Full Complexity Classification of the List Homomorphism Problem for Bounded-Treewidth Graphs

Karolina Okrasa
Warsaw University of Technology, Faculty of Mathematics and Information Science, Poland
University of Warsaw, Institute of Informatics, Poland
k.okrasa@mini.pw.edu.pl

Marta Piecyk
Warsaw University of Technology, Faculty of Mathematics and Information Science, Poland
m.piecyk@mini.pw.edu.pl

Paweł Rzążewski
Warsaw University of Technology, Faculty of Mathematics and Information Science, Poland
University of Warsaw, Institute of Informatics, Poland
p.rzazewski@mini.pw.edu.pl

Abstract

A homomorphism from a graph G to a graph H is an edge-preserving mapping from $V(G)$ to $V(H)$. Let H be a fixed graph with possible loops. In the list homomorphism problem, denoted by LHom(H), we are given a graph G, whose every vertex v is assigned with a list $L(v)$ of vertices of H. We ask whether there exists a homomorphism h from G to H, which respects lists L, i.e., for every $v \in V(G)$ it holds that $h(v) \in L(v)$.

The complexity dichotomy for LHom(H) was proven by Feder, Hell, and Huang [JGT 2003]. The authors showed that the problem is polynomial-time solvable if H belongs to the class called *bi-arc graphs*, and for all other graphs H it is NP-complete.

We are interested in the complexity of the LHom(H) problem, parameterized by the treewidth of the input graph. This problem was investigated by Egri, Marx, and Rzążewski [STACS 2018], who obtained tight complexity bounds for the special case of reflexive graphs H, i.e., if every vertex has a loop.

In this paper we extend and generalize their results for *all* relevant graphs H, i.e., those, for which the LHom(H) problem is NP-hard. For every such H we find a constant $k = k(H)$, such that the LHom(H) problem on instances G with n vertices and treewidth t

- can be solved in time $k^t \cdot n^{\mathcal{O}(1)}$, provided that G is given along with a tree decomposition of width t,
- cannot be solved in time $(k - \varepsilon)^t \cdot n^{\mathcal{O}(1)}$, for any $\varepsilon > 0$, unless the SETH fails.

For some graphs H the value of $k(H)$ is much smaller than the trivial upper bound, i.e., $|V(H)|$.

Obtaining matching upper and lower bounds shows that the set of algorithmic tools that we have discovered cannot be extended in order to obtain faster algorithms for LHom(H) in bounded-treewidth graphs. Furthermore, neither the algorithm, nor the proof of the lower bound, is very specific to treewidth. We believe that they can be used for other variants of the LHom(H) problem, e.g. with different parameterizations.

2012 ACM Subject Classification Mathematics of computing → Graph coloring; Theory of computation → Problems, reductions and completeness; Theory of computation → Graph algorithms analysis; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases list homomorphisms, fine-grained complexity, SETH, treewidth

Digital Object Identifier 10.4230/LIPIcs.ESA.2020.74

Related Version A full version of the paper is available at `http://arxiv.org/abs/2006.11155`.

Funding *Karolina Okrasa*: This research is a part of a project that has received funding from

the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme Grant Agreement no. 714704.
Marta Piecyk: Supported by Polish National Science Centre grant no. 2018/31/D/ST6/00062.
Paweł Rzążewski: Supported by Polish National Science Centre grant no. 2018/31/D/ST6/00062.

Acknowledgements We are grateful to Daniél Marx and Lászlo Egri for much advice regarding the topic and to Kamil Szpojankowski for an inspiring discussion about almost all graphs.

1 Introduction

A popular line of research in studying computationally hard problems is to consider restricted instances, in order to understand the boundary between easy and hard cases. For example, most of natural problems can be efficiently solved on trees, using a bottom-up dynamic programming. This observation led to the definition of *treewidth*, which, informally speaking, measures how much a given graph resembles a tree. The notion of treewidth appears to be very natural and it was independently discovered by several authors in different contexts [2, 8, 37, 13].

For many problems, polynomial-time algorithms for graphs with bounded treewidth can be obtained by adapting the dynamic programming algorithms for trees. Most of these straightforward algorithms follow the same pattern, which was captured by the seminal meta-theorem by Courcelle [11]: he proved that each problem expressible in *monadic second order logic* (MSO_2) can be solved in time $f(\mathrm{tw}(G)) \cdot n^{\mathcal{O}(1)}$ on graphs G with n vertices and treewidth $\mathrm{tw}(G)$, where f is some function, depending on the MSO_2 formula describing the particular problem. As a consequence of this meta-theorem, in order to show that some problem Π is *fixed-parameter tractable* (FPT), parameterized by the treewidth, it is sufficient to show that Π can be described in a certain way.

The main problem with using the meta-theorem as a black-box is that the function f it produces is huge (non-elementary). We also know that this bound cannot be improved, if we want to keep the full generality of the statement [23]. Because of this, as the area of the so-called *fine-grained complexity* gained popularity [32], researchers turned back to studying particular problems, and asking about the best possible dependence on treewidth, i.e., the function f. This led to many exciting algorithmic results which significantly improved the naive dynamic programming approach [40, 6, 31].

There is a little caveat that applies to most of algorithms mentioned above: Usually we assume that the input graph is given with its tree decomposition, and the running time is expressed in terms of the width of this decomposition. This might be a serious drawback, since finding an optimal tree decomposition is NP-hard [1]. However, finding a tree decomposition of given width (if one exists) can be done in FPT time [4, 7], which is often sufficient. Since we are interested in the complexity of certain problems, parameterized by the treewidth, we will not discuss the time needed to find a decomposition. Thus we will assume that the input graph is given with its tree decomposition.

In parallel to improving the algorithms, many lower bounds were also developed [33, 37, 13]. Let us point out that the main assumption from the classical complexity theory, i.e., P $\neq$ NP, is too weak to provide any meaningful lower bounds in our setting. The most commonly used assumptions in the fine-grained complexity world, are the *Exponential-Time Hypothesis* (ETH) and the *Strong Exponential-Time Hypothesis* (SETH), both introduced by Impagliazzo and Paturi [27, 28]. Informally speaking, the ETH implies that 3-SAT with n variables cannot be solved in subexponential time, i.e., in time $2^{o(n)}$, and the SETH implies that CNF-SAT with n variables and m clauses cannot be solved in time $(2-\varepsilon)^n \cdot m^{\mathcal{O}(1)}$, for any $\varepsilon > 0$.

For example the straightforward dynamic programming algorithm for k-Coloring works in time $k^{\mathrm{tw}(G)} \cdot n^{\mathcal{O}(1)}$. As one of the first SETH-based lower bounds for problems parameterized by the treewidth, Lokshtanov, Marx, and Saurabh [33] proved that this bound is tight.

▶ **Theorem 1** (Lokshtanov, Marx, Saurabh [33]). *For any $k \geqslant 3$, there is no algorithm solving k-Coloring on a graph with n vertices and treewidth t in time $(k-\varepsilon)^t \cdot n^{\mathcal{O}(1)}$, unless the SETH fails.*

Graph homomorphisms. In this paper we are interested in extending Theorem 1 for one of possible generalizations of the k-Coloring problem. For graph G and H (both with possible loops on vertices), a *homomorphism* from G to H is a mapping $h\colon V(G) \to V(H)$, which preserves edges, i.e., for every edge xy of G it holds that $h(x)h(y) \in E(H)$. The graph H is called a *target*. If h is a homomorphism from G to H, we denote it by writing $h : G \to H$. We also write $G \to H$ to indicate that some homomorphism from G to H exists.

By Hom(H) we denote the problem of deciding if an instance graph G admits a homomorphism to H (usually we consider H a fixed graph, but we might also treat it as a part of the input). Observe that if $H = K_k$, then Hom(H) is equivalent to the k-Coloring problem. Because of that, homomorphisms to H are often called *H-colorings*. We will also refer to vertices of H as *colors*.

