all paths P_i for $i = 1, \ldots, \lambda$. Let $v_i v_{i+1}$ be an edge of the backbone P_1. The *v_i-cherry* C_i is the set of all vertices u of G_1 such that every uY-path contains v_i.

The *lemon* M_i of $v_i v_{i+1}$ is the set consisting of v_i, v_{i+1} and of all vertices u of G_1 which do not belong to a cherry and such that every ux-path in G_1 contains v_i and every uy-path in G_1 contains v_{i+1}. Observe that when contracting $v_i v_{i+1}$, the lemon M_i becomes part of the v_i-cherry, where v_i denotes the resulting vertex. We denote by L_i the union of all C_j with $j \leq i$ and all M_j with $j < i$. We call L_i the *left subgraph* of v_i. Similarly, the *right subgraph* R_i of v_i is the union of all C_j with $j \geq i$ and all M_j with $j > i$. See Figure 9.

If a multicut F selects the edge $v_i v_{i+1}$ in the backbone, then the vertices $x, v_1, \ldots, v_i$ all lie in the same connected component of $G \setminus F$. When these vertices $x, v_1, \ldots, v_i$ are contracted to x, the set L_i becomes an x-cherry. Half-requests through y with an endpoint in L_i are automatically cut since F splits Y. Consider the terminals T_i of half-requests of L_i which are routed via x. Note that these half-requests are equivalent to usual requests, since L_i is now an x-cherry. By Theorem 17 we can reduce T_i to a bounded set of terminals K_i. This motivates the following key definition.

By Lemma 18, we define $\mathcal{L}_i$ to be a bounded active set of edges in the x-cherry obtained from L_i by contracting vertices $x, v_1, \ldots, v_i$. By Theorem 19, we can compute such sets $\mathcal{L}_i$ so that $\mathcal{L}_j \cap L_i \subseteq \mathcal{L}_i$ when $i \leq j$.

Let us say that a multicut F selecting $v_i v_{i+1}$ in P_1 is *proper* if $F \cap L_i$ is included in $\mathcal{L}_i$.

Lemma 33. *If there exists a multicut F of size at most k containing the backbone edge $v_i v_{i+1}$, then there is a proper multicut F' of size at most k containing $v_i v_{i+1}$.*

Proof. Consider a multicut F containing $v_i v_{i+1}$. As the set $\mathcal{L}_i$ is active in the cherry obtained by contracting the path $x, v_1, \ldots, v_i$ in L_i, there exists a multicut F' of size k such that $F' \setminus L_i = F \setminus L_i$ and $F' \cap L_i \subseteq \mathcal{L}_i$. Hence F' is proper and contains $v_i v_{i+1}$. $\square$

We denote by $\mathcal{L}$ the set of all subsets F of size at most k contained in some $\mathcal{L}_i$. We denote by c the maximum size of a set $\mathcal{L}_i$, whose size can be bounded by a function of k by Lemma 19. Note that c is bounded in terms of k.

Given two sets $F_i \subseteq \mathcal{L}_i$ and $F_j \subseteq \mathcal{L}_j$ with $j \geq i$, let us write $F_i \preceq F_j$ when $F_j \cap L_{i+1} \subseteq F_i$. Observe that $\preceq$ is a partial order. A subset $\mathcal{F}$ of $\mathcal{L}$ is *correlated* if:

- elements of $\mathcal{F}$ have the same size, and
- $\mathcal{F}$ is a chain for $\preceq$, *i.e.* for every F_i and F_j in $\mathcal{F}$, with $F_i \subseteq \mathcal{L}_i$, $F_j \subseteq \mathcal{L}_j$ and $j \geq i$, we have $F_j \cap L_{i+1} \subseteq F_i$.

Lemma 34. *There exists a partition $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_{k(2c)^k}$ of $\mathcal{L}$ into $k(2c)^k$ correlated sets.*

