Length-Bounded Cuts: Proper Interval Graphs and Structural Parameters

Matthias Bentert
Algorithmics and Computational Complexity, Faculty IV, TU Berlin, Germany
matthias.bentert@tu-berlin.de

Klaus Heeger
Algorithmics and Computational Complexity, Faculty IV, TU Berlin, Germany
heeger@tu-berlin.de

Dušan Knop
Faculty of Information Technology, Czech Technical University in Prague, Czech Republic
dusan.knop@fit.cvut.cz

Abstract

In the presented paper, we study the Length-Bounded Cut problem for special graph classes as well as from a parameterized-complexity viewpoint. Here, we are given a graph G, two vertices s and t, and positive integers β and λ. The task is to find a set F of edges of size at most β such that every s-t-path of length at most λ in G contains some edge in F.

Bazgan et al. [Networks, 2019] conjectured that Length-Bounded Cut admits a polynomial-time algorithm if the input graph G is a proper interval graph. We confirm this conjecture by providing a dynamic-programming based polynomial-time algorithm. Moreover, we strengthen the W[1]-hardness result of Dvořák and Knop [Algorithmica, 2018] for Length-Bounded Cut parameterized by pathwidth. Our reduction is shorter, and the target of the reduction has stronger structural properties. Consequently, we give W[1]-hardness for the combined parameter pathwidth and maximum degree of the input graph. Finally, we prove that Length-Bounded Cut is W[1]-hard for the feedback vertex number. Both our hardness results complement known XP algorithms.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Dynamic programming; Mathematics of computing → Graph algorithms; Mathematics of computing → Combinatorial optimization

Keywords and phrases Edge-disjoint paths, pathwidth, feedback vertex number

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2020.36

Related Version A full version of the paper is available at `https://arxiv.org/abs/1910.03409`.

Funding *Klaus Heeger*: Supported by DFG Research Training Group 2434 "Facets of Complexity". *Dušan Knop*: Supported by the OP VVV MEYS funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics". Part of the work was done while affiliated with TU Berlin and supported by DFG under project "MaMu", NI 369/19.

1 Introduction

The study of network flows and, in particular, the Edge Disjoint Paths (EDP) problem began in the 1950s with the work of Ford and Fulkerson [11] and has constituted a prominent research subarea in graph algorithms since then. In the EDP problem, we are given a graph G, two vertices s and t, called *source* and *target*, and a positive integer β. The question is whether there is a collection of at least β edge-disjoint s-t-paths in G. It is worth pointing out that nowadays there are many more efficient algorithms (than the one of Ford and Fulkerson [11]) for finding a maximum flow in a given graph (see e.g. [8, 20]). A natural counterpart of EDP is the Edge Cut problem, where the task is to resolve whether there is a set of at most β edges F such that there is no s-t-path in the graph $G - F$. There is a strong

dual relationship between EDP and Edge Cut in the sense that, if both problems admit a solution for a given β, then the value of β is optimal, that is, it is not possible to find $\beta + 1$ edge disjoint s-t-paths and the removal of any set of $\beta - 1$ edges leaves s and t in the same connected component. Consequently, both problems admit an efficient (polynomial-time) algorithm, since one can also construct the set of cut edges from a maximum flow. Quite naturally, there are many variants of the above described network flow/cut problems such as e.g. multicommodity flows/cuts, unsplittable flows and the related cut problem (see e.g. [21] for further examples and exact definitions). Unlike the basic variant of EDP and Edge Cut, it is not always the case that the flow and the cut belong to the same complexity class. As we shall see, Length-Bounded Cut is in fact harder than the respective flow problem.

In this paper, we continue the study of the so-called Length-Bounded Cut problem, which is the cut problem related to the variant of EDP where an additional bound λ is given and the sought collection of s-t-paths can only contain paths with at most λ edges. To the best of our knowledge, this problem has been introduced by Adámek and Koubek [1] and the Length-bounded Cut problem is formally defined as follows.

Length-Bounded Cut

Input:	An undirected graph $G = (V, E)$, two vertices s, t, and two positive integers β, λ.
Question:	Is there a subset $F \subseteq E$ with $\lvert F\rvert \leq \beta$ such that there is no s-t-path of length at most λ in $G - F$?

If in the above definition one plugs in $\lambda = |V|$, then one is left with the Edge Cut problem; a polynomial-time-solvable problem. Length-Bounded Cut is also solvable in polynomial time if $\lambda \leq 3$ [19]. However, Baier et al. [2] showed that the Length-Bounded Cut problem is NP-hard already for $\lambda = 4$. On the other hand, the related Length-Bounded Flow problem, where we restrict the flow to paths of length at most λ, can be solved in polynomial time via a reduction to linear programming [2, 19, 18].

Length-Bounded Cut originated in telecommunications area and network design. See e.g. the work of Huygens et al. [14, 15] (and references therein) for possible applications; for further applications see e.g. the work of Gouveia et al. [13].

Before we give an overview of our results, we discuss the related work focusing on parameterized algorithms and algorithms for special graph classes.

1.1 Related Work

The result of Baier et al. [2] in fact gives NP-hardness for Length-Bounded Cut for each constant $\lambda \geq 4$. Thus, in order to obtain tractability results, one has to either consider a different parameterization or combine λ with some other parameter. The first study of Length-Bounded Cut from the viewpoint of parameterized complexity was done by Golovach and Thilikos [12]. They showed that Length-Bounded Cut is in FPT for the combined parameter $\beta + \lambda$. It is worth noting that parameterization only by β leads to W[1]-hardness [12]. Later, Fluschnik et al. [10] proved that it is unlikely that a polynomial kernel in $\beta + \lambda$ exists. Dvořák and Knop [9] considered structural parameters for the Length-Bounded Cut problem. They showed that it is W[1]-hard when parameterized by the pathwidth of the input graph while it is fixed-parameter tractable when parameterized by the treedepth of the input graph. It is worth pointing out that Length-Bounded Cut is one of just a few problems with such a parameterized dichotomy. Kolman [17] gave an $O(\lambda^\tau \cdot |G|)$-time algorithm for Length-Bounded Cut, where τ is the treewidth of G. Furthermore, Length-Bounded Cut is in FPT for the parameter λ if G is planar [17] (it remains

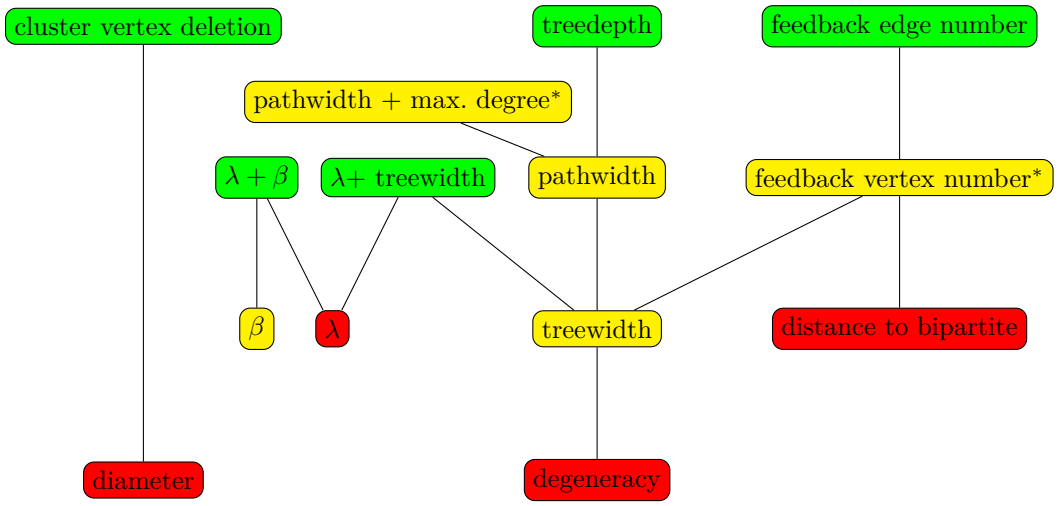

Figure 1 Known parameterized complexity results for Length-Bounded Cut. An edge between two parameters A and B (where A is above B) indicates that B is upper-bounded by A, that is, there is a computable function f such that $f(A(x)) \geq B(x)$ for each instance x. A red box indicated para-NP-hardness (NP-hardness for constant parameter values), a yellow box indicates W[1]-hardness, and a green box indicates containment in FPT. Except for feedback vertex number, all W[1]-hardness results are complemented by known XP-algorithms. Our results are marked by * and proven in Theorems 2 and 10.

