

List Homomorphisms by Deleting Edges and Vertices: Tight Complexity Bounds for Bounded-Treewidth Graphs

Bariş Can Esmer  



CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
Saarbrücken Graduate School of Computer Science, Saarland Informatics Campus, Germany

Jacob Focke  

CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Dániel Marx  

CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Paweł Rzażewski  

Warsaw University of Technology, Poland
University of Warsaw, Poland

Abstract

The goal of this paper is to investigate a family of optimization problems arising from list homomorphisms, and to understand what the best possible algorithms are if we restrict the problem to bounded-treewidth graphs. Given graphs G , H , and lists $L(v) \subseteq V(H)$ for every $v \in V(G)$, a *list homomorphism* from (G, L) to H is a function $f : V(G) \rightarrow V(H)$ that preserves the edges (i.e., $uv \in E(G)$ implies $f(u)f(v) \in E(H)$) and respects the lists (i.e., $f(v) \in L(v)$). The graph H may have loops. For a fixed H , the input of the optimization problem $\text{LHOMVD}(H)$ is a graph G with lists $L(v)$, and the task is to find a set X of vertices having minimum size such that $(G - X, L)$ has a list homomorphism to H . We define analogously the edge-deletion variant $\text{LHOMED}(H)$, where we have to delete as few edges as possible from G to obtain a graph that has a list homomorphism. This expressive family of problems includes members that are essentially equivalent to fundamental problems such as VERTEX COVER , MAX CUT , $\text{ODD CYCLE TRANSVERSAL}$, and $\text{EDGE/VERTEX MULTIWAY CUT}$.

For both variants, we first characterize those graphs H that make the problem polynomial-time solvable and show that the problem is NP-hard for every other fixed H . Second, as our main result, we determine for every graph H for which the problem is NP-hard , the smallest possible constant c_H such that the problem can be solved in time $c_H^t \cdot n^{\mathcal{O}(1)}$ if a tree decomposition of G having width t is given in the input. Let $i(H)$ be the maximum size of a set of vertices in H that have pairwise incomparable neighborhoods. For the vertex-deletion variant $\text{LHOMVD}(H)$, we show that the smallest possible constant is $i(H) + 1$ for every H :

- Given a tree decomposition of width t of G , $\text{LHOMVD}(H)$ can be solved in time $(i(H) + 1)^t \cdot n^{\mathcal{O}(1)}$.
- For any $\varepsilon > 0$ and H , an $(i(H) + 1 - \varepsilon)^t \cdot n^{\mathcal{O}(1)}$ algorithm would violate the Strong Exponential-Time Hypothesis (SETH).

The situation is more complex for the edge-deletion version. For every H , one can solve $\text{LHOMED}(H)$ in time $i(H)^t \cdot n^{\mathcal{O}(1)}$ if a tree decomposition of width t is given. However, the existence of a specific type of decomposition of H shows that there are graphs H where $\text{LHOMED}(H)$ can be solved significantly more efficiently and the best possible constant can be arbitrarily smaller than $i(H)$. Nevertheless, we determine this best possible constant and (assuming the SETH) prove tight bounds for every fixed H .

2012 ACM Subject Classification Theory of computation \rightarrow Parameterized complexity and exact algorithms

Keywords and phrases Graph Homomorphism, List Homomorphism, Vertex Deletion, Edge Deletion, Multiway Cut, Parameterized Complexity, Tight Bounds, Treewidth, SETH



© Barış Can Esmer, Jacob Focke, Dániel Marx, and Paweł Rzażewski;
licensed under Creative Commons License CC-BY 4.0

32nd Annual European Symposium on Algorithms (ESA 2024).

Editors: Timothy Chan, Johannes Fischer, John Iacono, and Grzegorz Herman; Article No. 39; pp. 39:1–39:20

Leibniz International Proceedings in Informatics



LIPIC Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Digital Object Identifier 10.4230/LIPIcs.ESA.2024.39

Related Version Full Version: <https://arxiv.org/abs/2210.10677>

Funding Paweł Rzażewski: Supported by the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme grant agreement number 948057.

1 Introduction

Typical NP-hard graph problems are known to be solvable in polynomial time when the input graph is restricted to be of bounded treewidth. In many cases, the problem is actually fixed-parameter tractable (FPT) parameterized by treewidth: given a tree decomposition of width t , the problem can be solved in time $f(t) \cdot n^{\mathcal{O}(1)}$ for some function f [6, 9, 8]. While early work focused on just establishing this form of running time, more recently there is increased interest in obtaining algorithms where the function f is growing as slowly as possible. New techniques such as representative sets, cut-and-count, subset convolution, and generalized convolution were developed to optimize the function $f(t)$.

On the complexity side, a line of work started by Lokshtanov, Marx, and Saurabh [35] provides tight lower bounds for many problems where $c^t \cdot n^{\mathcal{O}(1)}$ -time algorithms were known. These type of complexity results typically show the optimality of the base c of the exponent in the best known $c^t \cdot n^{\mathcal{O}(1)}$ -time algorithm, by proving that the existence of a $(c - \varepsilon)^t \cdot n^{\mathcal{O}(1)}$ -algorithm for any $\varepsilon > 0$ would violate the Strong Exponential-Time Hypothesis (SETH) [41, 40, 11, 4, 2, 31, 36, 5, 20, 37, 19]. The goal of this paper is to unify some of these lower bounds under the umbrella of *list homomorphism with deletion* problems, obtaining tight lower bounds for an expressive family of problems that include members that are essentially equivalent to fundamental problems such as VERTEX COVER, MAX CUT, ODD CYCLE TRANSVERSAL, and EDGE/VERTEX MULTIWAY CUT.

Graph homomorphisms. Given graphs G and H , a *homomorphism from G to H* is a (not necessarily injective) mapping $f : V(G) \rightarrow V(H)$ that preserves the edges of G , that is, if $uv \in E(G)$, then $f(u)f(v) \in E(H)$. For example, if H is the complete graph K_c on c vertices, then the homomorphisms from G to H correspond to the proper vertex c -colorings of G : adjacent vertices have to be mapped to distinct vertices of H . For a fixed graph H , the problem $\text{HOM}(H)$ asks if the given graph G has a homomorphism to H . Motivated by the connection to c -coloring when $H = K_c$, the problem is also called H -COLORING [24, 28, 27, 26, 38, 39, 21].

The list version of $\text{HOM}(H)$ is the generalization of the problem where the possible image of each $v \in V(G)$ is restricted [13, 25, 16, 15, 29, 10, 7, 11, 1, 40, 20]. This generalization allows us to express a wider range of problems and it makes complexity results more robust. Formally, for a fixed undirected graph H (possibly with selfloops), the input of the $\text{LHOM}(H)$ problem consists of a graph G and a *list assignment* $L : V(G) \rightarrow 2^{V(H)}$, the task is to decide if there is a *list homomorphism* f from (G, L) to H , that is, a homomorphism f from G to H that satisfies $f(v) \in L(v)$ for every $v \in V(G)$. Note that $\text{HOM}(H)$ is trivial if H has a vertex with a loop, but loops may have a non-trivial role in the $\text{LHOM}(H)$ problem as not every list may contain the same looped vertex. In fact, it is already non-trivial to consider the special case where H is *reflexive* [13], that is, every vertex of H has a loop.

The main topic of the current paper is a further generalization of $\text{LHOM}(H)$ to an optimization problem where we are allowed to delete some edges/vertices of G . The edge-deletion variant $\text{LHOMED}(H)$ is defined the following way: given a graph G and a list

assignment $L : V(G) \rightarrow 2^{V(H)}$, the task is to find a minimum set $X \subseteq E(G)$ of edges such that $(G \setminus X, L)$ has a list homomorphism to H . In other words, we want to find a mapping $f : V(G) \rightarrow V(H)$ that satisfies $f(v) \in L(v)$ for every $v \in V(G)$ and satisfies $f(u)f(v) \in E(H)$ for the maximum number of edges uv of G . The vertex-deletion variant $\text{LHOMVD}(H)$ is defined analogously: here the task is to find a minimum size set X of vertices such that $(G - X, L)$ has a list homomorphism to H . The $\text{LHOMVD}(H)$ problem was considered from the viewpoint of FPT algorithms parameterized by the number of removed vertices [3, 32].

While the $\text{HOM}(H)$ and $\text{LHOM}(H)$ problems can be seen as generalizations of vertex coloring, the framework of deletion problems we consider here can express a wide range of fundamental optimization problems. We show below how certain problems can be reduced to $\text{LHOMED}(H)$ or $\text{LHOMVD}(H)$ for some fixed H . The reductions mostly work in the other direction as well (we elaborate on that in Section 2), showing that this framework contains problems that are essentially equivalent to well-studied basic problems.