Let us briefly survey some results concerning the complexity of variants of the Hom(H) problem. For more, we refer the reader to the monograph by Hell and Nešetřil [25]. The complexity dichotomy for Hom(H) was shown by Hell and Nešetřil [26]: the problem is polynomial-time-solvable if H contains a vertex with a loop or is bipartite, and NP-complete for all other graphs H. Since then, many interesting results concerning the complexity of graph homomorphisms have appeared [22, 41, 38, 12, 14, 10]. The fine-grained complexity of the Hom(H) problem, parameterized by the treewidth of the input graph, was recently studied by Okrasa and Rzążewski [35]. They found tight SETH-bounds, conditioned on two conjectures from algebraic graph theory from early 2000s. As these conjectures remain wide open, we know no graph, for which the bounds from [35] do not apply.

A natural and interesting extension of the Hom(H) problem is its *list* version. In the *list homomorphism problem*, denoted by LHom(H), the input consists of a graph G and a *H-lists* L, which means that L is a function which assigns to each vertex of G a subset of vertices of H. We ask whether there is a homomorphism h from G to H, which respects lists L, i.e., for each $x \in V(G)$ it holds that $h(x) \in L(x)$. If h is such a list homomorphism, we denote it by $h : (G, L) \to H$. We also write $(G, L) \to H$ to indicate that some homomorphism $h : (G, L) \to H$ exists.

The complexity of the LHom(H) problem was shown in three steps. First, Feder and Hell [16] provided a classification for the case that H is *reflexive*, i.e., every vertex has a loop. They proved that if H is an interval graph, then the problem is polynomial-time solvable, and otherwise it is NP-complete. The next step was showing the complexity dichotomy for *irreflexive graphs* (i.e., with no loops). Feder, Hell, and Huang [17] proved that if H is bipartite and its complement is a circular-arc graph, then the problem is polynomial-time solvable, and otherwise it is NP-complete. Interestingly, bipartite graphs whose complement is circular-arc were studied independently by Trotter and Moore [39] in the context of some poset problems. Finally, Feder, Hell, and Huang [18] provided the full classification for general graph H: the polynomial cases appear to be *bi-arc graphs*, which are also defined in terms of some geometric representation. Let us now skip the exact definition of bi-arc graphs, and we will get back to it in Section 4.1.

Let us point out that in all three papers mentioned above, the polynomial-time algorithms for LHom(H) exploited the geometric representation of H. On the other hand, all hardness proofs followed the same pattern. First, for each "easy" class $\mathcal{C}$ (i.e., interval graphs, bipartite co-circular-arc graphs, and bi-arc graphs), the authors provided an alternative characterization in terms of forbidden subgraphs. In other words, they defined a (non-necessarily finite) family $\mathcal{F}$ of graphs, such that $H \in \mathcal{C}$ if and only if H does not contain any $F \in \mathcal{F}$ as an induced subgraph. Then, for each $F \in \mathcal{F}$, the authors showed that LHom(F) is NP-complete. Note that this is sufficient, as every "hard" graph H contains some $F \in \mathcal{F}$, and every instance of LHom(F) is also an instance of LHom(H), where no vertex from $V(H) - V(F)$ appears in any list.

If the input graph G is given with a tree decomposition of width $\text{tw}(G)$, then the straightforward dynamic programming solves the LHom(H) problem in time $|V(H)|^{\text{tw}(G)} \cdot |V(G)|^{\mathcal{O}(1)}$. The study of the fine-grained complexity of LHom(H), parameterized by the treewidth of the input graph, was initiated by Egri, Marx, and Rzążewski [15]. They provided the full complexity classification for reflexive graphs H, i.e., corresponding to the first step of the above-mentioned complexity dichotomy. The authors defined a new graph invariant, denoted by i^*, which is based on *incomparable sets* and the existence of a certain decomposition in H, and proved the following tight bounds.

▶ **Theorem 2** (Egri, Marx, Rzążewski [15]). *Let H be a fixed connected reflexive non-interval graph, and let $k = i^*(H)$. The LHom(H) problem on instances (G, L) with n vertices,*

(a) *can be solved in time $k^{\text{tw}(G)} \cdot n^{\mathcal{O}(1)}$, provided that an optimal tree decomposition of G is given,*

(b) *cannot be solved in time $(k - \varepsilon)^{\text{tw}(G)} \cdot n^{\mathcal{O}(1)}$ for any $\varepsilon > 0$, unless the SETH fails.*

Our results

In this paper we provide a full complexity classification of LHom(H), parameterized by the treewidth of an instance graph. Our results heavily extend the ones of Egri, Marx, Rzążewski [15] and generalize Theorem 2 to all relevant graphs H. Let us point out that instead of designing ad-hoc algorithms and reductions that are fine-tailored for a particular problem, we rather build a general framework that allows us to provide tight bounds for a natural and important family of problems.

Bipartite graphs H. We first deal with the case that H is bipartite (in particular, irreflexive), with bipartition classes X and Y. Recall that we are interested in graphs H, for which the LHom(H) problem is NP-hard, i.e., graphs that are not co-circular-arc graphs. Moreover, we consider only connected graphs H (as otherwise we can reduce to this case in polynomial time).

Let us present the high-level idea behind our algorithm for LHom(H). Consider an instance (G, L), such that G is connected, and let $n = |V(G)|$. We may assume that G is bipartite, as otherwise (G, L) is clearly a no-instance. Furthermore, in any homomorphism from G to H, each bipartition class of G is mapped to a different bipartition class of H. We can assume that this is already reflected in the lists (we might have to solve two independent instances).

The algorithm is based on two main ideas. First, observe that if H contains two vertices u, v, which are in the same bipartition class, and each neighbor of u is a neighbor of v, then we can always use v instead of u, unless this is forbidden by lists. Thus we might always assume that each list is an *incomparable set*, i.e., it does not contain two vertices u, v as above. By $i(H)$ we denote the size of a largest incomparable set contained in one bipartition class.

The second idea is related to a certain decomposition of H. By a *bipartite decomposition* we mean a partition of the vertex set of H into three subsets D, N, R, such that:

- at least one of sets $(D \cap X)$ and $(D \cap Y)$ has at least 2 elements,
- N is non-empty, induces a biclique in H, and separates D and R,
- the sets $(D \cap X) \cup (N \cap Y)$ and $(D \cap Y) \cup (N \cap X)$ induce bicliques in H.

We show that if H has a bipartite decomposition, then we can reduce solving an instance (G, L) of LHom(H) to solving several instances of LHom(H_1) and LHom(H_2), where H_1 is the subgraph of H induced by D, and H_2 is obtained from H by collapsing $D \cap X$ and $D \cap Y$ to single vertices.

This leads to the definition of $i^*(H)$ as the maximum value of $i(H')$ over all connected undecomposable induced subgraphs H' of H, which are not complements of a circular-arc graph (a graph is undecomposable if it has no bipartite decomposition). As our first result, we show that the algorithm exploiting decompositions recursively runs in time $i^*(H)^{\mathrm{tw}(G)} \cdot n^{\mathcal{O}(1)}$.

One might wonder whether some additional observations could be used to improve the algorithm. As our second result, we show that this is not possible, assuming the SETH. This means that unless something unexpected happens in complexity theory, our algorithmic toolbox allows to solve LHom(H), parameterized by the treewidth, as fast as possible. More formally, we show the following theorem, which fully classifies the complexity of LHom(H) for bipartite graphs H.

▶ **Theorem 3.** *Let H be a connected bipartite graph, whose complement is not a circular-arc graph, and let $k = i^*(H)$. Let G be a bipartite graph with n vertices and treewidth* $\mathrm{tw}(G)$.

(a) *Even if H is given as an input, the* LHom(H) *problem with instance (G, L) can be solved in time $k^{\mathrm{tw}(G)} \cdot (n \cdot |H|)^{\mathcal{O}(1)}$ for any lists L, provided that G is given with an optimal tree decomposition.*

(b) *Even if H is fixed, there is no algorithm that solves* LHom(H) *for every G and L in time $(k - \varepsilon)^{\mathrm{tw}(G)} \cdot n^{\mathcal{O}(1)}$ for any $\varepsilon > 0$, unless the SETH fails.*

Note that for Theorem 3 a), if H is not considered to be a constant, $n \cdot |H|$ is a natural measure of the size of an instance, as it is an upper estimate on the sum of sizes of all lists.