NP-complete even in this case [10]). Bazgan et al. [3] studied both restrictions on special graph classes as well as structural parameterizations for Length-Bounded Cut. They provided an XP algorithm for the maximum degree of the input graph G and FPT algorithms for the parameters feedback edge number and vertex deletion distance to cluster graphs. Furthermore, they showed para-NP-hardness for the parameters diameter, degeneracy, and vertex deletion distance to bipartite graphs. An overview over the parameterized complexity landscape of Length-Bounded Cut is given in Figure 1. Furthermore, Bazgan et al. [3] present a polynomial-time algorithm for co-graphs while showing NP-completeness even if the input is restricted to bipartite graphs or split graphs.

1.2 Our Contribution

In this paper, we mainly continue the study of Length-Bounded Cut for special graph classes and from the viewpoint of parameterized complexity. Bazgan et al. [3] conjectured that Length-Bounded Cut can be solved in polynomial time on proper interval graphs which we confirm here: We give a dynamic-programming based algorithm in Theorem 13.

We show that the Length-Bounded Cut problem is W[1]-hard for the feedback vertex number in Theorem 10; thus, closing one of the gaps left by Bazgan et al. [3]. Furthermore, together with the result of Bazgan et al. [3] this yields a structural parameter dichotomy, since they provided an FPT algorithm for the feedback edge number.

Last but not least, we show in Theorem 2 that Length-Bounded Cut is W[1]-hard for the combined parameter pathwidth and maximum degree of the input graph G. This is a nontrivial strengthening of the reduction provided by Dvořák and Knop [9], where the degree cannot be bounded by a function of the parameter. Furthermore, our reduction implies that assuming ETH, there is no $f(k) \cdot n^{o(k)}$-time algorithm for Length-Bounded Cut, where k is pathwidth of the input graph (whereas the reduction of Dvořák and Knop

refutes only $f(k) \cdot n^{\sqrt{k}}$-time algorithms). This implies that the algorithm of Kolman [17] with running-time n^{k+1} is (nearly) optimal (such a bound can be obtained using the trivial bound $\lambda \leq n$). Moreover, this hardness result constitutes a natural counterpart of the known XP algorithm for the parameter maximum degree [3].

Proofs omitted due to space restrictions can be found in the full version [4].

1.3 Preliminaries

For a given positive integer a, we use $[a]$ to denote the set $\{1, 2, \ldots, a\}$. We use standard graph notation. Given a graph $G = (V, E)$, a source s, and a target t, a *λ-cut* is a set F of edges such that a shortest s-t-path in $G - F$ is of length at least $\lambda + 1$. We identify specific paths by just some of their vertices, e.g. we denote by *a-b-c-path* a path that starts in a, then continues by some shortest a-b-path and ends with some shortest b-c path. The shortest paths between two consecutive vertices in our identifiers (a-b and b-c in our example) will always be unique (that is, there is exactly one shortest path between a and b, and exactly one between b and c). We use $G[X]$ to denote the subgraph of G *induced* by a set X of vertices in a graph G and $G - X$ to denote $G[V \setminus X]$. An *interval graph* is a graph $G = (V, E)$ such that each vertex v can be represented by an interval $[b_v, f_v]$ such that two vertices u, w are adjacent in G if and only if $[b_u, f_u] \cap [b_w, f_w] \neq \emptyset$. If a graph admits such a representation fulfilling additionally that there are no two vertices v and w such that $[b_v, f_v] \subseteq [b_w, f_w]$, then this graph is a *proper interval graph*. Equivalently, a proper interval graph can be defined as an interval graph where each interval has length 1, i.e., $b_v + 1 = f_v$ for each vertex v (see e.g. [5]).

A *parameterized problem* is a tuple (L, κ), where L is a language (in our case Length-Bounded Cut) and κ is a *parameter*.

A parameterized problem (L, κ) is *fixed-parameter tractable* (or FPT for short) if there is an algorithm deciding each instance of it in $f(k) \cdot \text{poly}(n)$ time, where f is some computable function, k is the the value of the parameter κ of the input instance, and n is the input size. To show that some parameterized problem is *presumably not* FPT, one regularly uses the standard complexity assumption that FPT $\neq$ W[1] and shows that a problem is *W[1]-hard*. To show W[1]-hardness for some problem P with respect to some parameter κ, we use many-one reductions which ensure that the parameter of the output problem is bounded by a function of the parameter of the input problem. In this work we consider the parameters pathwidth, maximum degree and feedback vertex number. The *pathwidth* of a graph is closely related to the treewidth. The only difference between the two concepts is that a path decomposition is restricted to a collection of paths as underlying graphs for the bags as opposed to forests for tree decompositions. Formally, it is defined as follows.

▶ **Definition 1.** *A* path decomposition *of a graph $G = (V, E)$ consists of a path $P = (W, F)$ and a function $\pi \colon W \to 2^V$ such that*

- $\bigcup_{x \in W} \pi(x) = V$,
- *for each edge $\{v, w\} \in E$, there exists an $x \in W$ such that $v, w \in \pi(x)$, and*
- *for each $v \in V$, we have that $\{x \in W \mid v \in \pi(x)\}$ induces a connected subgraph in P.*

The width of a path decomposition *is* $\max_{x \in W} |\pi(x)| - 1$.
The pathwidth *of a graph G is the minimum width of a path decomposition of G.*

The *maximum degree* of a graph is the maximum number of incident edges to any single vertex in the graph. The *feedback vertex number* is the size of a minimum feedback vertex set, i.e., the minimum number of vertices one needs to delete from the graph to obtain a forest.

The Exponential-Time Hypothesis (ETH) of Impagliazzo and Paturi [16] asserts that there is no $2^{o(m)}$ algorithm solving the Satisfiability problem, where m is the number of clauses. It is worth noting that assuming ETH, there is no $f(k) \cdot n^{o(k)}$ time algorithm solving k-(Multicolored) Clique [6], where f is a computable function, n the number of vertices and k is the size of the clique we are looking for. For further notions related to parameterized complexity and ETH, we refer the reader to the textbook by Cygan et al. [7].

2 W[1]-Hardness for Pathwidth and Maximum Degree

In this section, we prove that Length-Bounded Cut is W[1]-hard with respect to the combined parameter pathwidth plus maximum degree. We describe our reduction from the W[1]-hard Clique problem (i.e., the problem of deciding whether a given graph contains a clique of size k) parameterized by solution size k.