- **VERTEX COVER**: Let $H = K_1$ be a single vertex x without a loop. Then **VERTEX COVER** can be expressed by $\text{LHOMVD}(K_1)$ with single-element lists: as vertex x is not adjacent to itself, it follows that for every edge uv of G , at least one of u and v has to be deleted.
- **INDEPENDENT SET**: As G has an independent set of size k if and only if it has a vertex cover of size $|V(G)| - k$, the same reduction can be used.
- **MAX CUT**: Let $H = K_2$ be two adjacent vertices without loops. Then **MAX CUT** can be expressed by $\text{LHOMED}(H)$ with the list $V(H)$ at every vertex: the task is to delete the minimum number of edges to obtain a bipartition (X, Y) , that is, to maximize the number of edges between X and Y .
- **ODD CYCLE TRANSVERSAL**: Let $H = K_2$ be two adjacent vertices without loops. Then **ODD CYCLE TRANSVERSAL** can be expressed by $\text{LHOMVD}(H)$ with the list $V(H)$ at every vertex: the task is to delete the minimum number of vertices to obtain a bipartite graph.
- **s - t MIN CUT**: Let H contain two independent vertices v_s and v_t with loops. Then **s - t MIN CUT** can be expressed as $\text{LHOMED}(H)$ where $L(s) = \{v_s\}$, $L(t) = \{v_t\}$, and the list is $\{v_s, v_t\}$ for all remaining vertices. It is clear that s and t cannot be in the same component after removing the solution X from G .
- **EDGE MULTIWAY CUT** with c terminals t_1, \dots, t_c : Let H be c independent vertices v_1, \dots, v_c with selfloops. Then the problem can be expressed as $\text{LHOMED}(H)$ where $L(t_i) = \{v_i\}$ and non-terminals have list $\{v_1, \dots, v_c\}$. It is clear that if X is a solution of $\text{LHOMED}(H)$, then each component of $G \setminus X$ can contain at most one terminal.
- **VERTEX MULTIWAY CUT** with c (undeletable) terminals t_1, \dots, t_c : Let H be c independent vertices v_1, \dots, v_c selfloops. First we modify the graph: if terminal t_i is adjacent to a vertex w , then we replace t_i by $n = |V(G)|$ degree-1 copies adjacent to w . Then the problem can be expressed as $\text{LHOMVD}(H)$ where the list is $\{v_i\}$ for each copy of t_i and $\{v_1, \dots, v_c\}$ for each non-terminal vertex. Observe that it does not make sense to delete a copy of any terminal. Therefore, the optimal solution to $\text{LHOMVD}(H)$ is a set of vertices, disjoint from the terminals, that separates the original terminals.

For every fixed H , our results give tight lower bounds for $\text{LHOMVD}(H)$ and $\text{LHOMED}(H)$, parameterized by the width of the given tree decomposition problems. This comprehensive set of results reprove earlier lower bounds on basic problems in a uniform way, extend them to new problems that have not been considered before (e.g., **MULTIWAY CUT**), and in fact fully investigate a large, well-defined family of problems. Earlier results in this area typically focused on specific problems or relatively minor variants of a specific problem.

Compared to that, our results focus on a family of problems that include a diverse set of optimization problems interesting on their own right. The tight characterization also includes an algorithmic surprise: for some $\text{LHOMED}(H)$ problems, the obvious brute force algorithm is not optimal on its own, one needs to consider a new form of decomposition into subproblems to achieve the best possible algorithm.

Polynomial-time cases. The seminal work of Nešetřil and Hell [24] characterized the polynomial-time solvable cases of $\text{HOM}(H)$: it can be solved in polynomial time if H is bipartite or has a loop, and it is NP-hard for every other fixed H . For the more general list version, we need more restrictions: Feder, Hell, and Huang [16] showed that the problem is polynomial-time solvable if H is a bi-arc graph. It is somewhat surprising that the polynomial-time solvability of the deletion versions has not been systematically studied, despite the amount of attention this type of problems received in the literature [24, 13, 15, 25, 16, 17, 29, 10, 7, 1, 14, 30, 23, 18]. This list also includes a lot of research on generalizations of the deletion problems, namely, minimum cost homomorphism and valued constraint satisfaction problems. Our first contribution is a polynomial-time versus NP-hard dichotomy for $\text{LHOMVD}(H)$ and $\text{LHOMED}(H)$. As expected, these more general problems remain polynomial-time solvable only for an even more restricted class of graphs. In particular, the reduction above from VERTEX COVER shows that $\text{LHOMVD}(H)$ becomes NP-hard already when there is a single loopless vertex in H and hence we can expect polynomial-time algorithms only for reflexive graphs H .

► **Theorem 1.** *The $\text{LHOMVD}(H)$ problem is polynomial-time solvable if H is reflexive and does not contain any of the following:*

1. *three pairwise non-adjacent vertices,*
2. *an induced four-cycle, or*
3. *an induced five-cycle.*

Otherwise, $\text{LHOMVD}(H)$ is NP-hard.

Edge-deletion problems are typically easier than their vertex-deletion counterparts, but the boundary line between the easier and hard cases is more difficult to characterize. This is also true in our case: for $\text{LHOMED}(H)$, the graph H does not have to be reflexive to make the problem polynomial-time solvable, hence the proof of the classification result becomes significantly more complicated as graphs with both reflexive (i.e., looped) and irreflexive (i.e., non-looped) vertices must be handled as well. We need the following definition to state the dichotomy result. We say that the three vertices v_1, v_2, v_3 have *private neighbors* if there are vertices v'_1, v'_2, v'_3 (not necessarily disjoint from $\{v_1, v_2, v_3\}$) such that v_i and v'_j are adjacent if and only if $i = j$. In particular, if $\{v_1, v_2, v_3\}$ are independent reflexive vertices then they have private neighbors. *Co-private neighbors* are defined similarly, but $i = j$ is replaced by $i \neq j$. In particular, if $\{v_1, v_2, v_3\}$ are pairwise adjacent irreflexive vertices, then they have co-private neighbors. Finally, we say an edge is *irreflexive* if both of its endpoints are irreflexive vertices.

► **Theorem 2.** *The $\text{LHOMED}(H)$ problem is polynomial time solvable if H does not contain any of the following:*

1. *an irreflexive edge,*
2. *a 3-vertex set with private neighbors, or*
3. *a 3-vertex set with co-private neighbors.*

Otherwise, $\text{LHOMED}(H)$ is NP-hard.

The proof of Theorem 2 exploits a delicate interplay between the geometric bi-arc representation (in the algorithm) and the characterization by forbidden subgraphs (for hardness). While the proofs of these dichotomy results are non-trivial, we do not consider them to be the main results of the paper. Clearly, understanding the easy and hard cases of the problem is a necessary prerequisite for the lower bounds we are aiming at, hence we needed to prove these dichotomy results as they were not present in the literature in this form. We remark that $\text{LHOMED}(H)$ can be formulated as a Valued Constraint Satisfaction Problem (VCSP) with a single binary relation, hence the existence of a polynomial-time versus NP-hard dichotomy should follow from known results on the complexity of VCSP [33, 42, 34]. However, we obtain in a self-contained way a compact statement of an easily checkable classification property with purely graph-theoretic proofs and algorithms.

Bounded-treewidth graphs, vertex deletion. Let us first consider the vertex-deletion version $\text{LHOMVD}(H)$ and determine how exactly the complexity of the problem depends on treewidth. We assume that the input contains a tree decomposition of G having width t , we try to determine the smallest c such that the problem can be solved in time $c^t \cdot n^{\mathcal{O}(1)}$. This question has been investigated for $\text{HOM}(H)$ [41], $\text{LHOM}(H)$ [11, 40], and the counting version of $\text{LHOM}(H)$ [20].

Standard dynamic programming techniques show that $\text{LHOMVD}(H)$ can be solved in time $(|V(H)| + 1)^t \cdot n^{\mathcal{O}(1)}$ if a tree decomposition of width t is given: each vertex has $|V(H)|$ possible “states” corresponding to where it is mapped to, plus one more state corresponding to deleting the vertex. For some H , this naive algorithm can be improved the following way. First, if every vertex in G has a list of size at most ℓ , then $(|V(H)| + 1)$ can be improved to $\ell + 1$: each vertex has only ℓ states corresponding to the possible images, plus the state representing deletion. Second, we say that a set $S \subseteq V(H)$ is *incomparable* if the neighborhoods of any two vertices in S are incomparable, that is, for any $u, v \in S$, there is $u' \in \Gamma(u) \setminus \Gamma(v)$ and $v' \in \Gamma(v) \setminus \Gamma(u)$ (we denote by $\Gamma(v)$ the neighborhood of a vertex v , which includes v itself if it has a loop). Let $i(H)$ be the size of the largest incomparable set in H . The main observation (already made in [11, 40]) is that we can assume that every list $L(v)$ is an incomparable set: if $\Gamma(v) \subseteq \Gamma(v')$ for $v, v' \in L(v)$, then we can always use v' in place of v in a solution. Therefore, we can assume that every list has size at most $i(H)$, resulting in running time $(i(H) + 1)^t \cdot n^{\mathcal{O}(1)}$. Our main result for the vertex-deletion version shows the optimality of this running time.