The main tool used in the proof of Theorem 3 b) is the following technical lemma.

▶ **Lemma 4** (Constructing a NEQ(S)-gadget)**.** *Let H be a connected, bipartite, undecomposable graph, whose complement is not a circular-arc graph. Let S be an incomparable set of $k \geqslant 2$ vertices of H, contained in one bipartition class. Then there exists a* NEQ(S)*-gadget, i.e., a graph F with H-lists L and two special vertices $x, x' \in V(F)$, such that $L(x) = L(x') = S$ and*

- *for any list homomorphism $h : (F, L) \to H$, it holds that $h(x) \neq h(x')$,*
- *for any distinct $s, s' \in S$ there is $h : (F, L) \to H$, such that $h(x) = s$ and $h(x') = s'$.*

Let us point out that the graph constructed in Lemma 4 can be seen as a *primitive-positive definition* of the inequality relation on S (see e.g. Bulatov [9, Section 2.1]). However, we prefer to present our results using purely combinatorial terms.

The proof of Lemma 4 is technically involved, but as soon as we have it, the proof of Theorem 3 b) is straightforward. Consider an instance G of k-Coloring, where $k = i^*(H)$. Let H' be a connected, undecomposable, induced subgraph of H, whose complement is not a circular-arc graph, and contains an incomparable set S of size k. We construct a graph G^* by replacing each edge uv of G with a copy of the NEQ(S)-gadget, given by Lemma 4 (invoked for H' and S), so that u is identified with x and v is identified with x'. By the properties of the gadget, we observe that G^* has a list homomorphism to H if and only if

G is a yes-instance of k-COLORING. Furthermore, the construction of the NEQ(S)-gadget depends on H only, and H is assumed fixed, so we conclude that $\mathrm{tw}(G^*) = \mathrm{tw}(G) + \mathcal{O}(1)$. Therefore the statement of Theorem 3 b) follows from Theorem 1.

General graphs H. Next, we move to the general case. We aim to reduce the problem to the bipartite case. The main idea comes from Feder, Hell, and Huang [18] who showed a close connection between the LHOM(H) problem and the LHOM(H^*) problem, where H^* is the *associated bipartite graph* of H, i.e., the bipartite graph with bipartition classes $\{v' \colon v \in V(H)\}$ and $\{v'' \colon v \in V(H)\}$, where $u'v'' \in E(H^*)$ if and only if $uv \in E(H)$. Let us point out that we can equivalently define H^* as a categorical (direct) product of H and K_2 [24].

We extend the definition of i^* to non-bipartite graphs by setting $i^*(H) := i^*(H^*)$. Let us point out that this is consistent with the definition for bipartite graphs, and with the definition of i^* for reflexive graphs, introduced by Egri, Marx, and Rzążewski [15]. We show the following theorem, fully classifying the complexity of LHOM(H), parameterized by the treewidth of the instance graph[1].

▶ **Theorem 5.** *Let H be a connected non-bi-arc graph (with possible loops), and let $k = i^*(H)$. Let G be a graph with n vertices and treewidth $\mathrm{tw}(G)$.*

(a) *Even if H is given as an input, the LHOM(H) problem with instance (G, L) can be solved in time $k^{\mathrm{tw}(G)} \cdot (n \cdot |H|)^{\mathcal{O}(1)}$ for any lists L, provided that G is given with an optimal tree decomposition.*

(b) (♠) *Even if H is fixed, there is no algorithm that solves LHOM(H) for every G and L in time $(k - \varepsilon)^{\mathrm{tw}(G)} \cdot n^{\mathcal{O}(1)}$ for any $\varepsilon > 0$, unless the SETH fails.*

As we mentioned before, both statements of Theorem 5 follow from the corresponding statements in Theorem 3. For the algorithmic part, we define certain decompositions of general graphs H and show that they coincide with bipartite decompositions H^*. This lets us reduce solving an instance (G, L) of LHOM(H) to solving some instances of LHOM(H^*). On the complexity side, the reduction is even more direct: we show that an algorithm solving the LHOM(H) problem on instances with treewidth t in time $(i^*(H) - \varepsilon)^t \cdot n^{\mathcal{O}(1)}$ could be used to solve the LHOM(H^*) problem on instances with treewidth t in time $(i^*(H^*) - \varepsilon)^t \cdot n^{\mathcal{O}(1)}$, thus contradicting Theorem 3 b).

We also analyze the complexity of LHOM(H) for *typical* graphs H, and prove the following.

▶ **Corollary 6 (♠).** *For almost all graphs H with possible loops the following holds. Even if H is fixed, there is no algorithm that solves LHOM(H) for every instance (G, L) in time $\mathcal{O}\left((|V(H)| - \varepsilon)^{\mathrm{tw}(G)} \cdot n^{\mathcal{O}(1)}\right)$ for any $\varepsilon > 0$, unless the SETH fails.*

Finally, we show how to generalize our approach of reducing instances of LHOM(H) to instances of LHOM(H'), where H' is undecomposable. We believe that this idea could be exploited to study the complexity of LHOM(H) in various regimes, e.g., for different parameterizations of input instances.

Comparison to the previous work. Let us briefly discuss similarities and differences between our work and previous, closely related results by Egri, Marx, and Rzążewski [15] (about the complexity of the LHOM(H) problem for reflexive H), and by Okrasa and Rzążewski [35] (about the complexity of the HOM(H) problem).

[1] The proofs of results marked with ♠ can be found in the full version of the paper [34].

At the high level, we follow the direction used by Egri *et al.* [15], but since we generalize their result to *all* relevant graphs H, the techniques become much more involved. The crucial idea was to reduce the problem to the bipartite case, and to define decompositions of general graphs that correspond to the decompositions of H^*. On the contrary, the case of reflexive graphs H is much more straightforward. In particular, there is just one type of decomposition that could be exploited algorithmically. Also, the structure of "hard" subgraphs is much simpler in this case, so the necessary gadgets are significantly easier to construct.

On the other hand, in order to prove hardness for the HOM(H) problem, Okrasa and Rzążewski [35] used mostly *algebraic tools* that are able to capture the global structure of a graph. In contrast, our proofs are purely combinatorial. Furthermore, we are able to provide the full complexity classification for all graphs H, while the results of [35] are conditioned on two twenty-year-old conjectures.

Notation

Let H be a graph. By $\text{comp}(H)$ we denote the set of connected components of H. For a vertex v, by $N(v)$ we denote the *open neighborhood* of v, i.e., the set of vertices adjacent to v (note that $v \in N(v)$ if and only if v has a loop). For a set $U \subseteq V(H)$, we define $N(U) := \bigcup_{u \in U} N(u) - U$ and $N[U] := \bigcup_{u \in U} N(u) \cup U$. If $U = \{u_1, \ldots, u_k\}$, we omit one pair of brackets and write $N(u_1, \ldots, u_k)$ (respectively $N[u_1, \ldots, u_k]$) instead of $N(\{u_1, \ldots, u_k\})$ (respectively $N[\{u_1, \ldots, u_k\}]$).

We say that two vertices x, y are *comparable* if $N(y) \subseteq N(x)$ or $N(x) \subseteq N(y)$. If two vertices are not comparable, we say that they are *incomparable*. A set of vertices is *incomparable* if all vertices are pairwise incomparable.

We say that a set $A \subseteq V(H)$ is *complete* to a set $B \subseteq V(H)$ if for every $a \in A$ and $b \in B$ the edge ab exists. On the other hand, A is *non-adjacent* to B if there are no edges with one endvertex in A and the other in B.

Let H be a bipartite graph, whose bipartition classes are denoted by X and Y. For a set $S \subseteq V(H)$ and $Z \in \{X, Y\}$, by S_Z we denote $S \cap Z$. For $A, B \subseteq V(H)$, we say that A is *bipartite-complete* to B if A_X is complete to B_Y and A_Y is complete to B_X.