Let $(G = (V, E), k)$ be an instance of Clique parameterized by solution size k and $n := |V|$. Let $\mathsf{index}\colon V \to [n]$ be a bijection that assigns each vertices in V a number from $\{1, 2, \dots, n\}$ and let $\mathsf{vertex}\colon [n] \to V$ be the inverse of index. We order the edges $E = \{e_1, e_2, \dots, e_m\}$ lexicographically by their endpoints, that is, for $e_i = \{v_i, w_i\}$ and $e_{i+1} = \{v_{i+1}, w_{i+1}\}$ with $v_j < w_j$ for $j \in \{i, i+1\}$, we have either $\mathsf{index}(v_i) < \mathsf{index}(v_{i+1})$, or $\mathsf{index}(v_i) = \mathsf{index}(v_{i+1})$ and $\mathsf{index}(w_i) < \mathsf{index}(w_{i+1})$. We assume without loss of generality that $m \geq n$ as we can otherwise remove all connected components which are trees (possibly returning true if $k \leq 2$).

Let $\eta := 4m$. We construct a Length-Bounded Cut instance $(H, s, t, \beta, \lambda)$ as follows. We set the budget β of edges to delete to $2k + 2k(k-1) = 2k^2$ and the length $\lambda := 8\eta + 2n + 1$. The graph H will consist of vertex-selection gadgets, incidence-checking gadgets, connectivity paths, and the vertices s and t, which are not contained in any gadget. We describe the gadgets now; the full proof is deferred to the full version [4] due to space constraints.

▶ **Theorem 2.** *Length-Bounded Cut parameterized by the combined parameter pathwidth plus maximum degree is W[1]-hard. Assuming the ETH, Length-Bounded Cut cannot be solved in $f(\sqrt{\Delta(G)} + \mathrm{pw}(G)) \cdot n^{o(\sqrt{\Delta(G)} + \mathrm{pw}(G))}$ time for any computable function f, where $\mathrm{pw}(G)$ is the pathwidth of G and $\Delta(G)$ is the maximum degree of G.*

Vertex-Selection Gadgets (see Figure 2). Our reduction will produce k vertex-selection gadgets $A_1, \dots, A_k$. The gadget A_j looks as follows:

We start with two paths $u_0^j, \dots, u_n^j$ and $\ell_0^j, \dots, \ell_n^j$ of length n (an "upper" and a "lower" path). We add a u_{p-1}^j-u_p^j-path U_p^j of length $2\eta + p$, an ℓ_{p-1}^j-ℓ_p^j-path L_p^j of length $2\eta - p$, and a u_p^j-ℓ_p^j-path of length 2η for every $p \in [n]$. Finally, we add two paths of length $\eta + 2$ between the source s and the first vertex ℓ_0^j of the lower path and two paths of length two between s and the first vertex u_0^j of the upper path.

The main idea of this gadget is that if exactly two edges $\{u_{p-1}^j, u_p^j\}$ and $\{\ell_{p-1}^j, \ell_p^j\}$ (both with the same $p \in [n]$) are removed, then the distance between s and u_n^j and between s and ℓ_n^j encodes the selected vertex.

Incidence-Checking Gadgets (see Figure 3). For each pair (A_i, A_j) with $i < j$ of vertex-selection gadgets, we add an incidence-checking gadget $I^{i,j}$. This gadget verifies that the two vertices selected by A_i and A_j share an edge.

Starting at u_n^i and ℓ_n^i, we add a gadget similar to the vertex-selection gadget. More precisely, we have two paths $a_0^{i,j}, \dots, a_n^{i,j}$ and $b_0^{i,j}, \dots, b_n^{i,j}$. The vertex u_n^i is connected to $a_0^{i,j}$ by two paths of length 4η. The vertex ℓ_n^i is connected to $b_0^{i,j}$ by two paths of length 2. For each $p \in \{0, 1, \dots, n\}$, there is an $a_p^{i,j}$-$b_p^{i,j}$-path of length 4η, an $a_{p-1}^{i,j}$-$a_p^{i,j}$-path of length $2\eta - p$

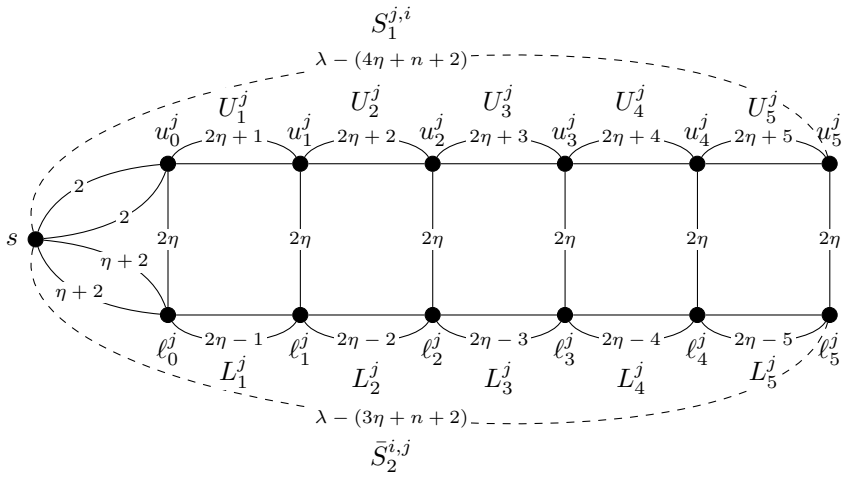

Figure 2 An example of a vertex-selection gadget with $n = 5$. An edge with a number x on it corresponds to a path of length x. The connectivity paths $S_1^{j,i}$ and $\bar{S}_2^{j,i}$ are dashed (and all other connectivity paths are not drawn for the simplicity of the picture).

and a $b_{p-1}^{i,j}$-$b_p^{i,j}$-path of length $2\eta + p$. Furthermore, there are two $a_n^{i,j}$-t-paths of length two, and two $b_n^{i,j}$-t-paths of length 3η. Next we add two paths $c_0^{i,j}, \ldots, c_m^{i,j}$ and $d_0^{i,j}, \ldots, d_m^{i,j}$ of length m. The vertex $c_0^{i,j}$ is connected to u_n^j by two parallel paths of length 3η. The vertex $d_0^{i,j}$ is connected to ℓ_n^j by two paths of length η. For each $p \in [m]$, let $e_p = \{v_p, w_p\}$ with $\mathsf{index}(v_p) < \mathsf{index}(w_p)$, there is an $c_p^{i,j}$-$d_p^{i,j}$-path of length 2η, and an $c_{p-1}^{i,j}$-$c_p^{i,j}$-path of length $2\eta - \mathsf{index}(w_p)$, and a $d_{p-1}^{i,j}$-$d_p^{i,j}$-path of length $2\eta + \mathsf{index}(w_p)$. Furthermore, there are two $c_n^{i,j}$-t-paths of length $\eta + n - m + 2$, and two $d_n^{i,j}$-t-paths of length $2\eta + n - m + 2$.

For each $p \in \{0, 1, \ldots, n-1\}$, let q be zero if $p = 0$, and otherwise maximum such that e_q is incident to $\mathsf{vertex}(p)$. We add an $a_p^{i,j}$-$c_q^{i,j}$-path of length 2η, an $a_p^{i,j}$-$d_q^{i,j}$-path of length 3η, a $b_p^{i,j}$-$c_q^{i,j}$-path of length 3η and a $b_p^{i,j}$-$d_q^{i,j}$-path of length 2η.