► **Theorem 3 (Main result for treewidth, vertex deletion).** *Let H be a fixed graph which contains either an irreflexive vertex or three pairwise non-adjacent reflexive vertices or an induced reflexive cycle on four or five vertices. Then $\text{LHOMVD}(H)$ on n -vertex instances given with a tree decomposition of width t*

(a) *can be solved in time $(i(H) + 1)^t \cdot n^{\mathcal{O}(1)}$ and*

(b) *cannot be solved in time $(i(H) + 1 - \varepsilon)^t \cdot n^{\mathcal{O}(1)}$ for any $\varepsilon > 0$, unless the SETH fails.*

Theorem 3 refines the NP-hardness of Theorem 1 by obtaining a lower bound that precisely matches the algorithm described previously. This shows that for $\text{LHOMVD}(H)$, restricting the lists to incomparable sets is the *only* algorithmic idea that can improve the running time of the naive algorithm. In particular, we cannot consider the connected components of H separately (as was possible in the earlier results [41, 40, 11, 20]). It is an essential feature of the deletion problem that hardness can stem from disconnected structures. (Note, for example, that MULTIWAY CUT is expressed by a graph H with multiple connected components, each of which individually does not entail hardness.) This difference makes

it necessary to approach the problem in a novel way: while earlier hardness proofs mostly relied on the idea of translating values by “moving them along a path in H ,” in our proofs we translate values by “jumping around,” potentially using nonedges as well. This makes it much less clear how the graph-theoretic structure of H should be connected to the complexity of the problem.

Bounded-treewidth graphs, edge deletion. For the edge-deletion version $\text{LHOMED}(H)$, the natural expectation is that $i(H)^t \cdot n^{\mathcal{O}(1)}$ is the best possible running time: as vertices cannot be deleted, each vertex v has only $|L(v)| \leq i(H)$ states in the dynamic programming. While this running time can be achieved using the idea of incomparable sets, it turns out that, somewhat surprisingly, this is *not* the optimal running time for every H . There are graphs H for which $\text{LHOMED}(H)$ can be solved significantly faster, thanks to a new algorithmic idea, the use of a specific form of decomposition. We need the following definition.

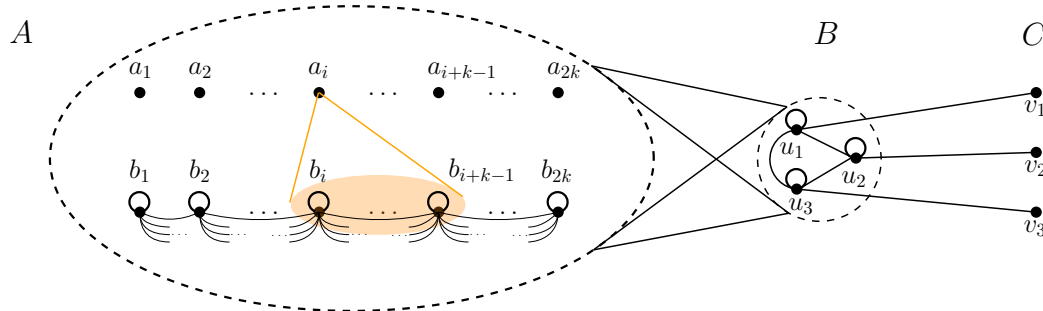
► **Definition 4 (Decomposition).** Given a graph H with vertex set V and a partition of V into three possibly empty sets A , B , and C , we say that (A, B, C) is a *decomposition* of H if the following hold:

- B is a reflexive clique with a full set of edges between A and B ,
- C is an (irreflexive) independent set with no edge between A and C ,
- $A \neq \emptyset$ and $B \cup C \neq \emptyset$.

The crucial property of this definition is that if S is an incomparable set, then it is fully contained in one of A , B , or C . Indeed, for any $a \in A$, $b \in B$, $c \in C$, we have $\Gamma(c) \subseteq \Gamma(a) \subseteq \Gamma(b)$. Therefore, if we assume that each list $L(v)$ is an incomparable set, then each $L(v)$ is a subset of one of these three sets. Let V_A, V_B, V_C be the sets of vertices of G whose lists are a subset of A , B , and C , respectively. Observe that if $u \in V_A$ and $v \in V_B$ are adjacent in G , then whenever assignment $f : V(G) \rightarrow V(H)$ respects the lists of u and v , then $f(u)f(v)$ is *always* an edge of H (as A and B are fully connected). Therefore, the edge uv of G does not play any role in the problem. Similarly, if $u \in V_A$ and $v \in V_C$, then $f(u)f(v)$ is *never* an edge of H (as A and C are independent), hence uv always has to be deleted in the solution. This means that the edges between V_A and $V_B \cup V_C$ can be ignored and the problem falls apart into two independent instances $G[V_A]$ of $\text{LHOMED}(H[A])$ and $G[V_B \cup V_C]$ of $\text{LHOMED}(H[V_B \cup V_C])$.

How does this observation help solving the problem more efficiently? As every incomparable set is a subset of one of the three sets, we have $i(H) = \max\{i(H[A]), i(H[B \cup C])\}$. Thus it seems that one of the two instances will be at least as hard as the original instance. The catch is that it could happen that one of the two instances is polynomial-time solvable and contains a large incomparable set, while the other is NP-hard but contains only small incomparable sets. For example, it is possible that $i(H) = i(H[A]) = k$, $i(H[B \cup C]) = 3$, but $\text{LHOMED}(H[A])$ is polynomial-time solvable. Then we can decompose the problem into an instance of $\text{LHOMED}(H[A])$ and an instance of $\text{LHOMED}(H[B \cup C])$, solve the former in polynomial time, and the latter in time $i(H[B \cup C])^t \cdot n^{\mathcal{O}(1)} = 3^t \cdot n^{\mathcal{O}(1)}$. Figure 1 shows an example where this situation occurs.

Our main result for the edge-deletion version is showing that there are *precisely two* algorithmic ideas that can improve the running time for $\text{LHOMED}(H)$: restricting the lists to incomparable sets and exploiting decompositions. Formally, let $i^\bullet(H)$ be the maximum of $i(H')$ taken over all induced undecomposable subgraphs H' of H that are *not* classified as polynomial-time solvable by Theorem 2, i.e., these are the graphs H that contain at least one of an irreflexive edge, 3 vertices with private neighbors, or 3 vertices with co-private neighbors.



■ **Figure 1** An example of a graph H that has a decomposition (A, B, C) , with $i(H) = k$ and $i^\bullet(H) = 3$. The a_i 's form an irreflexive independent set and the b_i 's form a reflexive clique. Every vertex a_i is adjacent to $\{b_i, \dots, b_{i+k-1}\}$, and $\{u_1, u_2, u_3\}$ is fully adjacent to every a_i and b_i . Observe that $i(H) \geq i(H[A]) \geq k$, as vertices a_1, \dots, a_k have incomparable neighborhoods. There is no irreflexive edge in $H[A]$, and it can be checked that there is no 3-element set with private or co-private neighbors, implying that $\text{LHOMED}(H[A])$ is polynomial-time solvable. But $\{u_1, u_2, u_3\}$ has private neighbors, making $\text{LHOMED}(H)$ NP-hard.

► **Theorem 5 (Main result for treewidth, edge deletion).** *Let H be a fixed graph that contains either an irreflexive edge, three vertices with private neighbors, or three vertices with co-private neighbors. Then $\text{LHOMED}(H)$ on n -vertex instances given with a tree decomposition of width t*

(a) *can be solved in time $i^\bullet(H)^t \cdot n^{\mathcal{O}(1)}$ and*

(b) *cannot be solved in time $(i^\bullet(H) - \varepsilon)^t \cdot n^{\mathcal{O}(1)}$ for any $\varepsilon > 0$, unless the SETH fails.*

For the lower bound of Theorem 7, it is sufficient to prove that $i^\bullet(H)$ is the correct base of the exponent if H is undecomposable. The flavor of the proof is similar to the proof of Theorem 6, but more involved. One reason for the extra complication is that vertex-deletion problems typically give us more power when designing gadgets in a reduction than edge-deletion problems. But beyond that, an inherent difficulty in the proof of Theorem 7 is that the proof needs to exploit somehow the fact that H is undecomposable. Therefore, we need to find an appropriate certificate that the graph is undecomposable and use this certificate in the gadget construction throughout the proof.

Parameterization by hub size. Esmer et al. [12] presented a new perspective on lower bounds parameterized by the width of the tree decomposition given in the input. It was shown that many of these lower bounds hold even if we consider a larger parameter. These results showed that for many problems hard instances do not have to use the full power of tree decompositions (not even of path decompositions), the real source of hardness is instances consisting of a large central “hub” connected to an unbounded number of constant-sized components.

Formally, we say that a set Q of vertices is a (σ, δ) -hub of G if every component of $G - Q$ has at most σ vertices and each such component is adjacent to at most δ vertices of Q in G . Observe that if a graph has a (σ, δ) -hub core of size p , then this can be turned into a tree decomposition of width less than $p + \sigma$. This shows that if a problem can be solved in time $c^t \cdot n^{\mathcal{O}(1)}$ given a tree decomposition of width t is given in the input, then for every fixed σ and δ , this problem can be solved in time $c^p \cdot n^{\mathcal{O}(1)}$ given a (σ, δ) -hub of size p is given in the input. Thus any lower bound ruling out the possibility of the latter type of algorithm for a given c also rules out the possibility of the former type of algorithm. Esmer et al. [12] showed

that for many fundamental problems, the previously known lower bounds parameterized by the width of the tree decomposition can be strengthened to parameterization by hub size. Following their work, we also present our lower bound results in such a stronger form¹.