A *walk* $\mathcal{P}$ is a sequence $\mathcal{P} = p_1, \ldots, p_\ell$ of vertices of H, such that $p_i p_{i+1} \in E(H)$, for every $i \in [\ell - 1]$.

2 Algorithm for bipartite target graphs

Observe that we might always assume that H is connected, as otherwise we can solve the problem for each connected component of H separately. Furthermore, without losing the generality we may assume certain properties of instances of LHOM(H) that we need to solve.

▶ **Observation 7** (♠). *Let (G, L) be an instance of LHOM(H), where H is connected and bipartite with bipartition classes X, Y. Without loss of generality, we might assume the following.*

1. *The graph G is connected and bipartite, with bipartition classes X_G and Y_G,*
2. *$\bigcup_{x \in X_G} L(x) \subseteq X$ and $\bigcup_{y \in Y_G} L(y) \subseteq Y$,*
3. *for each $x \in V(G)$, the set $L(x)$ is incomparable.*

An instance of LHOM(H) that respects conditions in Observation 7 is called *consistent*. From now on we will restrict ourselves to consistent instances. Let us introduce a graph parameter, which will play a crucial role in our investigations.

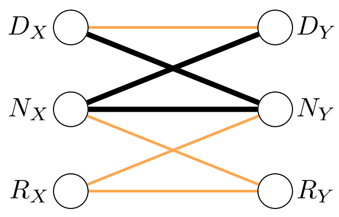


Figure 1 A schematic view of a bipartite decomposition. Disks correspond to independent sets of vertices. Thick black lines indicate that all possible edges between two sets exist, and thin orange lines depict edges that might exist, but do not have to. The lack of a line means that there are no edges between two sets.

▶ **Definition 8** ($i(H)$). *For a bipartite graph H, by $i(H)$ we denote the maximum size of an incomparable set in H, which is fully contained in one bipartition class.*

Clearly for every H we have $i(H) \leqslant |H|$. Note that by Observation 7 we obtain the following.

▶ **Corollary 9.** *Let (G, L) be a consistent instance of* LHom*(H), where H is bipartite. Then* $\max_{v \in V(G)} |L(v)| \leqslant i(H)$.

2.1 Decomposition of bipartite graphs

Throughout this section we assume that the target graph H is bipartite with bipartition classes X and Y. In particular, it has no loops. Our algorithm for LHom(H) is based on the existence of a certain decomposition of H.

▶ **Definition 10** (Bipartite decomposition). *A partition of $V(H)$ into an ordered triple of sets (D, N, R) is a* bipartite decomposition *if the following conditions are satisfied (see Figure 1).*

1. *N is non-empty and separates D and R,*
2. *$|D_X| \geqslant 2$ or $|D_Y| \geqslant 2$,*
3. *N induces a biclique in H,*
4. *D is bipartite-complete to N.*

Since so far we only consider bipartite decompositions, we will just call them decompositions. Later on we will introduce other types of decompositions and then the distinction will be important. If H admits a decomposition, then it is *decomposable*, otherwise it is *undecomposable*.

For a graph H with a decomposition (D, N, R), the *factors of the decomposition* are two graphs H_1, H_2 defined as follows. The graph H_1 is the subgraph of H induced by the set D. The graph H_2 is obtained in the following way. For $Z \in \{X, Y\}$, if D_Z is non-empty, then we contract it to a vertex d_Z. If there is at least one edge between the sets D_X and D_Y, we add the edge $d_X d_Y$.

Note that both H_1 and H_2 are proper induced subgraphs of H. For H_1 is follows directly from the definition and H_2 can be equivalently defined as a graph obtained from H by removing all but one vertex from D_X (if $D_X \neq \emptyset$) and all but one vertex from D_Y (if $D_Y \neq \emptyset$). We leave the vertices that are joined by an edge, provided that such a pair exists.

Now let us show how the bipartite decomposition can be used algorithmically. Let $T(H, n, t)$ denote an upper bound for the complexity of LHom(H) on instances with n vertices, given along a tree decomposition of width t. In the following lemma we do not assume that $|H|$ is a constant.

▶ **Lemma 11** (Bipartite decomposition lemma). *Let H be a bipartite graph with bipartition classes X and Y, whose complement is not a circular-arc graph, and suppose H has a bipartite decomposition with factors H_1, H_2. Assume that there are constants $\alpha \geqslant 1$, $c \geqslant 1$, and $d > 2$, such that $T(H_1, n, t) \leqslant \alpha \cdot c^t \cdot (n \cdot |H_1|)^d$ and $T(H_2, n, t) \leqslant \alpha \cdot c^t \cdot (n \cdot |H_2|)^d$. Then $T(H, n, t) \leqslant \alpha \cdot c^t \cdot (n \cdot |H|)^d$, if n is sufficiently large.*

Proof. Consider an instance (G, L) of LHom(H), recall that without loss of generality we may assume that it is consistent. Let the bipartition classes of G be X_G and Y_G and assume that $\bigcup_{x \in X_G} L(x) \subseteq X$ and $\bigcup_{y \in Y_G} L(y) \subseteq Y$.

Let (D, N, R) be a bipartite decomposition of H. We observe that for $Z \in \{X, Y\}$, and any two vertices $v \in D_Z, s \in N_Z$, we have $N(v) \subseteq N(s)$. Thus we may assume that no list contains both s and v. Let Q be the set of vertices of G which have at least one vertex from N in their lists.

▷ Claim 12. If there exists a list homomorphism $h\colon (G, L) \to H$, the image of each $C \in \text{comp}(G - Q)$ is entirely contained either in D or in R.

Proof. By the definition of comp$(G - Q)$, the image of C is disjoint with N. Suppose there exist $a, b \in C$, such that $h(a) = u \in D$ and $h(b) = r \in R$. Since C is connected, there exists an a-b-path P in C. The image of P is an u-r-walk in H. But since N separates D and R in H, there is a vertex of P, which is mapped to a vertex of N, a contradiction. ◁

Let us define lists L_1 as $L_1(x) := L(x) \cap D$, for every $x \in V(G) - Q$. For each $C \in \text{comp}(G - Q)$, we check if there exists a homomorphism $h_C\colon (C, L_1) \to H_1$. Let $\mathbb{C}_1$ be the set of those $C \in \text{comp}(G - Q)$, for which h_C exists. By Claim 12, we observe that if $C \notin \mathbb{C}_1$, then we can safely remove all vertices from D from the lists of vertices of C.

Now consider the graph H_2. Let $Z \in \{X, Y\}$ and let d_Z be the vertex to which the set D_Z is collapsed (if it exists). Let us define an instance (G, L_2) of the LHom(H_2) problem, where the lists L_2 are as follows. If $v \in Z_G$ is a vertex from some component of $\mathbb{C}_1$, then $L_2(v) := L(v) - D_Z \cup \{d_Z\}$ (note that in this case d_Z must exist). If v is a vertex from some component of comp$(G - Q) - \mathbb{C}_1$, then $L_2(v) := L(v) - D_Z$. Finally, if $v \in Q$, then $L_2(v) := L(v)$. Note that the image of each list is contained in $V(H_2)$. Moreover, note that $\bigcup_{x \in X_G} L_2(x) \subseteq R_X \cup N_X \cup \{d_X\}$ and $\bigcup_{y \in Y_G} L_2(y) \subseteq R_Y \cup N_Y \cup \{d_Y\}$.

▷ Claim 13. There is a list homomorphism $h : (G, L) \to H$ if an only if there is a list homomorphism $h' : (G, L_2) \to H_2$.