Proof. Let us prove by induction on $\ell = 0, \ldots, k$ that there exists no antichain for $\preceq$ in $\mathcal{L}$ consisting of $(2c)^\ell + 1$ sets of size at most ℓ. This clearly holds for $\ell = 0$. Assume that this holds for $\ell - 1$. By contradiction, let $A = \{F_1, F_2, \ldots, F_{(2c)^\ell+1}\}$ be an antichain of sets of size at most ℓ. Let t_i be an integer such that $F_i \subseteq \mathcal{L}_{t_i}$ for $i = 1, \ldots, (2c)^\ell + 1$. We assume that the sets F_i are enumerated in such a way that $t_i \leq t_j$ whenever $i \leq j$. The set F_1 is incomparable to all sets F_i with $i > 1$, hence $F_i \cap L_{t_1+1} \nsubseteq F_1$ for all $i > 1$. In particular $F_i \cap L_{t_1+1}$ is non-empty, hence all sets F_i, for $i = 1, \ldots, (2c)^\ell + 1$, have an edge in L_{t_1+1}. The sets F_i such that $t_i = t_1$ have an edge in $\mathcal{L}_{t_1}$ by definition. The sets F_i such that $t_i > t_1$ have an edge in $\mathcal{L}_{t_1+1}$ as $\mathcal{L}_{t_i} \cap L_{t_1+1} \subseteq \mathcal{L}_{t_1+1}$, by definition of the sets $\mathcal{L}_i$. Since the size of $\mathcal{L}_{t_1} \cup \mathcal{L}_{t_1+1}$ is at most $2c$, there exists a subset B of A of size at least $(2c)^{\ell-1} + 1$ of sets F_i sharing a same edge $e \in \mathcal{L}_{t_1} \cup \mathcal{L}_{t_1+1}$. The set $\{F \setminus e | F \in B\}$ has size $|B| \geq (2c)^{\ell-1} + 1$ and is an antichain of sets of size at most $\ell - 1$ by definition of $\preceq$. This contradicts the induction hypothesis.

By Dilworth's Theorem, there exists a partition of $\mathcal{L}$ into $(2c)^k$ sets totally ordered by $\preceq$, which can be be refined according to the cardinality to obtain a partition into $k(2c)^k$ correlated sets. Such a partition can be found in FPT time. $\square$

Let us now consider such a partition $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_{k(2c)^k}$ of $\mathcal{L}$ into correlated sets. Observe that by Lemma 33 we can restrict our search to multicuts of the following type in G_1:

- A backbone edge $v_i v_{i+1}$.
- Other edges in the lemon M_i, which separate v_i from v_{i+1} in M_i.
- Edges in $\mathcal{L}_i$.
- Edges in $\mathcal{R}_i$, which is defined analogously to $\mathcal{L}_i$, with the roles of vertices x and y reversed.

Lemma 35. *We can assume that there are no cherries C_i. Moreover, if a multicut of size at most k exists, there exists one which contains only edges in one lemon M_i.*

Proof. By Lemma 33, if there exists a multicut F containing the backbone edge $v_t v_{t+1}$, then there exists a proper multicut F' containing $v_t v_{t+1}$. By definition $F' \cap L_t \subseteq \mathcal{L}$.

We branch over the existence of a proper solution F' such that $F' \cap L_t \in \mathcal{F}_j$ for $j = 1, \ldots, k(2c)^k$, where t is the integer such that $v_t v_{t+1} \in F'$. Let us assume that we are in the branch where $F' \cap L_t \in \mathcal{F}_j$. A backbone edge $v_i v_{i+1}$ is in the *support* of $\mathcal{F}_j$ if there exists some $F_i \in \mathcal{F}_j$ such that $F_i \subseteq \mathcal{L}_i$. When $v_i v_{i+1}$ is in the support we say that lemon M_i is a *support lemon.* In this case, there actually exists a unique set in $\mathcal{F}_j$, which we denote by F_i, such that $F_i \subseteq \mathcal{L}_i$, as $\mathcal{F}_j$ is totally ordered under $\preceq$. Let ℓ be the number of edges of the sets in $\mathcal{F}_j$.

Claim 36. *For all $F_a \in \mathcal{F}_j$, if M_i is a support lemon then $F_a \cap M_i = \emptyset$.*

Proof. As $\mathcal{L}$ contains no backbone edge by definition, it is enough to show that u is not disconnected from v_i in $G_1 \setminus F_a$. As M_i is a support lemon, there exists a set $F_i \in \mathcal{F}_j$ such that $F_i \subseteq \mathcal{L}_i$. Consider a set $F_a \in \mathcal{F}_j$ with $F_a \subseteq \mathcal{L}_a$. If $a \leq i$, then $F_a \subseteq L_a \subseteq L_i$, so $F_a \cap M_i = \emptyset$. If $a \geq i$, then $F_a \cap L_{i+1} \subseteq F_i \subseteq L_i$ as $\mathcal{F}_j$ is correlated, hence $F_a \cap M_i = \emptyset$ holds as well. This completes the proof of Claim 36. $\square$

Consider a vertex u such that either u belongs to some cherry C_i or u belongs to a lemon M_i which is not a support lemon. An edge $v_a v_{a+1}$ in the support *affects* a half-request (u, x, v) if $a < i$ or if $i \leq a$ and the unique set $F_a \in \mathcal{F}_j$ such that $F_a \subseteq \mathcal{L}_a$ separates u from x in G_1. If

$v_a v_{a+1}$ does not affect (u, x, v), then neither does $v_b v_{b+1}$ when $b \geq a$. Indeed when $b \geq a$, $F_b \subseteq \mathcal{L}_b$ and $F_b \in \mathcal{F}_j$, we have that $F_b \cap L_a \subseteq F_a$.