Assuming that the vertices v_i and v_j have been selected in the vertex-selection gadgets A_i and A_j, respectively, there is only one possible way to cut all s-t-paths through $I^{i,j}$ of length at most λ by adding at most four edges from $I^{i,j}$ to the cut. The respective edges are $\{a^{i,j}_{\mathsf{index}(v_i)-1}, a^{i,j}_{\mathsf{index}(v_i)}\}$, $\{b^{i,j}_{\mathsf{index}(v_i)-1}, b^{i,j}_{\mathsf{index}(v_i)}\}$, $\{c_{p-1}^{i,j}, c_p^{i,j}\}$, and $\{d_{p-1}^{i,j}, d_p^{i,j}\}$ for $e_p = \{v_i, v_j\}$. In particular, if v_i and v_j are not adjacent, then any λ-cut must contain at least five edges from $I^{i,j}$.

Connectivity Paths. For each vertex u_n^i of a vertex-selection gadget A_i, we add three parallel paths T_1^i, T_2^i, and T_3^i of length $\lambda - (n+2)$ to t. Similarly, we add for each $i \in [k]$ three parallel paths $\bar{T}_1^i$, $\bar{T}_2^i$, and $\bar{T}_3^i$ from vertex ℓ_n^i, of length $\lambda - (\eta + 2 + n)$ to t (see Figure 3). For each $(i,j) \in [n]^2$, we add five parallel paths $S_q^{i,j}$ from s to u_n^i of length $\lambda - (4\eta + n + 2)$, and five parallel paths $\bar{S}_q^{i,j}$ from s to ℓ_n^i of length $\lambda - (3\eta + n + 2)$ (see Figure 2).

These connectivity paths ensure that there is no solution in which less than two edges from a vertex-selection gadget or less than four edges from an edge gadget are removed. This together with the budget β implies that every solution removes exactly two edges from each vertex-selection gadget and exactly four edges from each incidence-checking gadget.

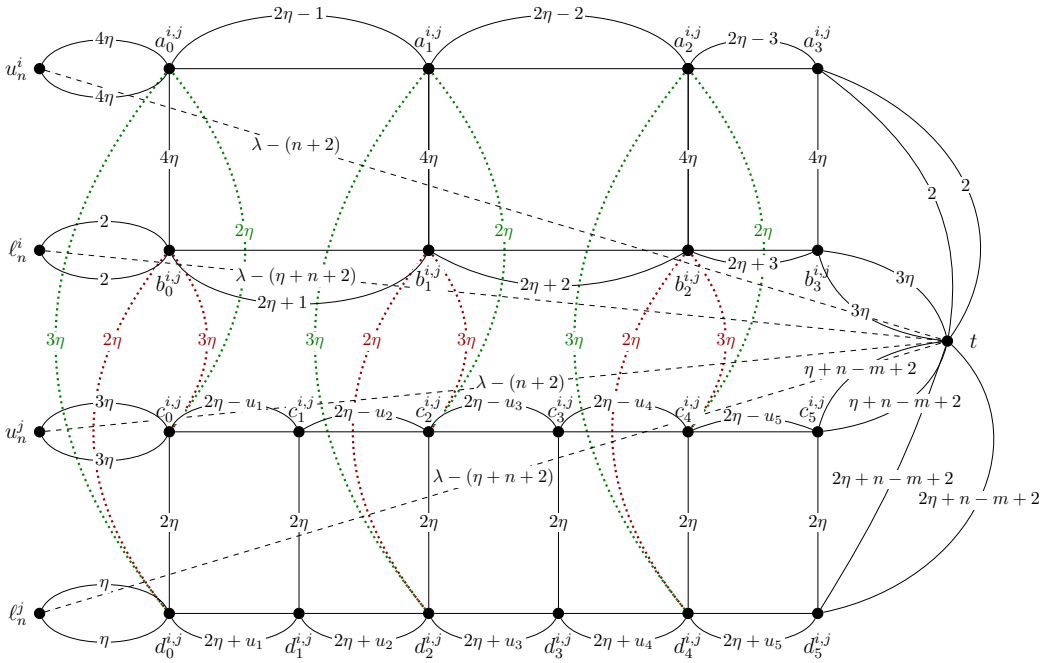

Figure 3 An example of an incidence-checking gadget $I^{i,j}$. An edge with a number x on it corresponds to a path of length x. Paths between a, b-vertices and c, d-vertices are dotted and colored to be better distinguishable from other paths. Connectivity paths T_1^i, T_1^j, $\bar{T}_1^i$, and $\bar{T}_1^j$ are dashed and connectivity paths T_2^i, T_3^i, T_2^j, T_3^j, $\bar{T}_2^i$, $\bar{T}_3^i$, $\bar{T}_2^j$, and $\bar{T}_3^j$ are not shown for the sake of readability.

3 W[1]-Hardness for Feedback Vertex Number

In this section, we prove that Length-Bounded Cut is W[1]-hard with respect to the feedback vertex number. We present a parameterized reduction from Multicolored Clique parameterized by solution size. In the Multicolored Clique problem, the input consists of a k-partite graph $G = (V_1 \uplus V_2 \uplus \ldots \uplus V_k, E)$ and an integer k, and the question is whether G contains a clique of size k.

After we present the reduction, we show its correctness and finally analyze its running time and the size of the feedback vertex number in the resulting instance.

3.1 The Reduction

We describe our reduction for a given instance $(G = (V, E), k)$ of Multicolored Clique and call the resulting graph $H = (V_H, E_H)$. Let $V = V_1 \uplus V_2 \uplus \ldots \uplus V_k$ be the k-partition of G. We assume that all V_i contain the same number of vertices; this can be achieved by adding isolated vertices. We define $\nu := |V_i|$. We assume that the vertices of each V_i are numbered from 1 to ν, and that this numbering is given via a function $\mathsf{index}\colon V \to [\nu]$ and for each $i \in [k]$, a function $\mathsf{vertex}_i\colon [\nu] \to V_i$ such that $\mathsf{vertex}_i(\mathsf{index}(v)) = v$ for every $v \in V_i$ and $\mathsf{index}(\mathsf{vertex}_i(x)) = x$ for all $x \in [\nu]$.

We start by adding for each V_i a specific vertex-selection gadget that is explained below. Second, we add for each $i \in [k]$, each $j \in [k] \setminus \{i\}$, and each edge between V_i and V_j an edge gadget (which is also described below). We then set $\lambda := \nu + 2n$ and $\beta := 2k(\nu-1)m + m - \binom{k}{2}$, where n and m denote the number of vertices and the number of edges of G, respectively. We remark that $n = k\nu$.