Proof. First, assume that $h : (G, L) \to H$ exists. Define $h' : V(G) \to V(H_2)$ in the following way:

$$h'(v) = \begin{cases} d_X & \text{if } h(v) \in D_X, \\ d_Y & \text{if } h(v) \in D_Y, \\ h(v) & \text{otherwise.} \end{cases}$$

Clearly h' is a homomorphism from G to H_2, we need to show that it also respects lists L_2. Suppose otherwise and let v be a vertex of G, such that $h'(v) \notin L_2(v)$. By symmetry, assume that $v \in X_G$, and thus $h'(v) \in X$. If $h'(v) \neq d_X$, then $h'(v) = h(v) \in (L(v) - D_X) \subseteq L_2(v)$. So suppose $h'(v) = d_X$ (and thus $h(v) \in D_X$) and $d_X \notin L_2(v)$. Observe that v cannot be a vertex from Q, since h maps v to a vertex of D_X and vertices from Q do not have any vertices of D in their lists. So the only case left is that v belongs to some connected component C of $G - Q$, which cannot be mapped to H_1. But then, by Claim 12, no vertex of C is mapped to any vertex of D, so $h(v) \notin D_X$, a contradiction.

Now suppose there exists a list homomorphism $h' : (G, L_2) \to H_2$. Define the following mapping h from $V(G)$ to $V(H)$. If $h'(v) \notin \{d_X, d_Y\}$, then we set $h(v) := h'(v)$. Otherwise, if $h'(v) \in \{d_X, d_Y\}$, then v is a vertex of some connected component $C \in \mathbb{C}_1$, and we define $h(v) := h_C(v)$. Clearly h preserves lists L: if $h(v) \notin D$, then $h(v) = h'(v) \in L_2(v) - \{d_X, d_Y\} \subseteq L(v)$; otherwise we use h_C, which preserves lists L by the definition.

Now suppose h does not preserve edges, so there are vertices $u \in X_G$ and $v \in Y_G$, such that uv is an edge of G and $h(u)h(v)$ is not an edge of H. If $h'(u) = d_X, h'(v) = d_Y$, or $h'(u) \neq d_X, h'(v) \neq d_Y$, then $h(u)h(v)$ must be an edge of H, otherwise we get a contradiction by the definitions of h_C (as since x and y are neighbors, they belong to the same C) and h', respectively. So, by symmetry, suppose $h'(u) = d_X$ and $h'(v) \neq d_Y$. But then we observe that $h(u) \in D_X$ and $h(v) = h'(v) \in N_Y$, since h' is a homomorphism. And because N_Y is complete to D_X, so $h(u)h(v)$ is an edge – a contradiction. ◁

Computing $\text{comp}(G - Q)$ can be done in time $\mathcal{O}(n \cdot |H| + n^2) = \mathcal{O}((n \cdot |H|)^2)$. Note that given a tree decomposition of G of width at most t, we can easily obtain a tree decomposition of each $C \in \text{comp}(G - Q)$ of width at most t. Computing h_C for all $C \in \text{comp}(G - Q)$ requires time at most

$$\sum_{C \in \text{comp}(G-Q)} T(H_1, |C|, t) \leqslant \sum_{C \in \text{comp}(G-Q)} \alpha \cdot c^t \cdot (|H_1| \cdot |C|)^d \leqslant \alpha \cdot c^t \cdot (|H_1| \cdot n)^d.$$

The estimation follows from the facts that $\sum_{C \in \text{comp}(G-Q)} |C| \leqslant n$, and n^d is superadditive with respect to n, i.e., $n_1^d + n_2^d \leqslant (n_1 + n_2)^d$. Computing lists L_2 can be performed in time $\mathcal{O}(|H| \cdot n)$. Finally, computing h' requires time $T(H_2, n, t) \leqslant \alpha \cdot c^t \cdot (|H_2| \cdot n)^d$. The total running time is therefore bounded by:

$$\mathcal{O}\left((n \cdot |H|)^2\right) + \alpha \cdot c^t \cdot (|H_1| \cdot n)^d + \mathcal{O}(|H| \cdot n) + \alpha \cdot c^t \cdot (|H_2| \cdot n)^d.$$

With a careful analysis one can verify that the above expression is bounded by $\alpha \cdot c^t \cdot (|H| \cdot n)^d$, provided that n is sufficiently large. ◀

2.2 Solving LHom(H) for bipartite targets

Let us define the main combinatorial invariant of the paper, $i^*(H)$:

▶ **Definition 14** ($i^*(H)$ for bipartite H). *Let H be a connected bipartite graph, whose complement is not a circular-arc graph. Define*

$$i^*(H) := \max\{i(H') \colon H' \text{ is an undecomposable, connected, induced subgraph of } H, \text{ whose complement is not a circular-arc graph}\}.$$

Observe that if H' is an induced subgraph of H, then $i(H') \leqslant i(H)$ and $i^*(H') \leqslant i^*(H)$, and thus $i^*(H) = i(H)$ for undecomposable H. Furthermore, we always have $i^*(H) \leqslant i(H) \leqslant |H|$.

Now we are ready to present an algorithm solving LHom(H), note that again we do not assume that $|H|$ is a constant. We present the following, slightly more general variant of Theorem 3 a), where we also do not assume that the tree decomposition of G is optimal.

▶ **Theorem 3' a).** *Let H be a connected bipartite graph (given as an input) and let (G, L) be an instance of LHom(H), where G has n vertices and is given along a tree decomposition of width t. Then there is an algorithm which decides whether $(G, L) \to H$ in time $\mathcal{O}\left(i^*(H)^t \cdot (n \cdot |H|)^{\mathcal{O}(1)}\right)$.*

Proof. Clearly we can assume that n is sufficiently large, as otherwise we can solve the problem in polynomial time by brute-force.

Observe that with H we can associate a *recursion tree* $\mathcal{R}$, whose nodes are labeled with induced subgraphs of H. The root, denoted by $node(H)$ corresponds to the whole graph H. If H is undecomposable or is a complement of a circular-arc graph, then the recursion tree has just one node. Otherwise H has a decomposition with factors H_1 and H_2, and then $node(H)$ has two children, $node(H_1)$ and $node(H_2)$, respectively. Recall that each factor has strictly fewer vertices than H, so we can construct $\mathcal{R}$ recursively. Clearly, each leaf of $\mathcal{R}$ is either the complement of a circular-arc graph (and thus the corresponding problem is polynomial-time solvable), or is an undecomposable induced subgraph of H. Note that a recursion tree may not be unique, as a graph may have more than one decomposition. However, the number of leaves is bounded by $\mathcal{O}(|H|)$ (actually, with a careful analysis we can show that it is at most $|H|-2$), so the total number of nodes is $\mathcal{O}(|H|)$. Furthermore, it can be shown that in time polynomial in H we can check if H is undecomposable, or find a decomposition (we show this in the full version of the paper). Since recognizing circular-arc graphs (and therefore of course their complements) is also polynomial-time solvable, we conclude that $\mathcal{R}$ can be constructed in time polynomial in $|H|$.

If H is the complement of a circular-arc graph, then we solve the problem in polynomial time [17]. If H is undecomposable, we run a standard dynamic programming algorithm on a tree decomposition of G (see [8, 5]). For each bag of the tree decomposition, and every partial list homomorphism from the graph induced by this bag to H we indicate whether this particular partial homomorphism can be extended to a list homomorphism of the graph induced by the subtree rooted at this bag. By Corollary 9, the size of each list $L(x)$ for $x \in V(G)$ is at most $i(H)$, thus the complexity of the algorithm is bounded by $\alpha \cdot i(H)^t \cdot (n \cdot |H|)^d$ for some constants α, d. We can assume that $d > 2$, as otherwise we can always increase it.