Let us now modify the instance. If no edge of the support affects a half request (u, x, v), where either u belongs to some cherry C_i or u belongs to a lemon M_i which is not a support lemon, we remove (u, x, v) from R and add the half-request (x, x, v). Otherwise we let $v_a v_{a+1}$ be the support edge with a maximal which affects (u, x, v). We replace (u, x, v) in R by (v_{a+1}, x, v). We call this process *projecting* the half-request (u, x, v). After projecting all half-requests via x with an endpoint in a cherry or in a lemon M_i which is not a support lemon, we decrease f_i by ℓ and contract every edge of P_1 which is not in the support of $\mathcal{F}_j$. Note that, if $v_i v_{i+1}$ is not in the support, then there remains no half-request via x with an endpoint in M_i.

Assume that F' is a solution of this reduced instance which uses an edge $v_a v_{a+1}$ in the support. Let F_a be the element of $\mathcal{F}_j$ such that $F_a \subseteq \mathcal{L}_a$. Then $F' \cup F_a$ is a solution in the original instance. Indeed the half-requests with an endpoint in the support lemons are cut in $F' \cup F_a$ if and only if they are cut by F', as F_a does not intersect these lemons by Claim 36. Also, the half-requests with an endpoint in the lemons which are not support lemons or with an endpoint in the cherries are cut in the reduced instance if and only if they are cut by F_a in the initial instance by construction.

Conversely, assume that F is a proper solution in the original instance which uses the edge $v_a v_{a+1}$ and such that $F \cap L_a \in \mathcal{F}_j$. In particular, $F_a = F \cap L_a$, so $F \setminus F_a$ is a solution of the reduced instance. Indeed, all half-requests (u, x, v) cut by F_a in the original instance are affected by $v_a v_{a+1}$, hence they have been projected to (v_i, x, v) with $i \geq a + 1$, and they are cut by $F \setminus F_a$ in the reduced instance.

The reduction, consisting in projecting all half-requests with an endpoint in a cherry or in a lemon which is not a support lemon, improves the invariant unless $\ell = 0$, *i.e.* unless the proper solution of the original instance with backbone edge $v_i v_{i+1}$ does not use any edge in L_i. In this case, all the requests via x of cherry C_j are projected to v_j, for all j. By the same argument, we can assume that no edge in a proper solution is selected to the right of M_i and that the half-requests via y of C_j are projected to v_j. In this case, there remains no terminal in the cherries, so we simply contract the cherries. We are only left with lemons, and we moreover know that if a solution exists, then there exists a solution which uses only edges in a single lemon. This concludes the proof of Lemma 35. □

Theorem 37. *We can assume that G_1 only consists of the backbone P_1.*

Proof. We assume that $\lambda > 1$ and show that we can improve the invariant. Let us consider a backbone edge $v_i v_{i+1}$. We denote by W the multiset of vertices $\{w_2, \ldots, w_\lambda\}$ where w_j is the vertex of the slice S_i of v_i in M_i which belongs to the path P_j and has a neighbor in $M_i \setminus S_i$. In other words, w_j is the rightmost vertex of each path P_j in the slice of v_i. These vertices w_j are not necessarily distinct, for instance if v_i has degree λ in M_i, the slice S_i is exactly $\{v_i\}$ hence all w_j are equal to v_i for $j = 2, \ldots, \lambda$. We also denote by $Z = \{z_2, \ldots, z_\lambda\}$ the multiset of vertices of the slice T_i of v_{i+1} in M_i which belong respectively to the paths $P_2, \ldots, P_\lambda$ and have a neighbor in $M_i \setminus T_i$.

A multicut F induces a partition of $W \cup Z$ according to the components of $G \setminus F$. A vertex of $W \cup Z$ has three possible *types*: it can be in the same component as x after the removal of F, in the same component as y, or in another component. Observe that, if two vertices a, b of $W \cup Z$ belong to components distinct from the components of x and y in $G \setminus F$, then F is still a multicut after contracting a and b. Hence F induces a partition of W into three parts, each of which can be contracted, and F remains multicut. We now branch over all partitions of $W \cup Z$ into three parts WZ_x, WZ_y, WZ_u, where WZ_x are vertices which are in the same component as x, WZ_y are vertices which are in the same component as y, and WZ_u are vertices of the same type, possibly disconnected from x and y (but not necessarily so). We branch over all possible partitions of W into WZ_x, WZ_y, WZ_u, and contract in each branch WZ_x to v_i, WZ_y to v_{i+1}, and WZ_u (if not empty) is contracted to a single vertex called u_i. In each branch, these contractions are performed simultaneously in all lemons M_i. We denote by G'_1 the resulting component, by M'_i the contracted lemon M_i, and by S'_i the contracted S_i.