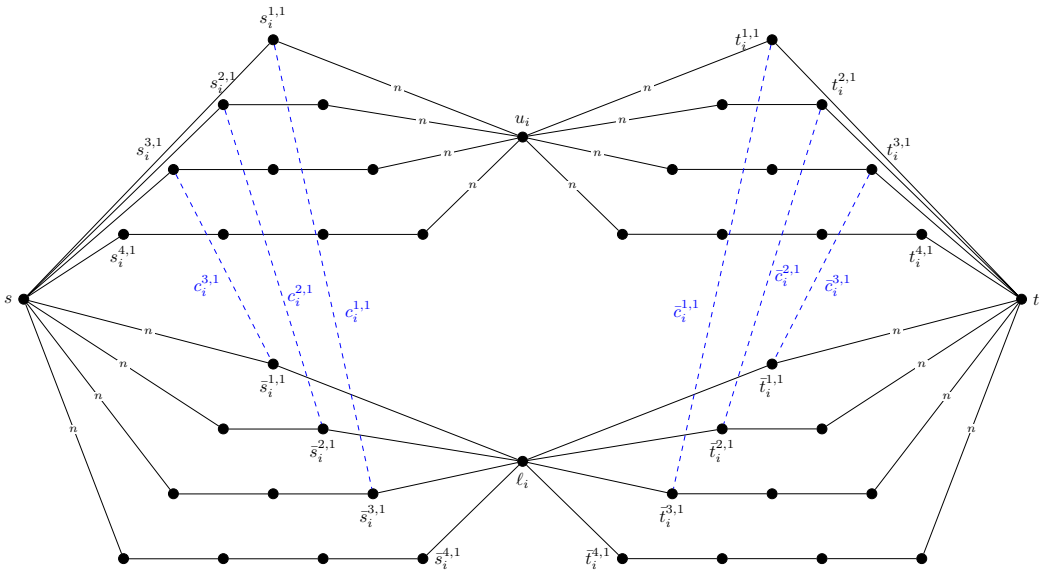

Figure 4 An example of a vertex-selection gadget for $\nu = 4$ and $m = 1$. Shortcut edges $c_i^{j,1}$ or $\bar{c}_i^{j,1}$ are drawn in dashed blue. For $m > 1$, the gadget contains m copies of this picture, sharing the vertices s, t, u_i, and ℓ_i.

Inside each vertex selection gadget, any λ-cut of size β contains exactly $2(\nu - 1)$ edges (as we shall see), selecting a vertex from V_i. Furthermore, for each pair of vertex-selection gadgets, any λ-cut of size at most β will contain an edge from each edge gadget for which the endpoints of this edge are not the two vertices selected by the vertex-selection gadgets.

Vertex-Selection Gadgets (see Figure 4). Each vertex selection gadget A_i starts in s and ends in t. It has two "middle vertices" u_i and ℓ_i. Between s and u_i (ℓ_i), there is a path $S_i^{j,p}$ ($\bar{S}_i^{j,p}$) of length $n+j$ for each $1 \leq j \leq \nu$ and $1 \leq p \leq m$, and a path $T_i^{j,p}$ ($\bar{T}_i^{j,p}$) of length $n+j$ between u_i (ℓ_i) and t for each $1 \leq j \leq \nu$ and $1 \leq p \leq m$.

Finally, we add "shortcut edges". From the second vertex $s_i^{j,p}$ (i.e., the vertex adjacent to s) of each path $S_i^{j,p}$, we add an edge $c_i^{j,p}$ to the second to last vertex $\bar{s}_i^{\nu-j,p}$ (i.e. the vertex adjacent to ℓ_i) of $\bar{S}_i^{\nu-j,p}$ for every $1 \leq j < \nu$ and $p \in [m]$. Similarly, from the second vertex $\bar{t}_i^{\nu-j,p}$ (i.e., the vertex adjacent to ℓ_i) of the path $\bar{T}^{\nu-j,p}$, we add an edge $\bar{c}_i^{j,p}$ to the second to last vertex $t_i^{j,p}$ (i.e., the vertex adjacent to t) of $T_i^{j,p}$ for every $1 \leq j < \nu$ and $p \in [m]$.

The idea behind these gadgets is that for each S_i and each $j \in [\nu - 1]$, the cut F has to contain an edge of every s-u_i-path of length at most $n + j$, or an edge of every u_i-t-path of length at most $\nu - j + n$ and an analogous statement holds for ℓ_i. The distance between s and u_i and the distance between s and ℓ_i then again encodes the selected vertex.

Edge Gadgets (see Figure 5). For each edge $e = \{v_i, v_j\}$ with $v_i \in V_i$ and $v_j \in V_j$, we add a vertex v_e. This vertex v_e is connected to t by an edge. We add a path from u_i to v_e of length $n + \nu - \mathsf{index}(v_i)$, a path from ℓ_i to v_e of length $n + \mathsf{index}(v_i)$, a path from u_j to v_e of length $n + \nu - \mathsf{index}(v_j)$ and a path from ℓ_j to v_e of length $n + \mathsf{index}(v_j)$. We say for any edge $e \in E$, that the edge gadget containing v_e is *corresponding* to e.

The union of all edge gadgets for edges between V_i and V_j verifies that the two vertices selected by A_i and A_j share an edge similar to the incidence-checking gadgets in Section 2. This is done as follows. We can remove one edge from all but $\binom{k}{2}$ edge gadgets. When

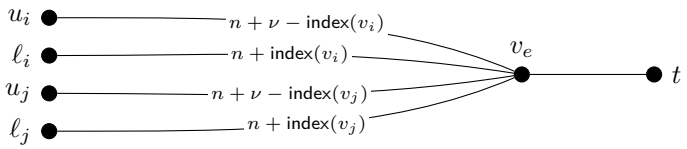

Figure 5 An example of an edge gadget for the edge $\{v_i, v_j\}$, where v_i belongs to V_i and v_j belongs to V_j.

deleting an edge from an edge gadget, one can always guarantee that no path of length λ uses this gadget. For each gadget that is not modified by the solution, there is a path of length λ using this gadget if and only if at least one of the two respective vertices was not selected. Hence, there is a solution if and only if there is a set $\{v_1, \ldots, v_k\}$ of k vertices with $v_i \in V_i$ such that $G[\{v_1, \ldots, v_k\}]$ contains $\binom{k}{2}$ edges, that is, a multicolored clique of size k.

3.2 Proof of Correctness

We first show how to construct a λ-cut F of size β from a clique $\{c_1, \ldots, c_k\}$ with $c_i \in V_i$ in G. For each vertex selection gadget, we add

- the first edge of all s-u_i-paths of length at most $n + \mathsf{index}(c_i) - 1$ (i.e., $\{\{s, s_i^{j,p}\} : j \leq \mathsf{index}(c_i) - 1, 1 \leq p \leq m\}$),
- the last edge of all s-ℓ_i-paths of length at most $n + \nu - \mathsf{index}(c_i)$ (i.e., $\{\{\bar{s}_i^{j,p}, \ell_i\} : j \leq \nu - \mathsf{index}(c_i), 1 \leq p \leq m\}$),
- the last edge of all u_i-t-paths of length at most $n + \nu - \mathsf{index}(c_i)$ (i.e., $\{\{t_i^{j,p}, t\} : j \leq \nu - \mathsf{index}(c_i), 1 \leq p \leq m\}$), and
- the first edge of all ℓ_i-t-paths of length at most $n + \mathsf{index}(c_i) - 1$ (i.e., $\{\{\ell_i, \bar{t}_i^{j,p}\} : j \leq \mathsf{index}(c_i) - 1, 1 \leq p \leq m\}$) to F.

For each edge e not corresponding to an edge inside the clique $\{c_1, \ldots, c_k\}$, we add the edge $\{v_e, t\}$ to F. First, we show that F contains exactly β edges.

▶ **Lemma 3.** *We have* $|F| = \beta = 2k(\nu - 1)m + m - \binom{k}{2}$.