► **Theorem 6 (Main result for hub size, vertex deletion).** *Let H be a fixed graph which contains either an irreflexive vertex or three pairwise non-adjacent reflexive vertices or an induced reflexive cycle on four or five vertices. Then for every $\varepsilon > 0$, there are $\sigma, \delta > 0$ such that $\text{LHOMVD}(H)$ with a (σ, δ) -hub of size p given in the input cannot be solved in time $(i(H) + 1 - \varepsilon)^p \cdot n^{\mathcal{O}(1)}$, unless the SETH fails.*

► **Theorem 7 (Main result for hub size, edge deletion).** *Let H be a fixed graph that contains either an irreflexive edge, three vertices with private neighbors, or three vertices with co-private neighbors. Then for every $\varepsilon > 0$, there are $\sigma, \delta > 0$ such that $\text{LHOMED}(H)$ on n -vertex instances with a (σ, δ) -hub of size p given in the input cannot be solved in time $(i^\bullet(H) - \varepsilon)^p \cdot n^{\mathcal{O}(1)}$, unless the SETH fails.*

Let us observe that the lower bounds in Theorems 6 and 7 imply the lower bounds in Theorems 3 and 5, respectively. We present these strengthened results in this paper because obtaining them did not require any extra effort: as we shall see, we simply need to use a stronger known lower bound as a starting point.

We prove all our lower bounds by reduction from two problems. In the q -COLORINGVD problem, given a graph G , the task is to remove the minimum number of vertices such that the resulting graph is q -colorable. The q -COLORINGED problem is similar, but here we need to remove the minimum number of edges instead. Tight lower bounds for these problems parameterized by the width of the tree decomposition are known [35, 22]. Recently, Esmer et al. [12] strengthened these results to parameterization by hub size.

► **Theorem 8 ([12]).** *For every $q \geq 1$ and $\varepsilon > 0$, there exist integers $\sigma, \delta \geq 1$ such that if there is an algorithm solving in time $(q + 1 - \varepsilon)^p \cdot n^{\mathcal{O}(1)}$ every n -vertex instance of q -COLORINGVD given with a (σ, δ) -hub of size at most p , then SETH fails.*

► **Theorem 9 ([12]).** *For every $q \geq 2$ and $\varepsilon > 0$, there are integers σ and δ such that if an algorithm solves in time $(q - \varepsilon)^p \cdot n^{\mathcal{O}(1)}$ every n -vertex instance of q -COLORINGED that is given with a (σ, δ) -hub of size p , then the SETH fails.*

Our reductions replace edges in a q -COLORINGVD or q -COLORINGED instance by constant-sized gadgets. One can observe that such a transformation has a small effect on treewidth and also on hub size (although might change σ and δ slightly). Thus we can use Theorems 8 and 9 in a transparent way to obtain the lower bounds in Theorems 6 and 7.

2 Technical Overview

In this section, we overview the most important technical ideas in our results. For clarity, we start with the discussion of the vertex-deletion version and then continue with the more complicated edge-deletion variant.

¹ An astute reader might wonder if the statements below cannot be strengthened by making σ and δ universal constants. These issues are discussed by Esmer et al. [12]; we refer to their work for more details.

2.1 Vertex-deletion version

We start with the vertex-deletion version, where both the P vs. NP-hard dichotomy and the complexity bounds for bounded-treewidth graphs are significantly easier to prove.

Equivalence of $\text{LHomVD}(H)$ with classic problems. We have seen earlier how VERTEX COVER, ODD CYCLE TRANSVERSAL, and VERTEX MULTIWAY CUT can be reduced to $\text{LHomVD}(H)$ for various graphs H . Let us briefly discuss reductions in the reverse direction. It is clear that $\text{LHomVD}(K_1)$ is actually equivalent to VERTEX COVER: if we remove those vertices that have empty lists, then the problem is precisely finding a vertex cover of minimum size. However, $\text{LHomVD}(K_2)$ seems to be more general than ODD CYCLE TRANSVERSAL: a list of size one can express that the vertex has to be on a certain side of the bipartition of $G - X$ (if the vertex is not removed). Therefore, $\text{LHomVD}(K_2)$ is slightly more general than ODD CYCLE TRANSVERSAL, and equivalent to an annotated generalization, where given G and two sets $L, R \subseteq V(G)$, the task is to find a set X of vertices of minimum size such that $G - X$ has a bipartition with R and L on different sides.

For VERTEX MULTIWAY CUT with undeletable terminals, we can reduce $\text{LHomVD}(H)$ (where H consists of k independent reflexive vertices w_1, \dots, w_k) to a multiway cut instance G' the following way. Given an instance (G, L) of $\text{LHomVD}(H)$, we obtain G' by first extending it with k terminals t_1, \dots, t_k . Then for every $v \in V(G)$, we introduce a clique of size $|L(v)|$ that is completely connected to v . We introduce a perfect matching between the vertices of this clique and the set of terminals that corresponds to the elements of $L(v)$. Therefore, in every solution of VERTEX MULTIWAY CUT, all but one vertex of each clique has to be deleted for sure. We can also assume that no more than $|L(v)| - 1$ vertices of the clique are deleted: if every vertex of the clique were deleted, then we can modify the solution by removing v instead. This means that if v is not deleted, then it is in the component of a terminal from $L(v)$. Therefore, it can be shown that there is a tight correspondence between the optimum cost of the $\text{LHomVD}(H)$ instance and the optimum cost of the VERTEX MULTIWAY CUT instance. We can also note that this transformation increases treewidth at most by an additive constant and if the original graph has a (σ, δ) -hub of size p , then the constructed graph has a $(\sigma(k + 1), \delta + k)$ -hub of size $p + k$. Therefore, we can state the following lower bound:

► **Theorem 10.** *For every $k \geq 3$ and $\varepsilon > 0$, there are $\sigma, \delta > 0$ such that VERTEX MULTIWAY CUT with k terminals with a (σ, δ) -hub of size p given in the input cannot be solved in time $(k + 1 - \varepsilon)^p \cdot n^{\mathcal{O}(1)}$, unless the SETH fails.*

Dichotomy for vertex deletion. If H contains an irreflexive vertex, then we have seen that VERTEX COVER can be reduced to $\text{LHomVD}(H)$. For reflexive H , the NP-hard cases of $\text{LHomVD}(H)$ can be easily established using the following alternative characterizations of the tractability condition:

► **Lemma 11.** *Let H be a reflexive graph. The following conditions are equivalent.*

1. $i(H) \leq 2$,
2. H does not contain three pairwise nonadjacent vertices, an induced four-cycle, nor an induced five-cycle,
3. H is an interval graph whose vertex set can be covered by two cliques.

So we need to prove that $\text{LHomVD}(H)$ is polynomial-time solvable if H is reflexive and $i(H) \leq 2$, and it is NP-hard for every other H . If H is reflexive and contains an induced four-cycle or an induced five-cycle, then already $\text{LHom}(H)$ is NP-hard [13]. If H contains three pairwise non-adjacent reflexive vertices, then we have seen that VERTEX MULTIWAY CUT with three (undeletable) terminals can be reduced to it.

39:10 List Homomorphisms by Deleting Edges and Vertices

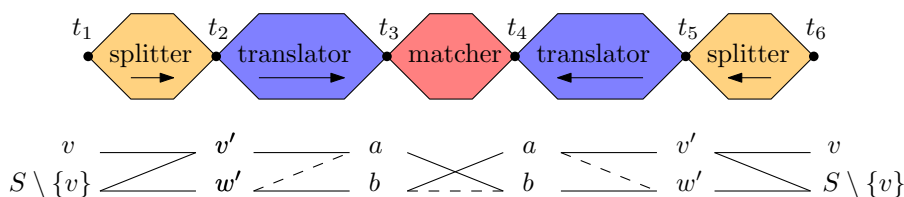
For the polynomial cases, by Lemma 11 we need to solve the problem only when H is an interval graph that can be partitioned into two cliques \mathcal{L} and \mathcal{R} . We can observe that in this case the neighborhoods of the vertices inside \mathcal{L} and \mathcal{R} form two chains. Thus if we assume that every list $L(v)$ is an incomparable set, then every list can contain at most two vertices: one from \mathcal{L} and one from \mathcal{R} .

We reduce $\text{LHOMVD}(H)$ to a minimum s - t cut problem. Note that using some form of minimum cut techniques cannot be avoided, as s - t MIN CUT can be reduced to the case when H consists of two independent reflexive vertices. Let V_L and V_R be the sets of vertices v where $L(v) \subseteq \mathcal{L}$ and $L(v) \subseteq \mathcal{R}$, respectively. If two vertices $u \notin V_R$ and $v \notin V_L$ are adjacent such that the vertex in $L(u) \cap \mathcal{L}$ is *not* adjacent to the vertex of $L(v) \cap \mathcal{R}$, then we add a directed edge from u to v . After a solution to $\text{LHOMVD}(H)$ is deleted, the remaining vertices can be partitioned into a “left” and “right” part according to whether they were mapped to \mathcal{L} or \mathcal{R} . The directed edge \vec{uv} represents the constraint that we cannot have u on the left part and v on the right part simultaneously. Then our problem is essentially reduced to deleting the minimum number of vertices such that there is no path from V_L to V_R .