If some vertex of W belongs to WZ_y, or if some vertex of Z belongs to WZ_x, or if WZ_u intersects both W and Z, then the xy edge-connectivity increases in G'_1. Indeed, in all these cases, there exists an xy-path in G'_1 without edges of $\lambda(x,y)$-separator in G_1. This improves the invariant, but we cannot directly conclude since the edges of the backbone may not be critical any longer. Indeed, the connectivity between v_i and v_{i+1} in M'_i could be smaller than the connectivity of another lemon M'_j, in which case the backbone edge v_jv_{j+1} is not critical. To get a correct instance of Backbone Multicut, we simply branch on the connectivity of the lemon M'_i corresponding to the chosen edge v_iv_{i+1}. In the branch corresponding to connectivity ℓ, we contract the backbone edges v_iv_{i+1} where M'_i has connectivity distinct from ℓ.

Hence we can assume without loss of generality that W is partitioned into WZ_u and WZ_x, and that $Z = WZ_y$. As we contract WZ_y to v_{i+1}, the vertex v_{i+1} has now degree λ in M'_i, and T_i is a v_{i+1}-cherry. Let us assume that $WZ_u \neq \emptyset$. As we have contracted the vertices of W to v_i and u_i, the set S'_i has exactly two vertices with a neighbor in $M'_i \setminus S'_i$, namely v_i and u_i. Note that the degree of v_i in $M'_i \setminus S'_i$ is exactly the number of vertices w_j chosen in WZ_x (with multiplicity, since WZ_x is a multiset). We denote this degree of v_i in $M'_i \setminus S'_i$ by d. Note that d does not depend on i since we have chosen in every M_i the same subset WZ_x inside $\{w_2, \dots, w_\lambda\}$.

Let λ_S be the v_iu_i edge-connectivity in S'_i. If $\lambda_S > f_1$, then u_i and v_i cannot be separated, so we contract u_i and v_i. We branch in order to assume that λ_S is some fixed value. In the branch corresponding to connectivity λ_S, we contract backbone edges v_iv_{i+1} where S_i has connectivity distinct from λ_S. Let $P'_1, \dots, P'_{\lambda_S}$ be a collection of edge disjoint paths between u_i and v_i in S'_i. We denote by S' the slice of v_i in S'_i, and once again we consider the rightmost vertices $W' = \{w'_1, \dots, w'_{\lambda_S}\}$ of S' in the paths P'_j. We branch over all possible partitions of W' into W'_x, W'_y, W'_u. Once again, if W'_y is not empty, we increase the connectivity between x and y. Observe that W'_u can be contracted to WZ_u, hence to u_i. In particular if W'_u is not empty, we increase the connectivity between v_i and u_i in S'_i. We iterate this process in S'_i until either W'_u is empty in which case v_i has degree λ_S in S'_i, or λ_S exceeds f_1 in which case we contract v_i and u_i.

We apply Lemma 35 to G'_1. Therefore, we can assume that no cherries are left and that if a solution exists, one multicut is contained in some M'_i. Note also that Lemma 35 does not modify lemons which have not disappeared, in particular properties obtained in the previous paragraphs still hold. Two cases can happen:

If $f_1 \geq d + \lambda_S + \lambda - 1$, and the edge v_iv_{i+1} is chosen in the backbone, then we can assume that the restriction of the multicut to M'_i simply consists of all the edges incident to v_i and v_{i+1} in M'_i. Indeed v_i is incident to $d + \lambda_S$ free edges, and v_{i+1} is incident to $\lambda - 1$ free edges. This is clearly the best solution since it separates all vertices of $M'_i \setminus \{v_i, v_{i+1}\}$ from v_i and v_{i+1}. Therefore, we project every request (u, x, v) where $u \in M'_i$ to (v_{i+1}, x, v) and project every request (u, y, v) where $u \in M'_i$ to (v_i, y, v). Finally we reduce f_1 to 0 and we delete all vertices of G'_1 which are not in P_1.