Proof. The cut F contains $2(\nu - 1)m$ edges from each of the k vertex selection gadgets, and one edge for each of the $m - \binom{k}{2}$ edge gadgets not corresponding to an edge of the form $\{c_i, c_j\}$. Thus, we have $|F| = 2(\nu - 1)mk + m - \binom{k}{2} = \beta$. ◀

It remains to show that any s-t-path in $H - F$ has length at least $\lambda + 1$. We do so by first showing that each s-u_i-path has length at least $n + \mathsf{index}(c_i)$, and each s-ℓ_i-path has length at least $n + \nu - \mathsf{index}(c_i)$. Afterwards, we also bound the length of an u_i-t- and an ℓ_i-t-path from below. Before we can show the main results, we first need an auxiliary lemma:

▶ **Lemma 4.** *Any s-u_i-path (s-ℓ_i-, u_i-t-, or ℓ_i-t-path) in $H - F$ has length at least n.*

Proof. We only prove the lemma for s-u_i- and s-ℓ_i-paths; for the u_i-t- and ℓ_i-t-paths, the statement can be proven analogously.

We prove the lemma by contradiction. To this end, let $v \in \{u_i, \ell_i : i \in [k]\}$ be the vertex closest to s in $H - F$ among all vertices in $\{u_i, \ell_i : i \in [k]\}$, and assume that the distance from s to v is $d < n$. Let P be an s-v-path of length d.

By the choice of v, no vertex from $\{u_i, \ell_i : i \in [k]\}$ is an interior vertex of P. Thus, P has to be of the form s-$s_i^{j,p}$-$c_i^{j,p}$-$\bar{s}_i^{\nu-j,p}$-ℓ_i (in particular, we have $v = \ell_i$), as all paths $S_i^{j,p}$ and $\bar{S}_i^{j,p}$ are of length at least $n + 1$.

However, by the construction of F, either the edge $\{s, s_i^{j,p}\}$ (if $j < \mathsf{index}(c_i)$) or the edge $\{\bar{s}_i^{\nu-j,p}, \ell_i\}$ (if $j \geq \mathsf{index}(c_i)$) is contained in F, a contradiction. ◀

We can now analyze the length of all s-u_i- and s-ℓ_i-paths in $H - F$.

▶ **Lemma 5.** *Any s-u_i-path in $H - F$ has length at least $n + \mathsf{index}(c_i)$, and any s-ℓ_i-path has length at least $n + \nu - \mathsf{index}(c_i) + 1$.*

Proof. We first show that we only need to consider s-u_i- and s-ℓ_i-paths containing no vertex from $\{u_j, \ell_j : j \in [k]\}$ as an interior vertex. To see this, first note that the only connections between a vertex from $\{u_i, \ell_i\}$ to a vertex from $\{u_j, \ell_j : j \in [k] \setminus \{i\}\}$ in $G - \{s\}$ are through an edge gadget and thus of length at least $2n$ or through t and thus by Lemma 4 of length at least $2n$. Also any u_i-ℓ_i-path is of length at least n (as any path leaving u_i starts with a path of at least $n - 1$ vertices of degree two). Thus, any s-ℓ_i- or s-u_i-path containing u_i or ℓ_i as an interior vertex is of length at least $2n > n + \max\{\mathsf{index}(c_i), \nu - \mathsf{index}(c_i) + 1\}$ by Lemma 4.

We first consider paths not containing shortcut edges. Any s-u_i-path not containing a shortcut edge is of the form s-$S_i^{j,p}$-u_i. By the construction of F, such a path has length at least $n + \mathsf{index}(c_i)$. Any shortest s-ℓ_i-path not containing a shortcut edge is of the form s-$\bar{S}_i^{j,p}$-ℓ_i. By the construction of F, such a path has length at least $n + \nu - \mathsf{index}(c_i) + 1$. Now, consider an s-v-path P with $v \in \{u_i, \ell_i\}$ containing a shortcut edge $c_i^{j,p}$ for some $j \in [\nu - 1]$ and $p \in [m]$. By the construction of F, we have that either $\{s, s_i^{j,p}\}$ (if $j \leq \mathsf{index}(c_i) - 1$) or $\{\bar{s}_i^{\nu-j,p}, \ell_i\}$ (if $j \geq \mathsf{index}(c_i)$) is contained in F. Thus, F has to be of the form s-$\bar{S}_i^{\nu-j,p}$-$c_i^{j,p}$-$S_i^{j,p}$-u_i and has length $(n + \nu - j - 1) + 1 + (n + j - 1) = 2n + \nu - 1 > n + \mathsf{index}(c_i)$. ◀

Analogously, we can also show the following bound for all u_i-t- and ℓ_i-t-paths in $H - F$.

▶ **Lemma 6.** *Any u_i-t-path in $H - F$ has length at least $n + \nu - \mathsf{index}(c_i) + 1$, and each ℓ_i-t-path has length at least $n + \mathsf{index}(c_i)$.*

The correctness of the forward direction now easily follows.

▶ **Lemma 7.** *If G contains a clique of size k, then H contains a λ-cut of size β.*

Proof. By Lemma 3, we have $|F| = \beta$, so it suffices to show that F is a λ-cut. Consider any s-t-path P in $H - F$. Then P passes through a vertex u_i or ℓ_i for some $i \in [k]$, as any s-t-path in H passes through a vertex of the form u_j or ℓ_j. If P passes through u_i, then we get by Lemma 5 and 6 that the length of P is at least $n + \mathsf{index}(c_i) + n + \nu - \mathsf{index}(c_i) + 1 = \lambda + 1$. If P passes through ℓ_i, then we get by Lemma 5 and 6 that the length of P is at least $n + \nu - \mathsf{index}(c_i) + 1 + n + \mathsf{index}(c_i) = \lambda + 1$. ◀

Due to space constraints, the backward direction is proven in the full version [4].

3.3 Feedback Vertex Number

It remains to analyze the time required to compute the reduction and to show that the feedback vertex number of H is bounded in terms of k. We start with the running time.

▶ **Observation 8.** *The given reduction of Multicolored Clique parameterized by solution size k to Length-Bounded Cut parameterized by feedback vertex number can be computed in $O(k \cdot m \cdot n)$ time.*

Last but not least, we need to analyze the feedback vertex number of H. We do this by simply giving a feedback vertex set of size $O(k)$.

▶ **Lemma 9.** *The set $X := \{s, t\} \cup \{u_i, \ell_i : i \in [k]\}$ is a feedback vertex set in H.*

Proof. Note that all vertices from $V_H \setminus X$ are contained in a path $S_p^{i,j}$, $\bar{S}_p^{i,j}$, $T_p^{i,j}$, or $\bar{T}_p^{i,j}$ or contained in an edge gadget. All edges from the graph $H - X$ not contained in one of these paths or an edge gadget are the shortcut edges $c_p^{i,j}$ and $\bar{c}_p^{i,j}$.

Thus, there are only three kinds of different connected components in $H - X$, and all of them are trees:

- Clearly, edge gadgets are trees.
- Components of the form $\{S_i^{j,p}, c_i^{j,p}, \bar{S}^{\nu-j,p}\}$ or $\{T_i^{j,p}, \bar{c}_i^{j,p}, \bar{T}^{\nu-j,p}\}$ with $1 \leq j \leq \nu - 1$ are paths.
- Components of the form $S_i^{\nu,p}$, $T_i^{\nu,p}$, $\bar{S}_i^{\nu,p}$, and $\bar{T}^{\nu,p}$ are paths. ◀

Combining Lemmata 7 and 9 with Observation 8 and the backwards direction yields our desired main result.

▶ **Theorem 10.** *Length-Bounded Cut parameterized by feedback vertex number k is W[1]-hard. Assuming ETH, it cannot be solved in $f(k) \cdot n^{o(k)}$ time for any computable function f.*

4 Polynomial-Time Algorithm on Proper Interval Graphs

In this section, we present a polynomial-time algorithm for Length-Bounded Cut on proper interval graphs. The algorithm is a dynamic program that stores for each vertex v and each possible distance d ($2 \leq d \leq \lambda$) the minimal size of a cut that makes each vertex in a particular subset of vertices including v have distance at least d from s.