Reduction using gadgets. To rule out algorithms with running time $(i(H) + 1 - \varepsilon)^t \cdot n^{\mathcal{O}(1)}$, we reduce from q -COLORINGVD for $q = i(H)$ to $\text{LHOMVD}(H)$. For this purpose, we take an incomparable set S of size $i(H)$ and construct gadgets that can express “not equal on S .” A gadget in this context means an instance of $\text{LHOMVD}(H)$ with a pair of distinguished vertices (x, y) . If neither of these vertices is removed, then they need to have different colors from S . Every solution has one of the $(|L(x)| + 1)(|L(y)| + 1)$ possible behaviors on (x, y) (mapping to $V(H)$ or deleting the vertices). Each behavior on (x, y) has some cost: the minimum number of vertex deletions we need to make inside the gadget to find a valid extension (note that this cost does not include the deletion of x and/or y). Our goal is to construct a gadget where $L(x) = L(y) = S$ and every behavior on (x, y) has the same cost α , except that mapping x and y to the same vertex of S extends only with cost strictly larger than α . We call such gadgets S -prohibitors. Then we can reduce q -COLORINGVD to $\text{LHOMVD}(H)$ by giving the list S to every vertex of the original graph G , and by replacing each of the m edges with a copy of the S -prohibitor gadget. Then it is easy to see that the original graph can be made q -colorable with k deletions if and only if the constructed $\text{LHOMVD}(H)$ instance has a solution with $\alpha \cdot |E(G)| + k$ deletions.

Constructing the prohibitor gadgets. A (v, S) -prohibitor gadget has two portals (x, y) with $L(x) = L(y) = S$, and every behavior has cost exactly α , except that it has cost strictly more than α when both x and y are mapped to v . By joining together (v, S) -prohibitors for every $v \in S$, we obtain the S -prohibitor defined in the previous paragraph.

The construction of the (v, S) -prohibitors is the core technical part of the proof of Theorem 6. The proof uses the fact that we are considering an NP-hard case of $\text{LHOMVD}(H)$ and hence one of the obstructions listed in Theorem 1 appears in the graph H (irreflexive vertex, three non-adjacent vertices, induced four-cycle, induced five-cycle). Some case analysis is needed based on, e.g., the type of the obstruction that appears, but in all cases the construction is surprisingly compact. We need three additional types of gadgets, which are put together in the way shown in Figure 2. We can interpret the two portals x and y of a gadget as input and output, respectively. Then setting a value on the input may “force” a single value on the output or “allow” some values on the output, meaning that these combinations on the input and the output can be extended with minimum cost.



■ **Figure 2** Construction of the (v, S) -prohibitor gadget. A dashed line means there is no edge between the two endpoints.

- *splitter*: if the input is assigned v , then the output is forced to v' ; if the input is from $S \setminus \{v\}$, then the output can be either v' or w' .
- *translator*: if the input is assigned v' , then the output is forced to a ; if the input is w' , then the output can be b .
- *matcher*: minimum cost can be achieved if one of the portals is assigned a and the other is b , but cannot be achieved if both portals are assigned a .

Suppose that vertices t_1 and t_6 are connected with these gadgets as in Figure 2. If both t_1 and t_6 are mapped to v , then the splitters force t_2 and t_5 to v' , the translators force t_3 and t_4 to a , which is incompatible with minimum cost of the matcher. On the other hand, if at least one of t_1 and t_6 is mapped to a vertex from $S \setminus \{v\}$, then the splitters allow us to map one of t_2 and t_5 to w' and the other to v' . In this case, the translators allow us to map one of t_3 to a and the other to b , which is now compatible with the minimum cost of the matcher.

The construction of the matcher is easy if we choose a and b to be non-adjacent vertices that are part of an obstruction. For example, if a, q, b, r is an induced reflexive four-cycle, then a path of 5 vertices with lists $\{a, b\} - \{q, b\} - \{q, r\} - \{b, r\} - \{b, a\}$ is an appropriate matcher. Indeed, the minimum cost 0 cannot be achieved if both endpoints are mapped to a .

The splitter can be constructed in the following way. Let us choose $w \in S \setminus \{v\}$. As v and w are incomparable, we can choose $v' \in \Gamma(v) \setminus \Gamma(w)$ and $w' \in \Gamma(w) \setminus \Gamma(v)$. Then the splitter is a four-vertex path with lists $S - V(H) \setminus \Gamma(v) - \{v\} - \{v', w'\}$. The gadget has cost at least 1, as at least one of the two inner vertices has to be deleted. If the first vertex is assigned v and the last vertex is assigned w' , then both inner vertices have to be deleted, making the cost 2.

Finally, a short case analysis gives a translator. Recall from the previous paragraph that $v' \in \Gamma(v) \setminus \Gamma(w)$ and $w' \in \Gamma(w) \setminus \Gamma(v)$, and, as a case, suppose that v' is not a neighbor of b . Then a six-vertex path with lists $\{v', w'\} - \{w\} - \{v'\} - \{b\} - \{a\} - \{a, b\}$ is a translator. At least two of the four inner vertices have to be deleted, meaning that the cost of this gadget is always at least 2. However, if we choose v' on the first vertex and b on the last vertex, then at least three of the inner vertices have to be deleted, raising the cost to 3.

2.2 Edge-deletion version

Let us turn our attention now to edge-deletion problems. While the high-level goals are similar to the vertex-deletion version, the proofs are necessarily more involved: there are two concepts, *bi-arc graphs* and *decompositions* that are relevant only for the edge-deletion version.

Equivalence of $\text{LHomED}(H)$ with classic problems. Earlier we have seen that MAX CUT and EDGE MULTIWAY CUT can be reduced to $\text{LHomED}(H)$ when H is an irreflexive edge or k independent reflexive vertices, respectively. Let us discuss reductions in the other direction.

39:12 List Homomorphisms by Deleting Edges and Vertices

Similarly to the case of ODD CYCLE TRANSVERSAL for vertex deletion, LHOMED(H) is actually equivalent to an annotated generalization of MAX CUT, where the two given sets L and R should be on the two sides of the bipartition. However, this annotated generalization is easy to reduce to the original MAX CUT problem. Introduce a new vertex w and for every $v \in L$, we connect w and v with $d(v)$ paths of length 2; for every $v \in R$, we connect w and v with $d(v)$ paths of length 3. We can verify that this extension forces every vertex of L to be on the same side as w and every vertex of R to be on the other side. Furthermore, this extension increases treewidth only by a constant and if the original graph has a (σ, δ) -hub of size p , then the constructed graph has a (σ', δ') -hub of size $p + 1$.

If H consists of k independent reflexive vertices w_1, \dots, w_k , then we can reduce an instance (G, L) of LHOMED(H) to EDGE MULTIWAY CUT the following way. Let us extend G to a graph G' by introducing k terminal vertices t_1, \dots, t_k . For every vertex $v \in V(G)$, let us introduce $d(v)$ paths of length 2 between v and t_i if $L(v)$ contains w_i . Suppose now that, in a solution of the multiway cut instance, vertex v is in the component of t_i . If w_i is not in $L(v)$, then the solution has to cut all the $|L(v)| \cdot d(v)$ paths. But then we could obtain a solution of the same size by removing all the $d(v)$ original edges incident to v and separating v from all but one terminal by breaking $(|L(v)| - 1) \cdot d(v)$ of the paths of length 2. Thus we can assume that vertex v is in the component of some terminal from $L(v)$, showing that we have a reduction from LHOMED(H) to EDGE MULTIWAY CUT. We can observe that this transformation increases treewidth at most by an additive constant. Therefore, we can obtain the following lower bound:

► **Theorem 12.** *For every $k \geq 3$ and $\varepsilon > 0$, there are $\sigma, \delta > 0$ such that EDGE MULTIWAY CUT with k terminals on n -vertex instances given with a tree decomposition of width at most t cannot be solved in time $(k - \varepsilon)^t \cdot n^{\mathcal{O}(1)}$, unless the SETH fails.*

Dichotomy for edge deletion. Feder, Hell, and Huang [16] proved that LHOM(H) is polynomial-time solvable if H is a bi-arc graph and NP-hard otherwise. Bi-arc graphs are defined by a geometric representation with two arcs on a circle; the precise definition appears in the full version. We start with an alternative characterization of the tractability criterion, which can be obtained using the forbidden subgraph characterization of bi-arc graphs [17, 16].

► **Lemma 13.** *The following two are equivalent:*

1. H does not contain an irreflexive edge, a 3-vertex set S with private neighbors, or a 3-vertex set S with co-private neighbors.
2. H is a bi-arc graph that does not contain an irreflexive edge or a 3-vertex set S with private neighbors.

With Lemma 13 in hand, the NP-hardness part of Theorem 2 follows easily. If H is not a bi-arc graph, then already LHOM(H) is NP-hard; if H contains an irreflexive edge or three vertices with private neighbors, then we can reduce from MAX CUT or EDGE MULTIWAY CUT with 3 terminals, respectively.