So suppose H is decomposable and let us show that we can solve the problem in time $\alpha \cdot i^*(H)^t \cdot (n \cdot |H|)^d$. Let $\mathcal{R}$ be a recursion tree of H and recall that its every leaf corresponds to an induced subgraph of H with strictly fewer vertices. Therefore, for any leaf node of $\mathcal{R}$, corresponding to the subgraph H' of H, we can solve every instance of LHom(H') with n vertices and a tree decomposition of width at most t in time $\alpha \cdot i(H')^t \cdot (n \cdot |H'|)^d \leqslant \alpha \cdot i^*(H)^t \cdot (n \cdot |H'|)^d$. Now, applying Lemma 11 in a bottom-up fashion, we conclude that we can solve LHom(H) in time $\alpha \cdot i^*(H)^t \cdot (n \cdot |H|)^d = \mathcal{O}(i^*(H)^t \cdot (n \cdot |H|)^{\mathcal{O}(1)})$. ◀

3 Hardness for bipartite target graphs

In this section we aim to prove Theorem 3 b). Actually we will show a version, which gives the lower bound parameterized by the pathwidth of G. Clearly such a statement is be stronger, as $\text{pw}(G) \geqslant \text{tw}(G)$. This corresponds to the pathwidth variant of Theorem 1, also shown by Lokshtanov, Marx, Saurabh [33].

▶ **Theorem 1'** (Lokshtanov, Marx, Saurabh [33]). *For any $k \geqslant 3$, there is no algorithm solving k-Coloring on a graph with n vertices and pathwidth t in time $(k-\varepsilon)^t \cdot n^{\mathcal{O}(1)}$, unless the SETH fails.*

Thus we show the following strengthening of Theorem 3 b).

▶ **Theorem 3' b).** *Let H be a fixed bipartite graph, whose complement is not a circular-arc graph. Unless the SETH fails, there is no algorithm that solves the LHom(H) problem on instances with n vertices and pathwidth t in time $(i^*(H)-\varepsilon)^t \cdot n^{\mathcal{O}(1)}$, for any $\varepsilon > 0$.*

In order to prove Theorem 3' b), it is sufficient to show the following.

▶ **Theorem 15.** *Let H be a fixed connected bipartite undecomposable graph, whose complement is not a circular-arc graph. Unless the SETH fails, there is no algorithm that solves the* LHom(H) *problem on instances with n vertices and pathwidth t in time $(i(H)-\varepsilon)^t \cdot n^{\mathcal{O}(1)}$, for any $\varepsilon > 0$.*

Let us show that Theorem 3' b) and Theorem 15 are equivalent.

(Theorem 15 → Theorem 3' b)). Assume the SETH and suppose that Theorem 15 holds and Theorem 3' b) fails. So there is a bipartite graph H, whose complement is not a circular-arc graph, and an algorithm A that solves LHom(H) in time $(i^*(H)-\varepsilon)^{\mathrm{pw}(G)} \cdot n^{\mathcal{O}(1)}$ for every input (G, L), assuming that G is given along with its optimal path decomposition.

Let H' be an undecomposable connected induced subgraph of H, whose complement is not a circular-arc graph, and $i(H') = i^*(H)$. Let (G, L') be an arbitrary instance of LHom(H'). Since H' is an induced subgraph of H, the instance (G, L') can be seen as an instance of LHom(H), where no vertex from $V(H) - V(H')$ appears in any list. The algorithm A solves this instance in time $(i^*(H)-\varepsilon)^{\mathrm{pw}(G)} \cdot n^{\mathcal{O}(1)} = (i(H')-\varepsilon)^{\mathrm{pw}(G)} \cdot n^{\mathcal{O}(1)}$, contradicting Theorem 15.

(Theorem 3' b) → Theorem 15). Assume the SETH and suppose Theorem 3' b) holds and Theorem 15 fails. So there is a connected bipartite undecomposable graph H, whose complement is not a circular-arc graph, and an algorithm A that solves LHom(H) in time $(i(H)-\varepsilon)^{\mathrm{pw}(G)} \cdot n^{\mathcal{O}(1)}$ for every input (G, L). But since H is connected, bipartite, and undecomposable, we have $i^*(H) = i(H)$, so algorithm A contradicts Theorem 3' b).

3.1 Hardness reduction

Let H be an undecomposable, bipartite graph, whose complement is not a circular-arc graph. First, we will show that the structure of H is sufficiently rich to express some basic relations.

For a k-ary relation $\mathrm{R}_k \subseteq V(H)^k$, by an R_k-gadget we mean a graph F with H-lists L and k specified vertices $v_1, v_2, \ldots, v_k$, called *interface*, such that for every $i \in [k]$ it holds that

$$\{f(v_1)f(v_2)\ldots f(v_k) \mid f : (F, L) \to H\} = \mathrm{R}_k.$$

In other words, the set of all possible colorings of the interface vertices that can be extended to a list H-coloring of the whole gadget is precisely the relation we are expressing. In the definition of an R_k-gadget we do not insist that interface vertices are pairwise different.

Let (α, β) be a fixed pair of vertices of H. First, we need to express a k-ary relation $\mathrm{OR}_k = \{\alpha, \beta\}^k - \{\alpha^k\}$ and a binary relation $\mathrm{NAND}_2 = \{\alpha\alpha, \alpha\beta, \beta\alpha\}$ for (α, β). To make the definitions more intuitive, note that we can assign logic values to vertices α, β in the following way: α is interpreted as false, and β is interpreted as true. Note that the relations OR_k and NAND_2 are symmetric with respect to interface vertices.

The other type of gadget which we need to introduce is distinguisher.

▶ **Definition 16** (Distinguisher). *Let S be an incomparable set in H and let (α, β) be a fixed pair of vertices of H, such that $\{\alpha, \beta\} \cup S$ is contained in one bipartition class of H. Let $a, b \in S$. A* distinguisher *gadget for (α, β) is a graph $D_{a/b}$ with two specified vertices x (called* input*) and y (called* output*), and H-lists L such that:*

(D1) *$L(x) = S$ and $L(y) = \{\alpha, \beta\}$,*

(D2) *there is a list homomorphism $\varphi_a : (D_{a/b}, L) \to H$, such that $\varphi_a(x) = a$ and $\varphi_a(y) = \alpha$,*

(D3) *there is a list homomorphism* $\varphi_b : (D_{a/b}, L) \to H$*, such that* $\varphi_b(x) = b$ *and* $\varphi_b(y) = \beta$*,*
(D4) *for any* $c \in \mathcal{S} - \{a, b\}$ *there is* $\varphi_c : (D_{a/b}, L) \to H$*, such that* $\varphi_c(x) = c$ *and* $\varphi_c(y) \in \{\alpha, \beta\}$*,*
(D5) *there is no list homomorphism* $\varphi : (D_{a/b}, L) \to H$*, such that* $\varphi(x) = a$ *and* $\varphi(y) = \beta$*.*

The existence of the OR$_k$-gadget, the NAND$_2$-gadget, and the distinguisher gadgets follows from the lemma.

▶ **Lemma 17** (♠)**.** *Let* H *be an undecomposable bipartite graph, whose complement is not a circular-arc graph and let* S *be an incomparable set in* H*,* $|S| \geqslant 2$*. Then there exists a pair of vertices* (α, β)*, such that* $S \cup \{\alpha, \beta\}$ *is contained in one bipartition class of* H *and:*
1. *for every* $k \geqslant 2$ *there exists an* OR$_k$*-gadget for* (α, β)*,*
2. *there exists a* NAND$_2$*-gadget for* (α, β)*,*
3. *for every pair* (a, b) *of distinct elements of* S *there exists a distinguisher* $D_{a/b}$ *for* (α, β)*.*

With Lemma 17 in hand we proceed to the construction of the NEQ(S)-gadget.

▶ **Lemma 4** (Constructing a NEQ(S)-gadget)**.** *Let* H *be a connected, bipartite, undecomposable graph, whose complement is not a circular-arc graph. Let* S *be an incomparable set of* $k \geqslant 2$ *vertices of* H*, contained in one bipartition class. Then there exists a* NEQ(S)*-gadget, i.e., a graph* F *with* H*-lists* L *and two special vertices* $x, x' \in V(F)$*, such that* $L(x) = L(x') = S$ *and*
- *for any list homomorphism* $h : (F, L) \to H$*, it holds that* $h(x) \neq h(x')$*,*
- *for any distinct* $s, s' \in S$ *there is* $h : (F, L) \to H$*, such that* $h(x) = s$ *and* $h(x') = s'$*.*

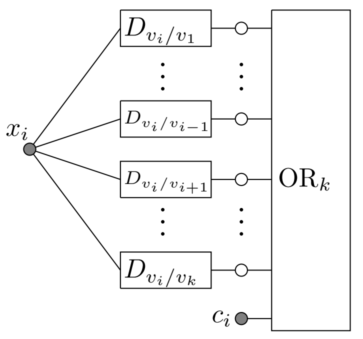
Proof. We denote the vertices of S by $v_1, v_2, \ldots, v_k$. We call Lemma 17 for S, let (α, β) be a pair of vertices given by this lemma. We will construct the NEQ(S)-gadget in three steps.