Assume now that $f_1 < d + \lambda_S + \lambda - 1$. We branch over $2(\lambda - 1)$ choices, where the branches are named B_j and B'_j for all $j = 2, \dots, \lambda$. In the branch B_j, we assume that only one edge of the solution is selected in P_j, and that this edge is critical. In the branch B'_j, we assume that all the edges of the solution selected in P_j are not critical. In the branch B'_j, we contract critical edges of P_j and improve the invariant. In the branch B_j, we find a new backbone P_j. In this last case, we delete the edges of P_1 and reduce the number of free edges to $f_1 - 1$. We also translate the clauses in terms of edges of the new backbone P_j. Indeed the number of edges in the backbone of G_1 has changed. Clauses of the form $c_1 \leq i$ become $c_1 \leq \epsilon(i)$ where $\epsilon(i)$ denotes the index of the rightmost edge of P_j in the lemon M'_i.

This branching process covers all the cases where $v_i = u_i$ since in this case $f_1 < 2\lambda - 2$ and therefore one path P_j contains only one edge of the multicut. In the case $v_i \neq u_i$, assume that a multicut F is not of a type treated in one of the branches. In other words, F contains at least two edges in each path P_j for $j = 2, \dots, \lambda$, and at least one of them is critical. Then F contains two edges in each of the d paths P_j not containing u_i since F does not respect the branches B_j for $j = 2, \dots, \lambda$. Also, F contains one edge outside S'_i in each path P_j containing u_i since edges in S'_i are not critical and F is not treated in the branches B'_j. Thus F contains at least $2d + (\lambda - d - 1)$ free edges outside S'_i. Hence less than λ_S edges of F lie in S'_i, thus v_i and u_i belong to the same component in $G - F$. This case is covered in another branch in which v_i and u_i are contracted. Hence this branching process is exhaustive, and this completes the proof of Theorem 37. $\square$

5.7 Reducing to 2-SAT

We are left with instances in which the Y-components with two attachment vertices only consist of a backbone. We now reduce the last components.

Lemma 38. *We can assume that there is no component with one attachment vertex.*

Proof. Let $Y = \{y_1, \dots, y_p\}$ and let k be the number of free edges in the multicut. A vertex $y_i \in Y$ is *safe* if there is no request between two components attached only to y_i. If y_i is not safe then there is a request (u, y_i, v), with u and y in components attached to y_i, hence y_i must be either disconnected from u or disconnected from v by the solution. We explore one branch where u is added to Y, and one branch where v is added to Y. This creates a component with two attachment vertices. This component has a backbone, and the number of free edges decreases.

Hence, we can assume that all the vertices of Y are safe. The y_i-cherry is the union of all the components attached to y_i. We branch over all possible integer partitions of k into a sum $k_1 + k_2 + \cdots + k_p = k$. In each branch, we require that k_i edges are deleted in the y_i-cherry for $i = 1, \dots, p$. By Lemma 18, the y_i-cherry has a bounded active set $\mathcal{L}_i$, hence in the y_i-cherry we can consider only a bounded number of separators of size k_i: all subsets of $\mathcal{L}_i$ of size k_i. We branch over these different choices. In a given branch, we delete a particular set of edges F_i in the y_i-cherry. Thus, we delete the vertices of the y_i-cherry isolated from y_i by F_i, and contract the other vertices of the y_i-cherry to y_i. Finally, no Y-cherry remains. $\square$

Theorem 39. *Multicut is FPT.*

Proof. By Lemma 38, to prove that Backbone Multicut is FPT, we only have to deal with an input graph G which is a subdivision of a graph with at most k edges, and where a multicut must consist of exactly one edge in each subdivided path. Let us consider a half-request (v_i, x, v'_j). Assume without loss of generality that $v_i \in G_1$, $v'_j \in G_2$, and x belongs to G_1 and G_2 (if x does not belong to G_1 or G_2, then splitting Y automatically results in cutting the half-request (v_i, x, v'_j)). For simplicity, we assume that the edges of both P_1 and P_2 are enumerated in increasing order from x. We add to $\mathcal{C}$ the clauses $x_1 \geq i \Rightarrow x_2 \leq j - 1$ and $x_2 \geq j \Rightarrow x_1 \leq i - 1$. We transform all the half requests in this way. We are only left with a set of clauses which we have to satisfy.

We finally add all the relations $x_i \geq a \Rightarrow x_i \geq a - 1$ and $x_i \leq a \Rightarrow x_i \leq a + 1$ and $x_i \geq a \Rightarrow \neg(x_i \leq a - 1)$ and $x_i \leq a \Rightarrow \neg(x_i \geq a + 1)$ to ensure the coherence of a satisfying assignment. We now have a 2-SAT instance which is equivalent to the original multicut instance. As 2-SAT is solvable in polynomial time, this shows that Backbone Multicut is FPT. Hence the simpler Component Multicut problem is FPT. Together with Theorem 20 which reduces Multicut to Component Multicut, this concludes the proof of Theorem 39. $\square$

To sum up, the invariants in the proof that Backbone Multicut is FPT are (in decreasing order of importance):

- The total number of free edges in the multicut, to minimize, bounded by k.
- The sum of the free connectivity in each component, to maximize, bounded by k.
- The sum of the slice connectivity in each component, to minimize, bounded by k.[4]

[4]By "bounded" we mean bounded above, all the invariants described in this chapter being trivially non-negative. In both cases, maximize or minimize, the upper bound corresponds to the maximum number of times an invariant can be improved.