Observe that we can assume without loss of generality that $b_s \leq b_t$ as we can otherwise "mirror" the graph by setting $b_v = -f_v$ and $f_v = -b_v$ for each vertex $v \in V$. It is folklore that one can assume that all b-values are distinct, that is, $|\{b_v \mid v \in V\}| = |V|$. We now sort all the vertices in $V \setminus \{s, t\}$ by their respective b-value in increasing order and rename the vertices such that v_i is the i^{th} vertex in this order. Thus, we have $V = \{s, t\} \cup \{v_1, \ldots, v_{n-2}\}$, and $b_{v_i} < b_{v_{i+1}}$ for all $i \in [n-3]$. We first show that we can safely ignore all vertices v with $f_v < b_s$ or $f_t < b_v$. It is worth pointing out that the following lemma holds for interval graphs in general.

▶ **Lemma 11.** *Let $I = (G = (V, E), s, t, \beta, \lambda)$ be an instance of Length-Bounded Cut where G is an interval graph and $b_s < b_t$. Let $L = \{u \in V \mid f_u < b_s\}$ and $R = \{u \in V \mid f_t < b_u\}$. Then, $I' = (G[V \setminus (L \cup R)], s, t, \beta, \lambda)$ is an equivalent instance of Length-Bounded Cut.*

Proof. Let $I, I', G, s, t, \beta, \lambda, L$, and R be as defined above. We first show that $I_L = (G[V \setminus R], s, t, \beta, \lambda)$ is an equivalent instance. The argumentation for then removing L from I_L to obtain the equivalent instance I' is analogous and hence skipped here. First observe that $s, t \notin L \cup R$ and hence I_L and I' are instances of Length-Bounded Cut. Observe that deleting vertices from any input graph cannot decrease the distance between any pair of vertices and hence if I is a yes-instance, then so is I_L. Hence it remains to show that if I_L is a yes-instance, then so is I. Assume towards a contradiction that this is not the case and hence I_L is a yes–instance and I is a no-instance. Then there is a set F_L of β edges in $G[V \setminus R]$ such that the distance between s and t in $G_L = (V \setminus R, E \setminus (F_L \cup \{\{u, v\} \in E \mid u \in R\}))$ is at least $\lambda + 1$. Since I is a no-instance, there is a path P of length at most λ between s and t in $G^* = (V, E \setminus F_L)$. As G_L and G^* only differ in R, each path of length at most λ between s and t in G^* contains at least one vertex from R. We show that $\deg_G(t) \leq |F_L|$ and hence there is an s-t-cut of size at most β in G and thus I is a yes-instance. This contradicts the assumption that I is a no-instance and hence finishes the proof that I_L is equivalent to I.

We start by giving some basic notation for the proof to come. We use sets of vertices that have a certain distance from s in some subgraph H of G. To this end, we define $X_H^p = \{u \in V \mid \text{dist}_H(s,u) = p\}$ for each distance p. Analogously, we define $X_H^{\leq p} = \{u \in V \mid \text{dist}_H(s,u) \leq p\}$ and $X_H^{\geq p} = \{u \in V \mid \text{dist}_H(s,u) \geq p\}$.

Let $d = \text{dist}_{G^*}(s,t)$ and let t' be the vertex in P with maximum $b_{t'}$. Since P contains a vertex from R, it holds that $b_{t'} > f_t$ and hence $t' \notin N_G(t)$. Since t' is on a shortest s-t-path in G^* and $t' \notin N_G(t)$ it holds that $t' \in X_{G^*}^{\leq d-2}$. Now consider the set K of vertices that are part of a shortest s-t'-path in G^* and that are neighbors of t in G. By construction $K \subseteq X_{G^*}^{\leq d-3}$ and for each $y \in [b_t, f_t]$ there is a vertex $v \in K$ with $y \in [b_v, f_v]$. We next show that $|F_L| \geq \deg_G(t)$. To this end, consider any vertex $u \in N_G(t)$. If $u \in X_{G^*}^{\leq d-2}$, then it holds that $\{u,t\} \in F_L$. Otherwise we have $u \in X_{G^*}^{\geq d-1}$. Observe that for each $u \in N_G(t)$ it holds by definition that $[b_u, f_u] \cap [b_t, f_t] \neq \emptyset$ and hence there is a vertex $v \in K$ with $\{u,v\} \in E$. Since $\text{dist}_{G^*}(s,u) \geq d-1 > d-3+1 \geq \text{dist}_{G^*}(s,v)+1$ it holds that $\{u,v\} \in F_L$. As $K \cap X_{G^*}^{\geq d-1} = \emptyset$, it is then easy to verify that $\beta = |F_L| \geq \deg_G(t)$ and hence there is a trivial s-t-cut of size β in G that just removes all incident edges of t. This contradicts the assumption that I is a no-instance and thus concludes the proof. ◀

Using Lemma 11, we assume that there is no vertex v with $f_v < b_s$ or $b_v > f_t$. We next show that there always exists a solution in which the distance from s to v_j is non-decreasing in j. The proofs of the following results can be found in the full version [4].

▶ **Lemma 12.** *Let $G = (V,E)$ be a proper interval graph such that there is no vertex v with $f_t < b_v$ or $b_s > f_v$ and let F be a set of edges. Let d be the distance from s to t in $(V, E \setminus F)$. Then, there is a set F' of edges with $|F'| \leq |F|$ such that for $G'' = (V, E \setminus F')$ it holds that $\text{dist}_{G''}(s,t) \geq d$ and for each $v_i, v_j \in V \setminus \{s,t\}$ with $b_{v_i} < b_{v_j}$ it holds that $\text{dist}_{G''}(s,v_i) \leq \text{dist}_{G''}(s,v_j)$.*

Using this, we are now able to state a dynamic program to show that Length-Bounded Cut can be solved in polynomial time on proper interval graphs. The dynamic program stores for each vertex v and each possible distance d the minimal size of a cut that makes each vertex u with $b_u \geq b_v$ have distance at least d from s.

▶ **Theorem 13.** *Length-Bounded Cut can be solved in $O(n^3 \cdot m)$ time if the input graph is a proper interval graph.*