Similarly to the proof of Theorem 1, the polynomial-time part of Theorem 2 is based on a reduction to a flow problem. The fundamental difference is that in the edge-deletion case, there are graphs H such that $i(H) > 2$, but LHOMED(H) is polynomial-time solvable (an example of such a graph is $H[A]$ from Figure 1). Thus, even if we assume that the list of a vertex is an incomparable set, it can have size larger than 2. Therefore, a simple reduction to s - t MIN CUT where placing a vertex v on one of two sides of the cut corresponds to the choice between the two elements of the list $L(v)$ cannot work. Instead, we represent each vertex v with multiple vertices. Let $\ell = |L(v)|$. We represent vertex v with a directed path

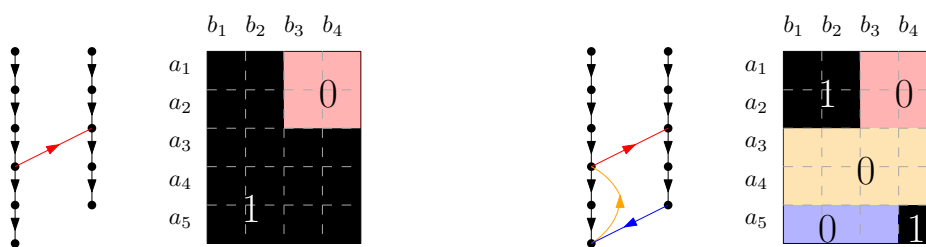


Figure 3 Representing the interaction of two vertices u and v with $L(v) = \{a_1, \dots, a_5\}$ and $L(u) = \{b_1, \dots, b_4\}$. Black areas denote ones in the interaction matrix.

on $\ell + 1$ vertices, where we enforce (with edges of large cost) that the first and last vertices are always on the right and left side of the cut. We imagine the edges of the path to be undeletable, for example, each edge has large weight, implying that a minimum weight s - t cut would not remove any of them. This means that the path has ℓ possible states in a minimum s - t cut: the only possibility is that for some $i \in [\ell]$, the first i vertices of the path are on the right side (the side corresponding to t), and the remaining $\ell + 1 - i$ vertices are on the left side (corresponding to s). Based on the geometric representation on the bi-arc graph H , we define an ordering $L(v) = \{a_1, \dots, a_\ell\}$ of each list. The idea is that assigning a_i to v corresponds to the state where the first i vertices of the path are on the right side of the cut.

To enforce this interpretation, whenever u and v are adjacent vertices in G , we introduce some edges between the paths representing u and v . These edges are introduced in a way that faithfully represents the *interaction matrix* of u and v , which is defined as follows. Let $L(u) = \{a_1, \dots, a_{\ell_u}\}$ and $L(v) = \{b_1, \dots, b_{\ell_v}\}$ in the ordering of the lists. The interaction matrix of u and v is a $|L(u)| \times |L(v)|$ matrix where the element in row i and column j is 1 if $a_i b_j \in E(H)$, and 0 otherwise.

Figure 3 (left) shows an example where $L(u) = \{a_1, \dots, a_5\}$, $L(v) = \{b_1, \dots, b_4\}$, and the interaction matrix is as shown in the figure, i.e., the 0s form a rectangle in the top-right corner. Then we introduce an edge from the fourth vertex of the path of u to the third vertex of the path of v . In the minimum s - t cut problem, this edge has to be removed whenever the tail of the edge is on the left side and the head of the edge is on the right side, which corresponds to assigning one of $\{a_1, a_2, a_3\}$ to u and one of $\{b_3, b_4\}$ to v . Therefore, we need to remove this edge if and only if the states of the two paths correspond to a 0 entry in the interaction matrix, that is, when the edge uv has to be removed since its image is not an edge of H . This means that this single edge indeed faithfully represents this particular interaction matrix.

Through a detailed analysis of bi-arc graphs without irreflexive edges and 3-vertex sets with private neighbors, we determine how interaction matrices can look like. It turns out that the 0s in the matrix can be partitioned into at most three “nice” rectangles: appearing in the top-right corner, appearing in the lower-left corner, or having full width $|L(v)|$ (see Figure 3, right). Each such nice rectangle can be represented by an edge, thus every interaction matrix can be represented by at most three edges such that in the solution we need to remove at most one of them.

Algorithms on bounded-treewidth graphs. As discussed on Page 6, if H has a decomposition as in Definition 4, then $\text{LHOMED}(H)$ can be reduced to an instance of $\text{LHOMED}(H_1)$ and an instance of $\text{LHOMED}(H_2)$, where $H_1 = H[A]$ and $H_2 = H[B \cup C]$. It follows that if we use the $i(H)^t \cdot n^{\mathcal{O}(1)}$ -time algorithm whenever H is undecomposable, then we obtain an $i^\bullet(H)^t \cdot n^{\mathcal{O}(1)}$ -

39:14 List Homomorphisms by Deleting Edges and Vertices

time algorithm for every H . Furthermore, proving that there is no $(i(H) - \varepsilon)^t \cdot n^{\mathcal{O}(1)}$ -time algorithm for undecomposable H proves that there is no $(i^\bullet(H) - \varepsilon)^t \cdot n^{\mathcal{O}(1)}$ -time algorithm for arbitrary H .

Reductions using gadgets. For the lower bound of Theorem 7, it is sufficient to prove the statement under the assumption that H is undecomposable, hence $i^\bullet(H) = i(H)$. For $q \geq 3$, we prove the lower bound by a reduction from q -COLORINGED, whose hardness was established in Theorem 9. As in the vertex-deletion case, we use gadgets that allow a straightforward reduction and the construction of these gadgets is the core technical part of the proof.

Here, a gadget is an instance with a set of distinguished vertices called portals. Defining the intended behavior of gadgets is neater in the edge-deletion case as the possibility of deleting portals does not complicate matters. For every assignment of the portals, the *cost* of the assignment is the minimum number of edges that needs to be deleted if we want to extend the assignment to the rest of the gadget. We can use the gadget to enforce that the assignment of the portals is one of minimum cost. Therefore, in order to reduce $\text{LHOMED}(K_q)$ to $\text{LHOMED}(H)$, we choose an incomparable set S of size q and design a gadget that has two portals (p_1, p_2) with $L(p_1) = L(p_2) = S$, and every assignment f with $f(p_1) \neq f(p_2)$ has cost exactly α , while every assignment with $f(p_1) = f(p_2)$ has cost β strictly more than α . We replace every edge of the original graph G with such a gadget. It is clear that the constructed $\text{LHOMED}(H)$ instance has a solution of cost $\alpha|E(G)|$ if and only if G is q -colorable.

Realizing relations. Let $c, d \in V(H)$ be two vertices and let $R \subseteq \{c, d\}^r$ be an arbitrary r -ary relation. We would like to prove a general statement saying that every such relation can be realized by some gadget: there is a gadget with r portals such that

- the list of each portal vertex is $\{c, d\}$,
- an assignment on the portal vertices has cost exactly α if it corresponds to a vector in R , and
- the cost of every other assignment is $\beta > \alpha$.

We show that if c and d are two vertices chosen from one of the obstructions appearing in Lemma 13(1) (irreflexive edges, three vertices with private or co-private neighbors), then such a gadget representing $R \subseteq \{c, d\}^r$ can indeed be constructed. Crucially, this requires to construct some gadget that realizes the “Not Equals” relation on $\{c, d\}$, i.e., $\text{NEQ} = \{(c, d), (d, c)\}$. With NEQ in hand, we use an earlier result from [12] for the list coloring problem, which shows that NEQ can be used to model arbitrary relations. Note that this is the point where we use the assumption that we are in the NP-hard case of Theorem 2 (which we definitively have to use at some point): We exploit the structure of an obstruction to model NEQ on two of its vertices.

Indicators. Our next goal is to construct *indicator gadgets*, defined as follows. The gadget has $\lambda + 1$ portals for some constant λ . Portal p has list S , and the remaining λ portals have list $\{c, d\}$. Let α be the minimum number of edge deletions that are needed in the gadget. We can think of p as the input and the rest of the portals as the outputs. If we are interested only in solutions where exactly α deletions are made inside the gadget, then assigning a value a to the input is compatible with some set $I(a) \subseteq \{c, d\}^\lambda$ of assignments on the outputs. The indicator gadget has two properties: (1) $I(a)$ is non-empty for any $a \in S$ and (2) $I(a) \cap I(b) = \emptyset$ for any two distinct $a, b \in S$.

If we can construct indicator gadgets, then we can construct the gadget needed to reduce from q -COLORING (that is, expressing $f(p_1) \neq f(p_2)$) in the following way. Let us introduce two copies of the indicator gadget on vertices $(p_1, u_1, \dots, u_\lambda)$ and on $(p_2, v_1, \dots, v_\lambda)$. We have $L(p_1) = L(p_2) = S$ and $L(u_i) = L(v_i) = \{c, d\}$ for $i \in [\lambda]$. Then we define an appropriate 2λ -ary relation $R \subseteq \{c, d\}^{2\lambda}$, realize it with a gadget as discussed above, and then put this gadget on the vertices $\{u_1, \dots, u_p, v_1, \dots, v_p\}$. We define the relation R such that it rules out for any $a \in S$ that the assignment on (u_1, \dots, u_λ) is from $I(a)$ and the assignment on (v_1, \dots, v_λ) is also from $I(a)$; as we can realize any relation R , we can certainly realize such a gadget. Then this gadgets enforces, for any a , that the value a cannot appear on both p_1 and p_2 simultaneously, but allows every other combination.

We construct indicator gadgets for $\lambda = |S|(|S| - 1)$. For every pair (a, b) of distinct vertices from S , we construct a subgadget with two portals (q_1, q_2) with $L(q_1) = S$ and $L(q_2) = \{a', b'\}$ for some $a', b' \in V(H)$, and satisfying the following:

1. assigning a on q_1 forces a' on q_2 .
2. assigning b on q_1 forces b' on q_2 .
3. for any $e \in S \setminus \{a, b\}$, assigning e on q_1 allows at least one of a' or b' on q_2 .