Step I. In this step we will show that for every $i \in [k]$ we can construct a graph I_i with two special vertices x_i and c_i and H-lists L, satisfying the following properties.
- $L(x_i) = S$ and $L(c_i) = \{\alpha, \beta\}$,
- for every list homomorphism $\varphi : (I_i, L) \to H$, if $\varphi(x_i) = v_i$, then $\varphi(c_i) = \beta$,
- for every $j \neq i$ there exists a list homomorphism $\varphi_j : (I_i, L) \to H$ such that $\varphi_j(x_i) = v_j$ and $\varphi_j(c_i) = \alpha$.

Let us fix any $i \in [k]$. For every $j \in [k] - \{i\}$ we take a distinguisher D_{v_i/v_j} given by Lemma 17 for $a = v_i, b = v_j$ with the input $x_{i,j}$ and the output $y_{i,j}$. We identify the vertices $x_{i,j}$, for all feasible j, to a single vertex x_i, and introduce a new vertex c_i. Then we add to our construction a copy of the OR$_k$ gadget and identify its k interface vertices with distinct elements of $\{c_i\} \cup \{y_{i,j}\}_{j \neq i}$. This completes the construction of I_i (see Figure 2, left).

Recall that the properties of the OR$_k$-gadget imply that every list homomorphism from I_i to H maps at least one of vertices in $\{c_i\} \cup \{y_{i,j}\}_{j \neq i}$ to β. By the property (D5) in Definition 16, for any $\varphi : (I_i, L) \to H$ with $\varphi(x_i) = v_i$, and for every $j \neq i$ it holds that $\varphi(y_{i,j}) = \alpha$. This in turn forces $\varphi(c_i) = \beta$.

On the other hand, by properties (D2), (D3), and (D4), for any $j \neq i$ there is a homomorphism $\varphi_j : (I_i, L) \to H$, such that $\varphi_j(x_i) = v_j$ and $\varphi(y_{i,j}) = \beta$, which allows us to set $\varphi_j(c_i) = \alpha$. So I_i satisfies all desired properties.



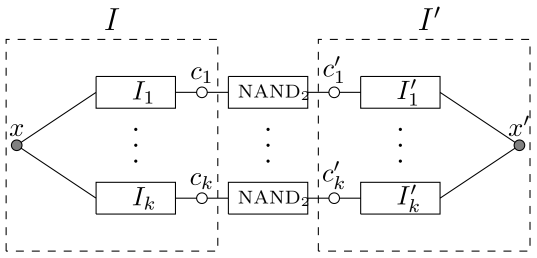


Figure 2 The graph I_i with special vertices x_i and c_i (left) and NEQ(S)-gadget with interface vertices x, x' (right).

Step II. In this step we will construct a graph I with H-lists L and special vertices $x, c_1, \ldots, c_k$, satisfying the following properties.

- $L(x) = S$ and $L(c_i) = \{\alpha, \beta\}$ for every $i \in [k]$,
- for every list homomorphism $\varphi : (I, L) \to H$, if $\varphi(x) = v_i$, then $\varphi(c_i) = \beta$.
- for every $i \in [k]$ there exists a list homomorphism $\varphi_i : (I, L) \to H$, such that $\varphi_i(x) = v_i$ and $\varphi_i(c_i) = \beta$, and $\varphi_i(c_j) = \alpha$ for every $j \in [k] - \{i\}$.

The graph I is constructed by introducing k gadgets $I_1, \ldots, I_k$, and identifying the vertices $x_1, \ldots, x_k$ into a single vertex x (see Figure 2, right). The desired properties of I follow directly from properties of I_i's.

Step III. Finally, we can construct a NEQ(S)-gadget. We introduce two copies of the gadget from the previous step, call them I and I' (we will use primes to denote the appropriate vertices of I'). For each $i \in [k]$, we introduce a NAND$_2$-gadget and identify its interface vertices with vertices c_i and c'_i (see Figure 2, right). Let us call such constructed graph F. We claim that F is a NEQ(S)-gadget, whose interface vertices are x and x'.

Clearly, $L(x) = L(x') = S$. Suppose that $\varphi : (F, L) \to H$ is a list homomorphism such that $\varphi(x) = \varphi(x') = v_i$. Then, by definition of I, we have $\varphi(c_i) = \varphi(c'_i) = \beta$, but this is impossible due to the properties of the NAND$_2$-gadget joining c_i and c'_i.

On the other hand, let us choose any distinct $v_i, v_j \in S$. We can color I according to the homomorphism φ_i, and I' according to the homomorphism φ_j (both φ_i and φ_j are defined in Step II). In particular this means that x is mapped to v_i, x' is mapped to v_j, c_i and c'_j are mapped to β, and all other vertices in $\{c_1, \ldots, c_k\} \cup \{c'_1, \ldots, c'_k\}$ are mapped to α. Since $i \neq j$, by the definition of a NAND$_2$-gadget, we can extend such defined mapping to all vertices of F. This completes the proof of the lemma. ◀

Now, equipped with Lemma 4, we can easily prove Theorem 15.

Proof of Theorem 15. Recall that H is an undecomposable bipartite graph, whose complement is not a circular-arc graph. Let S be the largest incomparable set in H, contained in one bipartition class. Let $k = |S|$, i.e., $k = i(H)$.

Observe that since the complement of H is not a circular-arc graph, we have that $k \geqslant 3$. Indeed, recall that H contains an obstruction, which is either an induced C_6, an induced C_8, or an asteroidal subgraph. Observe that all vertices from one bipartition class of C_6 or C_8 form an incomparable set of size at least 3. On the other hand, recall that a special edge asteroid contains at least three independent edges, so their appropriate endvertices form the desired incomparable set.

We reduce from k-Coloring, let G be an instance. Clearly we can assume that G is connected and has at least 3 vertices. We will construct a graph G^* with H-lists L and the following properties:

- $(G^*, L) \to H$ if and only if G is k-colorable,
- the number of vertices of G^* is at most $g(H) \cdot |E(G)|$ for some function g of H,
- the pathwidth of G^* is at most $g(H) + \mathrm{pw}(G)$,
- G^* can be constructed in time $(|V(G)|)^{\mathcal{O}(1)} \cdot g'(H)$ for some function g'.

Observe that this will be sufficient to prove the theorem. Indeed, suppose that for some $\varepsilon > 0$ we can solve LHom(H) in time $\mathcal{O}^*((k-\varepsilon)^t)$ on instances of pathwidth t. Let us observe that applying this algorithm to G^* gives an algorithm solving the k-Coloring problem on G in time

$$(k-\varepsilon)^{\mathrm{pw}(G^*)} \cdot |V(G^*)|^{\mathcal{O}(1)} \leqslant (k-\varepsilon)^{\mathrm{pw}(G)+g(H)} \cdot (g(H) \cdot |E(G)|)^{\mathcal{O}(1)} = (k-\varepsilon)^{\mathrm{pw}(G)} \cdot |V(G)|^{\mathcal{O}(1)},$$

where the last step follows since $|H|$ is a constant. Recall that by Theorem 1', the existence of such an algorithm for k-Coloring contradicts the SETH.

We start the construction of G^* with the vertex set of G. The lists of these vertices are set to S. Then, for each edge uv of G, we introduce a copy F_{uv} of the NEQ(S)-gadget introduced in Lemma 4. We identify the interface vertices of this gadget with u and v, respectively. This completes the construction of G^*. Let us show that it satisfies the properties (a)–(d).