The algorithm starts by branching over the tag XY of the backbone edge chosen in the multicut in a given component. When $X \neq Y$, the invariant improves by Lemma 29. The dominant complexity term comes from the $\mathcal{O}(2^k)$ branches where $X = Y$. Tags may change, and another such branching is done, and again the dominant term comes from the 2^{2k} cases where $X = Y$. If the chosen edge is unstable, we improve the invariant with a factor two branching in Lemma 30. If the edge is stable, we branch over k choices to decide which part of a partition modulo $k+1$ does not intersect the solution F in Lemma31, and branch again over $k+1$ choices for the rank of the backbone edge chosen in the solution. This yields $\mathcal{O}(k^2 2^{2k})$ cases where all vertices are full. If a backbone vertex is not a cut-vertex, we increase the invariant by Lemma 32. Otherwise, we apply Theorem 37, which branches over $3^{\mathcal{O}(k)}$ cases, in which either the component consists only in its backbone, or the invariant has improved.

When the whole process described in the previous paragraph has been performed over all components with two attachment vertices, yielding $\mathcal{O}^*(k^{\mathcal{O}(k)})$ branches, we apply Lemma 38. If a vertex in Y is not safe, the invariant improves. When all vertices are safe, the tree which represents the branchings made by the algorithm thus far (where child nodes are instances with a better invariant) has depth at most k^3 and the degree of its nodes is bounded by $\mathcal{O}^*(k^{\mathcal{O}(k)})$. Hence the total number of leaves in the branching process thus far is $\mathcal{O}^*(k^{\mathcal{O}(k^4)})$.

Lemma 38 proceeds by branching over $\mathcal{O}(k^k)$ cases to fix the number of edges chosen by the solution in each component, and then applies Lemma 18, branching exhaustively over all subsets with the adequate number of active edges in each component. This gives a branching factor of $\mathcal{O}((|K| \cdot 4^k)^k)$, where K is the set of terminals in a given cherry, which is bounded by $\mathcal{O}(k^{\mathcal{O}(k^3)})$ by Theorem 17. The total number of branches obtained thus far is $\mathcal{O}(k^{\mathcal{O}(k^4)})$. Finally, Theorem 39 directly translates to a 2-SAT instance. Thus:

Observation 40. The FPT algorithm for Backbone Multicut runs in time $\mathcal{O}^*(k^{\mathcal{O}(k^4)})$.

6 Conclusion

6.1 A single-exponential algorithm

Our proof was originally designed only to prove that Multicut is FPT, with no particular focus on algorithmic efficiency. For the sake of completeness, we briefly analyzed the complexity after each of the two major parts of the proof. Multicut is reduced to Component Multicut in time $\mathcal{O}^*(k^{\mathcal{O}(k^6)})$ by Remark 27, and Backbone Multicut is solved in time $\mathcal{O}^*(k^{\mathcal{O}(k^4)})$ by Remark 40. Hence the overall running time of the algorithm is $\mathcal{O}^*(k^{\mathcal{O}(k^6)})$. This algorithm and its complexity analysis are definitely not fine tuned, and the running time could probably be improved. Finding a $2^{\mathcal{O}(k \log k)}$-running time algorithm would reveal a widely better comprehension of the Multicut problem.

6.2 Other leads

Considering finer concepts than the notion of request can also be envisioned. In the simpler case of Multicut In Trees, a request is simply a path. In general graphs, a request can be seen as the set of paths between its endpoints. In our proof, we simulate requests by half-requests, partitioning the set of paths naturally associated to a request according to an intermediate point. This could be done thanks to the vertex-multicut Y. But we originally wanted to go much further, and consider the more general problem of cutting a prescribed set of paths, not necessarily all paths between given pairs of vertices. The obvious problem is that a request can be realized by exponentially many paths (exponentially many in n), but we loosely conjectured that this difficulty can be avoided as follows:

Problem 41. Given a graph G on n vertices, an integer k and two vertices u, v of G, does there always exist a set S of at most $f(k) \cdot Poly(n)$ paths between u and v such that removing at most

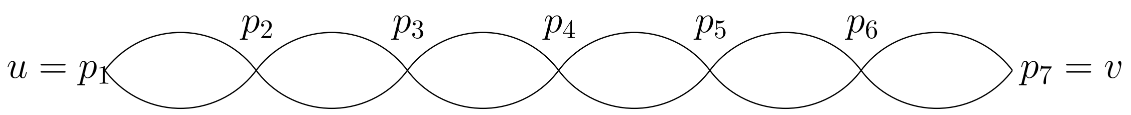

Figure 10: A double path on seven vertices.