We construct $|S|(|S| - 1)$ such subgadgets – one for every distinct (a, b) . The construction of these subgadgets is fairly simple, but in general the pair (a', b') can be different for every pair (a, b) . If we were so lucky that every pair (a', b') is actually (c, d) , then we would be done with the construction of the indicator. In this case, we can simply join these $|S|(|S| - 1)$ subgadgets at q_1 to obtain a gadget with input q_1 and $|S|(|S| - 1)$ output vertices. Now it is clear that if we assign values a and b to the input, then they cannot be compatible with the same assignment on the output vertices: in the subgadget corresponding to pair (a, b) , value a on the input forces c on the output, while value b forces d on the output.

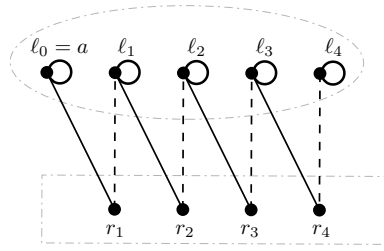
In general, however, we cannot expect (a', b') to be the same pair (c, d) for every choice of (a, b) . Therefore, the final component is a gadget that “moves” an arbitrary pair (a', b') to (c, d) .

Moving pairs. We say that there is an $(a, b) \rightarrow (c, d)$ *move* if there is a gadget with two portals (x, y) with $L(x) = \{a, b\}$, $L(y) = \{c, d\}$, and the following property: assigning a (resp., b) to x forces c (resp., d) on y . In most cases, it is not very important to us which of a and b is mapped to c or d , only the uniqueness of the mapping is important. Therefore, we introduce the notation $\{a, b\} \rightsquigarrow \{c, d\}$ *move* to mean either an $(a, b) \rightarrow (c, d)$ move or an $(a, b) \rightarrow (d, c)$ move. The main result is that if the graph H is undecomposable, then we can have such moves between any two pairs of incomparable vertices.

► **Lemma 14.** *Let H be an undecomposable graph. Let $\{a, b\}$ and $\{c, d\}$ be (not necessarily disjoint) 2-vertex incomparable sets in H . Then $\{a, b\} \rightsquigarrow \{c, d\}$.*

The assumption that H is undecomposable is essential here: one can observe that if there is a decomposition (A, B, C) and $a, b \in A$ and $c, d \in B$, then an $\{a, b\} \rightarrow \{c, d\}$ move cannot exist: intuitively, we cannot transmit information through the complete connection between A and B .

The first step of the proof is to show that such a move exists if the 2-vertex incomparable sets intersect: that is, there is a $\{a, b\} \rightsquigarrow \{a, c\}$ move whenever $\{a, b\}$ and $\{a, c\}$ are incomparable sets. This suggests defining the following auxiliary graph $\text{Aux}(H)$: the vertices of $\text{Aux}(H)$ correspond to 2-vertex incomparable sets, and two such vertices are connected if they represent pairs that intersect. Our main goal is showing that (a large part of) $\text{Aux}(H)$ is connected. As discussed above, the proof has to use the fact that H is undecomposable.



■ **Figure 4** An alternating path certifying that a is moved to B . Vertex l_4 is maximal.

We consider two cases depending on whether H is a *strong split graph* or not, that is, whether it can be partitioned into a reflexive clique and an irreflexive independent set. The way we can exploit the non-existence of decompositions depends on whether H is in this class or not.

Case I: strong split graphs. In the case of a strong split graph, the following algorithm can be used to detect if there is a non-trivial decomposition. Let us assume that H does not have universal or independent vertices. We say that a vertex is *maximal* if its neighborhood is inclusionwise maximal, that is, there is no vertex that is adjacent to a proper superset of the neighborhood. The key observation is that every maximal vertex has to be in part B of the decomposition. Therefore, we initially move every maximal vertex into B and move every other vertex to A . Then we repeatedly apply the following two steps as long as possible:

- If $v \in A$ is irreflexive and not adjacent to some vertex in B , then we move v into C .
- If $v \in A$ is reflexive and adjacent to C , then we move v into B .

It can be checked that the algorithm is correct: if it stops with a non-empty set A , then (A, B, C) is a valid decomposition. Thus the assumption that H has no decomposition implies that the algorithm moves every vertex to $B \cup C$.

Consider an incomparable pair $\{a, b\}$ that we want to move to $\{c, d\}$. It is sufficient to consider only the case where a and b are both reflexive. The algorithm eventually moves a to B , and there is a sequence of moves that certify this. That is, there is a sequence $\ell_0, r_1, \ell_1, r_2, \dots, r_k, \ell_k$ such that $\ell_0 = a$, ℓ_k is a maximal reflexive vertex, ℓ_i is a reflexive vertex adjacent to r_{i+1} , and r_i is an irreflexive vertex not adjacent to ℓ_i (see Figure 4). If we choose this alternating path certificate to be of minimal length, then ℓ_i and ℓ_{i+1} are incomparable: r_{i+1} and r_{i+2} are adjacent to exactly one of them. Therefore, the pairs $\{\ell_i, \ell_{i+1}\}$ and $\{\ell_{i+1}, \ell_{i+2}\}$ are adjacent in $\text{Aux}(H)$, implying that $\{a, b\}$ is in the same component of $\text{Aux}(H)$ as $\{\ell_{k-1}, \ell_k\}$. If q is some maximal vertex with a neighborhood distinct from ℓ_k , then $\{\ell_k, q\}$ is also incomparable, and it is adjacent to $\{\ell_{k-1}, \ell_k\}$. The conclusion is that every incomparable pair $\{a, b\}$ is in the same component as some pair $\{a', b'\}$ of incomparable *maximal* vertices. Therefore, it is sufficient to show that whenever $\{a', b'\}$ and $\{c', d'\}$ are two pairs of incomparable vertices such that a', b', c', d' are all maximal, then $\{a', b'\}$ and $\{c', d'\}$ are in the same component of $\text{Aux}(H)$. Then at least one of $\{a', d'\}$ or $\{a', c'\}$ is incomparable (depending on whether $\Gamma(a') = \Gamma(c')$ or not). Either of these pairs is adjacent to both $\{a', b'\}$ and $\{c', d'\}$.

Case II: graphs that are not strong split graphs. If H is not a strong split graph, then either it contains two adjacent irreflexive vertices, or two non-adjacent reflexive vertices. We can find a decomposition the following way. Initially, we

- put into A every reflexive vertex that is not adjacent to some other reflexive vertex, and
- put into A every irreflexive vertex that is adjacent to some other irreflexive vertex.

Then we repeat the following two steps as long as possible:

- If $v \notin A$ is irreflexive and adjacent to A , then we move v into A .
- If $v \notin A$ is reflexive and not adjacent to some vertex in A , then we move v into A .

Again, we can verify that if the algorithm stops without moving every vertex to A , then we have a non-trivial decomposition. Therefore, for every vertex a , the algorithm provides a sequence of moves that certifies that a has to be in part A of any decomposition. Similarly to the previous case, we can use such a (minimal) certificate to show that every (a, b) is in the same component of $\text{Aux}(H)$ as some (a', b') , where a' and b' are either adjacent irreflexive vertices or non-adjacent reflexive vertices. Therefore, all that is left to show is that if both (a', b') and (c', d') have this property, then they are in the same component of $\text{Aux}(H)$. This can be proved with a short case analysis.

References

- 1 Jan Bok, Richard C. Brewster, Tomáš Feder, Pavol Hell, and Nikola Jedlicková. List homomorphism problems for signed graphs. In Javier Esparza and Daniel Král', editors, *45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020, August 24-28, 2020, Prague, Czech Republic*, volume 170 of *LIPIcs*, pages 20:1–20:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.MFCS.2020.20.
- 2 Glencora Borradaile and Hung Le. Optimal dynamic program for r -domination problems over tree decompositions. In Jiong Guo and Danny Hermelin, editors, *11th International Symposium on Parameterized and Exact Computation, IPEC 2016, August 24-26, 2016, Aarhus, Denmark*, volume 63 of *LIPIcs*, pages 8:1–8:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.IPEC.2016.8.
- 3 Rajesh Chitnis, László Egri, and Dániel Marx. List H -coloring a graph by removing few vertices. *Algorithmica*, 78(1):110–146, 2017. doi:10.1007/s00453-016-0139-6.
- 4 Radu Curticapean, Nathan Lindzey, and Jesper Nederlof. A tight lower bound for counting Hamiltonian cycles via matrix rank. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1080–1099. SIAM, 2018. doi:10.1137/1.9781611975031.70.
- 5 Radu Curticapean and Dániel Marx. Tight conditional lower bounds for counting perfect matchings on graphs of bounded treewidth, cliquewidth, and genus. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1650–1669. SIAM, 2016. doi:10.1137/1.9781611974331.ch113.
- 6 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
- 7 Víctor Dalmau, László Egri, Pavol Hell, Benoît Larose, and Arash Rafiey. Descriptive complexity of list H -coloring problems in logspace: A refined dichotomy. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 487–498. IEEE Computer Society, 2015. doi:10.1109/LICS.2015.52.
- 8 Josep Díaz, Maria J. Serna, and Dimitrios M. Thilikos. Recent results on parameterized H -colorings. In Jaroslav Nešetřil and Peter Winkler, editors, *Graphs, Morphisms and Statistical Physics, Proceedings of a DIMACS Workshop, New Brunswick, New Jersey, USA, March 19-21, 2001*, volume 63 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 65–85. DIMACS/AMS, 2001. doi:10.1090/dimacs/063/05.
- 9 Josep Díaz, Maria J. Serna, and Dimitrios M. Thilikos. Counting h -colorings of partial k -trees. *Theor. Comput. Sci.*, 281(1-2):291–309, 2002. doi:10.1016/S0304-3975(02)00017-8.
- 10 László Egri, Pavol Hell, Benoît Larose, and Arash Rafiey. Space complexity of list H -colouring: a dichotomy. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM*

- Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 349–365. SIAM, 2014. doi:10.1137/1.9781611973402.26.
- 11 László Egri, Dániel Marx, and Paweł Rzażewski. Finding list homomorphisms from bounded-treewidth graphs to reflexive graphs: a complete complexity characterization. In Rolf Niedermeier and Brigitte Vallée, editors, *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018, February 28 to March 3, 2018, Caen, France*, volume 96 of *LIPICs*, pages 27:1–27:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICs.STACS.2018.27.
 - 12 Barış Can Esmer, Jacob Focke, Dániel Marx, and Paweł Rzażewski. Fundamental problems on bounded-treewidth graphs: The real source of hardness. *CoRR*, abs/2402.07331, 2024. doi:10.48550/arXiv.2402.07331.
 - 13 Tomas Feder and Pavol Hell. List homomorphisms to reflexive graphs. *Journal of Combinatorial Theory, Series B*, 72(2):236–250, 1998. doi:10.1006/jctb.1997.1812.
 - 14 Tomás Feder and Pavol Hell. Complexity of correspondence H -colourings. *Discret. Appl. Math.*, 281:235–245, 2020. doi:10.1016/j.dam.2019.11.005.
 - 15 Tomás Feder, Pavol Hell, and Jing Huang. List homomorphisms and circular arc graphs. *Comb.*, 19(4):487–505, 1999. doi:10.1007/s004939970003.
 - 16 Tomás Feder, Pavol Hell, and Jing Huang. Bi-arc graphs and the complexity of list homomorphisms. *J. Graph Theory*, 42(1):61–80, 2003. doi:10.1002/jgt.10073.
 - 17 Tomás Feder, Pavol Hell, and Jing Huang. The structure of bi-arc trees. *Discret. Math.*, 307(3-5):393–401, 2007. doi:10.1016/j.disc.2005.09.031.
 - 18 Tomás Feder, Pavol Hell, David G. Schell, and Juraj Stacho. Dichotomy for tree-structured trigraph list homomorphism problems. *Discret. Appl. Math.*, 159(12):1217–1224, 2011. doi:10.1016/j.dam.2011.04.005.
 - 19 Jacob Focke, Dániel Marx, Fionn Mc Inerney, Daniel Neuen, Govind S. Sankar, Philipp Schepper, and Philip Wellnitz. Tight complexity bounds for counting generalized dominating sets in bounded-treewidth graphs. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 3664–3683. SIAM, 2023. doi:10.1137/1.9781611977554.ch140.
 - 20 Jacob Focke, Dániel Marx, and Paweł Rzażewski. Counting list homomorphisms from graphs of bounded treewidth: tight complexity bounds. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 431–458. SIAM, 2022. doi:10.1137/1.9781611977073.22.
 - 21 Geña Hahn and Claude Tardif. *Graph homomorphisms: structure and symmetry*, pages 107–166. Springer Netherlands, Dordrecht, 1997. doi:10.1007/978-94-015-8937-6_4.
 - 22 Falko Hegerfeld and Stefan Kratsch. Towards exact structural thresholds for parameterized complexity. In Holger Dell and Jesper Nederlof, editors, *17th International Symposium on Parameterized and Exact Computation, IPEC 2022, September 7-9, 2022, Potsdam, Germany*, volume 249 of *LIPICs*, pages 17:1–17:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.IPEC.2022.17.
 - 23 Pavol Hell, Monaldo Mastrolilli, Mayssam Mohammadi Nevisi, and Arash Rafiey. Approximation of minimum cost homomorphisms. In Leah Epstein and Paolo Ferragina, editors, *Algorithms - ESA 2012 - 20th Annual European Symposium, Ljubljana, Slovenia, September 10-12, 2012. Proceedings*, volume 7501 of *Lecture Notes in Computer Science*, pages 587–598. Springer, 2012. doi:10.1007/978-3-642-33090-2_51.
 - 24 Pavol Hell and Jaroslav Nešetřil. On the complexity of H -coloring. *J. Comb. Theory, Ser. B*, 48(1):92–110, 1990. doi:10.1016/0095-8956(90)90132-J.
 - 25 Pavol Hell and Jaroslav Nešetřil. Counting list homomorphisms and graphs with bounded degrees. In Jaroslav Nešetřil and Peter Winkler, editors, *Graphs, Morphisms and Statistical Physics, Proceedings of a DIMACS Workshop, New Brunswick, New Jersey, USA, March*

- 19–21, 2001, volume 63 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 105–112. DIMACS/AMS, 2001. doi:10.1090/dimacs/063/08.
- 26 Pavol Hell and Jaroslav Nešetřil. *Graphs and homomorphisms*, volume 28 of *Oxford lecture series in mathematics and its applications*. Oxford University Press, 2004.
- 27 Pavol Hell and Jaroslav Nešetřil. Colouring, constraint satisfaction, and complexity. *Comput. Sci. Rev.*, 2(3):143–163, 2008. doi:10.1016/j.cosrev.2008.10.003.
- 28 Pavol Hell and Jaroslav Nešetřil. In praise of homomorphisms. *Comput. Sci. Rev.*, 40:100352, 2021. doi:10.1016/j.cosrev.2020.100352.
- 29 Pavol Hell and Arash Rafiey. The dichotomy of list homomorphisms for digraphs. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23–25, 2011*, pages 1703–1713. SIAM, 2011. doi:10.1137/1.9781611973082.131.
- 30 Pavol Hell and Arash Rafiey. The dichotomy of minimum cost homomorphism problems for digraphs. *SIAM J. Discret. Math.*, 26(4):1597–1608, 2012. doi:10.1137/100783856.
- 31 Ioannis Katsikarelis, Michael Lampis, and Vangelis Th. Paschos. Structural parameters, tight bounds, and approximation for (k, r) -center. *Discret. Appl. Math.*, 264:90–117, 2019. doi:10.1016/j.dam.2018.11.002.
- 32 Eun Jung Kim, Stefan Kratsch, Marcin Pilipczuk, and Magnus Wahlström. Directed flow-augmentation. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 938–947. ACM, 2022. doi:10.1145/3519935.3520018.
- 33 Vladimir Kolmogorov, Andrei A. Krokhin, and Michal Rolínek. The complexity of general-valued CSPs. *SIAM J. Comput.*, 46(3):1087–1110, 2017. doi:10.1137/16M1091836.
- 34 Vladimir Kolmogorov and Stanislav Živný. The complexity of conservative valued CSPs. *J. ACM*, 60(2):10:1–10:38, 2013. doi:10.1145/2450142.2450146.
- 35 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs of bounded treewidth are probably optimal. *ACM Trans. Algorithms*, 14(2):13:1–13:30, 2018. doi:10.1145/3170442.
- 36 Dániel Marx, Govind S. Sankar, and Philipp Schepper. Degrees and gaps: Tight complexity results of general factor problems parameterized by treewidth and cutwidth. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12–16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPICs*, pages 95:1–95:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICs.ICALP.2021.95.
- 37 Dániel Marx, Govind S. Sankar, and Philipp Schepper. Anti-factor is FPT parameterized by treewidth and list size (but counting is hard). In Holger Dell and Jesper Nederlof, editors, *17th International Symposium on Parameterized and Exact Computation, IPEC 2022, September 7–9, 2022, Potsdam, Germany*, volume 249 of *LIPICs*, pages 22:1–22:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.IPEC.2022.22.
- 38 Hermann A. Maurer, Arto Salomaa, and Derick Wood. Colorings and interpretations: a connection between graphs and grammar forms. *Discret. Appl. Math.*, 3(2):119–135, 1981. doi:10.1016/0166-218X(81)90037-8.
- 39 Hermann A. Maurer, Ivan Hal Sudborough, and Emo Welzl. On the complexity of the general coloring problem. *Inf. Control.*, 51(2):128–145, 1981. doi:10.1016/S0019-9958(81)90226-6.
- 40 Karolina Okrasa, Marta Piecyk, and Paweł Rzażewski. Full complexity classification of the list homomorphism problem for bounded-treewidth graphs. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *28th Annual European Symposium on Algorithms, ESA 2020, September 7–9, 2020, Pisa, Italy (Virtual Conference)*, volume 173 of *LIPICs*, pages 74:1–74:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ESA.2020.74.

39:20 List Homomorphisms by Deleting Edges and Vertices

- 41 Karolina Okrasa and Paweł Rzażewski. Fine-grained complexity of the graph homomorphism problem for bounded-treewidth graphs. *SIAM J. Comput.*, 50(2):487–508, 2021. doi:10.1137/20M1320146.
- 42 Johan Thapper and Stanislav Živný. The complexity of finite-valued CSPs. *J. ACM*, 63(4):37:1–37:33, 2016. doi:10.1145/2974019.