Proof sketch. We assume that there is no s-t-cut of size at most β in the input graph G as this case can easily be detected in $O(n \cdot m)$ time [11] and the answer for Length-Bounded Cut is then always yes. This implies that $\deg_G(s), \deg_G(t) > \beta$. Furthermore, by Lemma 11 we can assume that there is no vertex v with $f_v < b_s$ or $b_v > f_t$. By Lemma 12 we can assume that we search for a solution in which for all $v_i, v_j \in V \setminus \{s,t\}$ with $b_{v_i} < b_{v_j}$ it holds that $\text{dist}(s,v_i) \leq \text{dist}(s,v_j)$. Hence, we construct a table T which stores for each vertex $v_i \in V \setminus \{s,t\}$ and each possible distance $d \in \{2, \ldots, \lambda\}$ the minimum number of edges that have to be deleted from $G - \{t\}$ to ensure that all vertices $v_j \in V \setminus \{s,t\}$ with $b_{v_j} \geq b_{v_i}$ have distance at least d from s, and furthermore, $\text{dist}(s,v_k) \leq \text{dist}(s,v_\ell)$ holds for all $k \leq \ell \leq i$. Observe that $\text{dist}(s,s) = 0$ in any graph and since we are looking for a solution in which $\text{dist}(s,t) > \lambda$, we search for a solution in which all neighbors u of t satisfy $\text{dist}(s,u) \geq \lambda$. In the last step we then try all neighbors of t to be the last vertex before t in a shortest s-t-path to find an optimal solution. To avoid confusion recall that all vertices *except for s and t* are labeled by $v_1, v_2, \ldots, v_{n-2}$. We initialize T by setting $T[v_i, 2] = |\{v_j \mid b_{v_j} \leq f_s \wedge j \geq i\}|$ for all v_i with $b_{v_i} \leq f_s$ and $T[v_\ell, 2] = 0$ for all vertices v_ℓ that are not adjacent to s as any non-neighbor w of s has distance at least

two from s, and $b_w > f_s$. We further initialize $T[v_1, d] = \deg(s)$ for all $d \geq 3$. We also store for each table entry $T[v_i, d]$ with $d \geq 2$ in a second table $S[v_i, d]$ the vertex v_j with maximum b_{v_j}-value such that all edges $\{v_\ell, v_r\}$ with $\ell < j$ and $r \geq i$ are contained in a minimal cut (a set of edges to delete from G) guaranteeing that each vertex $v_{r'}$ with $r' \geq i$ has distance at least d from s. We initialize $S[v_i, 2] = 1$ for all v_i and $S[v_1, d] = 1$ for all $d > 2$ as we only delete edges incident to s in these cases. For increasing values of d, we iterate over all vertices $v_i \in V \setminus \{s, t, v_1\}$ in order of b_{v_i} and compute

$$T[v_i, d] = \min_{j \leq i} \{T[v_j, d-1] + C[S[v_j, d-1], v_j, v_i]\}, \text{ and}$$
$$S[v_i, d] = \operatorname{argmin}_{j \leq i} \{T[v_j, d-1] + C[S[v_j, d-1], v_j, v_i]\},$$

where $C[v_h, v_i, v_j]$ is a function that represents, for each triple (v_h, v_i, v_j) with $h < i < j$ of vertices, the size of a minimal cut (the number of edges to delete from G) to ensure that there is no edge between a vertex v_ℓ with $h \leq \ell < i$ and a vertex v_r with $r \geq j$. For technical reasons, we exclude s here and hence the formal definition is

$$C[v_h, v_i, v_j] = |\{\{v_\ell, v_r\} \in E \mid h \leq \ell < i \wedge r \geq j \wedge v_\ell \neq s \neq v_r\}|.$$

The vertex v_h will only be used to avoid double counting. The correctness and running time of the algorithm is proven in the full version [4]. ◀

5 Conclusion

In this paper, we studied Length-Bounded Cut with respect to feedback vertex number, the combined parameter pathwidth plus maximum degree and the special case when the input graph is a proper interval graph. We showed that it is W[1]-hard with respect to the feedback vertex number and polynomial-time solvable on proper interval graphs. The latter confirms a conjecture by Bazgan et al. [3] and both fill-in gaps in their hierarchies for Length-Bounded Cut from a parameterized respectively graph-classes point of view. Natural next steps include the remaining open questions in these hierarchies. In particular, interval graphs are the last remaining graph class in their graph-class hierarchy for Length-Bounded Cut. We conjecture that it should be possible to extend Theorem 13 to also work on interval graphs. Lastly, we showed that Length-Bounded Cut is W[1]-hard with respect to the combined parameter pathwidth and maximum degree. This combines two results by Dvořák and Knop [9] and Bazgan et al. [3]. It strengthens the former, which states that the problem is W[1]-hard with respect to the parameter pathwidth, and complements the latter, which shows that the problem is in XP for the parameter maximum degree. The question whether it is FPT or W[1]-hard for the parameter maximum degree was left open by Bazgan et al. [3].

References

1 Jiří Adámek and Václav Koubek. Remarks on flows in network with short paths. *Comment. Math. Univ. Carolinae*, 12:661–667, 1971.

2 Georg Baier, Thomas Erlebach, Alexander Hall, Ekkehard Köhler, Petr Kolman, Ondřej Pangrác, Heiko Schilling, and Martin Skutella. Length-bounded cuts and flows. *ACM Trans. Algorithms*, 7(1):4:1–4:27, 2010.

3 Cristina Bazgan, Till Fluschnik, André Nichterlein, Rolf Niedermeier, and Maximilian Stahlberg. A more fine-grained complexity analysis of finding the most vital edges for undirected shortest paths. *Networks*, 73(1):23–37, 2019.

4 Matthias Bentert, Klaus Heeger, and Dusan Knop. Length-bounded cuts: Proper interval graphs and structural parameters. *CoRR*, abs/1910.03409, 2019. arXiv:1910.03409.

5 Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. *Graph Classes: A Survey*. Philadelphia, PA: SIAM, 1999.

6 Jianer Chen, Benny Chor, Mike Fellows, Xiuzhen Huang, David W. Juedes, Iyad A. Kanj, and Ge Xia. Tight lower bounds for certain parameterized np-hard problems. *Inf. Comput.*, 201(2):216–231, 2005.

7 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

8 Yefim Dinitz. Dinitz' algorithm: The original version and Even's version. In *Theoretical Computer Science, Essays in Memory of Shimon Even*, pages 218–240, 2006.

9 Pavel Dvořák and Dušan Knop. Parameterized complexity of length-bounded cuts and multicuts. *Algorithmica*, 80(12):3597–3617, 2018.

10 Till Fluschnik, Danny Hermelin, André Nichterlein, and Rolf Niedermeier. Fractals for kernelization lower bounds. *SIAM J. Discrete Math.*, 32(1):656–681, 2018.

11 Lester Randolph Ford and Delbert Ray Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

12 Petr A. Golovach and Dimitrios M. Thilikos. Paths of bounded length and their cuts: Parameterized complexity and algorithms. *Discrete Optimization*, 8(1):72–86, 2011.

13 Luís Gouveia, Pedro Patrício, and Amaro de Sousa. Hop-constrained node survivable network design: An application to mpls over wdm. *Networks and Spatial Economics*, 8(1):3–21, March 2008. doi:10.1007/s11067-007-9038-3.

14 David Huygens, Martine Labbé, A. Ridha Mahjoub, and Pierre Pesneau. The two-edge connected hop-constrained network design problem: Valid inequalities and branch-and-cut. *Networks*, 49(1):116–133, 2007. doi:10.1002/net.20146.

15 David Huygens and A. Ridha Mahjoub. Integer programming formulations for the two 4-hop-constrained paths problem. *Networks*, 49(2):135–144, 2007. doi:10.1002/net.20147.

16 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

17 Petr Kolman. On algorithms employing treewidth for l-bounded cut problems. *J. Graph Algorithms Appl.*, 22(2):177–191, 2018.

18 Petr Kolman and Christian Scheideler. Improved bounds for the unsplittable flow problem. *J. Algorithms*, 61(1):20–44, 2006.

19 Ali Ridha Mahjoub and S. Thomas McCormick. Max flow and min cut with bounded-length paths: complexity, algorithms, and approximation. *Math. Program.*, 124(1-2):271–284, 2010.

20 Vishv M. Malhotra, M. Pramodh Kumar, and S. N. Maheshwari. An $O(|V|^3)$ algorithm for finding maximum flows in networks. *Inf. Process. Lett.*, 7(6):277–278, 1978.

21 Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.