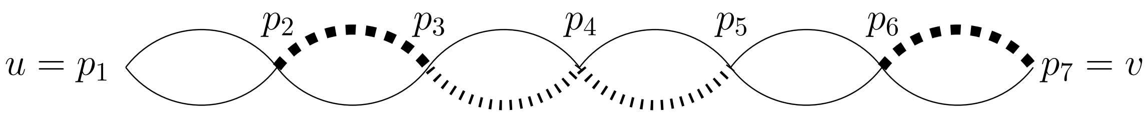

Figure 11: Assuming $k = 4$, one constraint would be generated by $T = \{2, 3, 4, 6\}$ partitioned into $\overline{T} = \{2, 6\}$ (corresponding to the dashed edges) in and $\underline{T} = \{3, 4\}$ (corresponding to the dotted edges). A solution must contain a uv-path avoiding all dotted edges and dashed edges.

k edges of G to disconnect all paths in S must actually disconnect u and v? In other words, can an FPT number of paths simulate a request with respect to k-multicuts?

Consider for starters the multigraph G consisting of a path on n vertices with endpoints $u = p_1$ and $v = p_n$, where each edge has been duplicated (into a 2-cycle), as in Figure 10. We are looking for a small set of uv-paths, such that every hitting set of these paths is a uv-edge-separator.

Given a (simple) uv-path P and an integer $i \in \{1, \ldots, n-1\}$, let us say that P *takes* i if P contains the top edge in *position* i, *i.e.* between p_i and p_{i+1}. We can reformulate the constraint on the solution set S as follows: for every set T of k positions and for every bipartition of T into $\overline{T}$ and $\underline{T}$, there must exist a path in S which takes all positions in $\overline{T}$ and takes no position in $\underline{T}$. Indeed, if this is not the case, then the bottom edges for positions in $\overline{T}$ and the top edges for positions in $\underline{T}$ form a set of k edges which hits S but does not cut u from v. See Figure 11 for an example.

Hence every bipartitioned set of k positions gives a constraint. There are $\binom{n}{k}2^k$ such constraints and a given simple uv-path satisfies (in the above sense) $\binom{n}{k}$ such constraints. Hence a random simple path satisfies a fraction $\frac{1}{2^k}$ of the constraints. In particular, there exists one path P which satisfies at least a fraction $\frac{1}{2^k}$ of the constraints. We pick the path P in our solution S and repeat. This process (not taking into account the computability of P) finds a solution of size at most $log_{2^k}(\binom{n}{k}2^k)$ paths, which is actually even better than needed, with a logarithmic dependence on n rather than a polynomial dependence.

Problem 42. If Conjecture 41 has a positive answer, can such an FPT set of paths emulating the request uv be computed in FPT time?

In the simple example worked out above, the randomized process should easily be derandomizable into an FPT algorithm.

If Conjecture 41 and Conjecture 42 turn out to have positive answers, then MULTICUT can be reduced to the following problem:

HITTING PATH:
Input: A graph G, a set R of paths in G, an integer k.
Parameter: k.
Output: TRUE if there is a set at most k edges of G which hits R, otherwise FALSE.

Problem 43. Can HITTING PATH be solved in FPT time?

It is not clear a priori whether this problem should be easier or harder than MULTICUT. Directly emulating a MULTICUT instance with HITTING PATH would require an exponential number of paths, and conversely the structure of the objects to be hit can be more complicated in HITTING PATH than in MULTICUT.

We did not pursue this insight further when the ideas exposed in this chapter proved to be fruitful, but Problems 41, 42 and 43 remain very interesting nonetheless, on their own right as well as with respect to MULTICUT.

References

[1] H. Bodlaender, L. Cai, J. Chen, M. Fellows, J.A. Telle, and D. Marx. Open problems in parameterized and exact computation. In *Proceedings of the 2nd International Workshop on Parameterized and Exact Computation*, IWPEC 2006, 2006.

[2] N. Bousquet, J. Daligault, and S. Thomassé. Multicut is FPT. In *STOC*, pages 459–468, 2011.

[3] N. Bousquet, J. Daligault, S. Thomassé, and A. Yeo. A polynomial kernel for multicut in trees. In *STACS*, pages 183–194, 2009.

[4] L. Brunetta, M. Conforti, and M. Fischetti. A polyhedral approach to an integer multicommodity flow problem. *Discrete Applied Mathematics*, 101(1-3):13 – 36, 2000.

[5] S. Chawla, R. Krauthgamer, R. Kumar, Y. Rabani, and D. Sivakumar. On the hardness of approximating multicut and sparsest-cut. *Comput. Complex.*, 15(2):94–114, 2006.

[6] J. Chen, J. Fan, I. Kanj, Y. Liu, and F. Zhang. Multicut in trees viewed through the eyes of vertex cover. *J. Comput. Syst. Sci.*, 78(5):1637–1650, September 2012.

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

[8] J. Chen, Y. Liu, S. Lu, B. O'Sullivan, and I. Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, STOC '08, pages 177–186, New York, NY, USA, 2008. ACM.

[9] R. Chitnis, M. Hajiaghayi, and D. Marx. Fixed-parameter tractability of directed multiway cut parameterized by the size of the cutset. In *SODA*, pages 1713–1725, 2012.

[10] M.-C. Costa, L. Létocart, and F. Roupin. Minimal multicut and maximal integer multiflow: A survey. *European Journal of Operational Research*, 162(1):55–69, 2005.

[11] M. Cygan, S. Kratsch, M. Pilipczuk, M. Pilipczuk, and M. Wahlström. Clique cover and graph separation: New incompressibility results. In *ICALP (1)*, pages 254–265, 2012.

[12] M. Cygan, M. Pilipczuk, M. Pilipczuk, and J. Wojtaszczyk. On multiway cut parameterized above lower bounds. *TOCT*, 5(1):3, 2013.

[13] J. Daligault, C. Paul, A. Perez, and S. Thomassé. Treewidth reduction for the parameterized multicut problem. Technical report, 2008.

[14] R.G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer, 1999.

[15] P. Erdős and R. Rado. Intersection theorems for systems of sets. *Journal of the London Mathematical Society*, 35:85–90, 1960.

[16] J. Flum and M. Grohe. *Parameterized Complexity Theory.* Text in Theoretical Computer Science. Springer, 2006.

[17] N. Garg, V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. pages 698–707, 1993.

[18] N. Garg, V. Vazirani, and M. Yannakakis. Multiway cuts in directed and node weighted graphs. In *Automata, Languages and Programming*, volume 820 of *Lecture Notes in Computer Science*, pages 487–498, 1994.

[19] N. Garg, V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1):3–20, 1997.

[20] G. Gottlob and S. Tien Lee. A logical approach to multicut problems. *Information Processing Letters*, 103(4):136–141, 2007.

[21] S. Guillemot. FPT algorithms for path-transversals and cycle-transversals problems in graphs. In *IWPEC*, pages 129–140, 2008.

[22] J. Guo, F. Hüffner, E. Kenar, R. Niedermeier, and J. Uhlmann. Complexity and exact algorithms for multicut. In *SOFSEM*, pages 303–312, 2006.

[23] J. Guo and R. Niedermeier. Fixed-parameter tractability and data reduction for multicut in trees. *Networks*, 46(3):124–135, 2005.

[24] S. Kratsch, M. Pilipczuk, M. Pilipczuk, and M. Wahlström. Fixed-parameter tractability of multicut in directed acyclic graphs. In *ICALP (1)*, pages 581–593, 2012.

[25] Stefan Kratsch and Magnus Wahlström. Representative sets and irrelevant vertices: New tools for kernelization. In *FOCS*, pages 450–459, 2012.

[26] D. Marx. Parameterized graph separation problems. *Theor. Comput. Sci.*, 351(3):394–406, 2006.

[27] D. Marx, B. O'Sullivan, and I. Razgon. Treewidth reduction for constrained separation and bipartization problems. In *STACS*, 2010.

[28] D. Marx, B. O'Sullivan, and I. Razgon. Finding small separators in linear time via treewidth reduction. *CoRR*, abs/1110.4765, 2011.

[29] D. Marx and I. Razgon. Constant ratio fixed-parameter approximation of the edge multicut problem. In *ESA*, volume 5757 of *Lecture Notes in Computer Science*, pages 647–658. Springer, 2009.

[30] D. Marx and I. Razgon. Fixed-parameter tractability of multicut parameterized by the size of the cutset. In *STOC*, pages 469–478, 2011.

[31] R. Niedermeier. *Invitation to Fixed Parameter Algorithms (Oxford Lecture Series in Mathematics and Its Applications).* Oxford University Press, USA, March 2006.

[32] N. Robertson and P.D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Combin. Theory, Ser. B*, 63(1):65–110, 1995.