Note that (a) follows directly from Lemma 4. Indeed, consider an edge uv of G. On one hand, for every list homomorphism $f : (G^*, L) \to H$ we have that $f(u) \neq f(v)$. On the other hand, mapping u and v to any distinct vertices from S can be extended to a homomorphism of the whole graph F_{uv}.

To show (b), recall that the number of vertices of each F_{uv} depends only on H, let it be $g(H)$. Every original vertex of G belongs to some gadget in G^*, so G^* contains at most $g(H) \cdot |E(G)|$ vertices.

Next, consider a path decomposition $\mathcal{T}$ of G of width $\mathrm{pw}(G)$, let the consecutive bags be $X_1, X_2, \ldots, X_\ell$. We extend $\mathcal{T}$ to a path decomposition $\mathcal{T}^*$ of G^* as follows: for every edge uv in G we choose one bag X_i such that $u, v \in X_i$, and we add a new bag $X_i' = X_i \cup V(F)$, which becomes the immediate successor of X_i. We repeat this for every edge, making sure that for X_i we can only choose the original bags coming from $\mathcal{T}$. Note that it might happen that we will insert several new bags in a row, if the same X_i was chosen for different edges, but this is not a problem. Is it straightforward to observe that $\mathcal{T}^*$ is a path decomposition of G^*, and the width of $\mathcal{T}^*$ is at most $g(H) + \mathrm{pw}(G)$. This proves (c).

Finally, it is straightforward to observe that the construction of G^* was performed in time polynomial in G (recall that we treat H as a constant-size graph). ◀

4 Algorithm for general target graphs

In this section we will generalize the invariant $i^*(H)$ and extend Theorem 3' a) to all relevant target graphs H. Let us start with a simple observation, which is an analogue of Observation 7.

▶ **Observation 18** (♠). *Let (G, L) be an instance of LHom(H). Without loss of generality we might assume the following.*

1. *The graph G is connected,*
2. *for each $x \in V(G)$, the set $L(x)$ is incomparable,*
3. *for each edge $xy \in E(G)$, for every $u \in L(x)$ there is $v \in L(y)$, such that $uv \in E(H)$.*

The high-level idea is to reduce the general case of $\text{LHom}(H)$ to the case, when the target is bipartite, and then use Theorem 3' a). For this, we will consider the so-called *associated instances*, introduced by Feder, Hell, and Huang [18] (see Section 4.1).

We will also separately consider some special graphs that we call *strong split graphs.* A graph H is a *strong split graph*, if its vertex set can be partitioned into two sets B and P, where B is independent and P induces a reflexive clique. We call (B, P) *the partition of* H. Note that the partition is unique: all vertices without loops must be in B and all vertices with loops must be in P.

4.1 Associated instances and clean homomorphisms

For a graph $G = (V, E)$, by G^* we denote the *associated bipartite graph*, defined as follows. The vertex set of G^* is the union of two independent sets: $\{x' \colon x \in V\}$ and $\{x'' \colon x \in V\}$. The vertices x' and y'' are adjacent if and only if $xy \in E$. Note that the edges of type $x'x''$ in G^* correspond to loops in G. The vertices x' and x'' are called *twins.* For any $W \subseteq V(G)$, we define two subsets of $V(G^*)$ as follows: $W' := \{x' : x \in W\}$ and $W'' := \{x'' : x \in W\}$.

Let (G, L) be an instance of $\text{LHom}(H)$. An *associated instance* is the instance (G^*, L^*) of $\text{LHom}(H^*)$, where L^* are *associated lists* defined as follows. For $x \in V(G)$, we set $L^*(x') = \{u' \colon u \in L(x)\}$ and $L^*(x'') = \{u'' \colon u \in L(x)\}$. Note that in the associated lists, the vertices appearing in the list of x' are precisely the twins of the vertices appearing in the list of x''. A homomorphism $f \colon (G^*, L^*) \to H^*$ is *clean* if it maps twins to twins, i.e., $f(x') = u'$ if and only if $f(x'') = u''$. The following simple observation was the crucial step of the proof of the complexity dichotomy for list homomorphisms, shown by Feder, Hell, and Huang [18]. We state it using slightly different language, which is more suitable for our purpose.

▶ **Proposition 19** (Feder, Hell, Huang [18]). *Let (G, L) be an instance of LHom(H). Then it is a yes-instance if and only if (G^*, L^*) admits a clean homomorphism to H^*.*

Let us point out that the restriction to clean homomorphisms is necessary for the equivalence. Indeed, consider for example $G = K_3$, $H = C_6$ and $L(v) = V(H)$ for every $v \in V(G)$. Clearly $(G, L) \not\to H$, however, we have $G^* \simeq C_6$ and $H^* \simeq 2C_6$, so $(G^*, L^*) \to H^*$.

Recall that if H is bipartite, then $\text{LHom}(H)$ is polynomial-time solvable if H is the complement of a circular-arc graph [17], and NP-complete otherwise. Feder, Hell, and Huang [18] proved the following dichotomy theorem.

▶ **Theorem 20** (Feder, Hell, Huang [18]). *Let H be an arbitrary graph (with loops allowed).*

1. *The LHom(H) problem is in P if H is a bi-arc graph, and NP-complete otherwise.*
2. *The graph H is a bi-arc-graph if and only if H^* is the complement of a circular-arc graph.*

So for our problem the interesting graphs H are those, for which H^* is not the complement of a circular-arc graph. In the observation below we summarize some properties of associated instances.

▶ **Observation 21** (♠). *Consider an instance (G, L) of LHom(H) and the associated instance (G^*, L^*) of LHom(H^*). Suppose that G is given along with a tree decomposition of width t.*

1. *For each $v \in V(G)$ and $u \in V(H)$, we have $u \in L(v)$ if and only if $u' \in L^*(v')$ if and only if $u'' \in L^*(v'')$. In particular, each list is contained in one bipartition class of H^*.*
2. *In polynomial time we can construct a tree decomposition $\mathcal{T}^*$ of G^* of width at most $2t$ with the property that for each $x \in V(G)$, each bag of $\mathcal{T}$ either contains both x', x'' or none of them.*

Observe that for bipartite H, the graph H^* consists of two disjoint copies of H, so clearly $i^*(H) = i^*(H^*)$. This motivates the following definition, generalizing Definition 14.

▶ **Definition 22** ($i^*(H)$)**.** *For a connected non-bi-arc graph H, we define $i^*(H) := i^*(H^*)$.*

4.2 Decompositions of generals target graphs

In this section we generalize the notion of decompositions of bipartite graphs, introduced in Section 2.1, to all graphs (with possible loops). The high-level idea is to define decompositions of H, so that they will correspond to bipartite decompositions of H^*. We consider the following three types of decompositions of a graph H. Note that unless stated explicitly, we do not insist that any of the defined sets is non-empty.

▶ **Definition 23** (F-decomposition)**.** *A partition of $V(H)$ into an ordered triple of sets (F, K, Z) is an* F-decomposition *if the following conditions are satisfied (see Figure 3, left).*

1. *K is non-empty and it separates F and Z,*
2. *$|F| \geqslant 2$,*
3. *K induces a reflexive clique,*
4. *F is complete to K.*

▶ **Definition 24** (BP-decomposition)**.** *A partition of $V(H)$ into an ordered five-tuple of sets (B, P, M, K, Z) is a* BP-decomposition *if the following conditions are satisfied (see Figure 3, middle).*

1. *$K \cup M$ is non-empty and separates $(P \cup B)$ and Z,*
2. *$|P| \geqslant 2$ or $|B| \geqslant 2$,*
3. *$K \cup P$ induces a reflexive clique and B is an independent set,*
4. *M is complete to $P \cup K$ and B is complete to K,*
5. *B is non-adjacent to M.*

▶ **Definition 25** (B-decomposition)**.** *A partition of $V(H)$ into an ordered six-tuple of sets $(B_1, B_2, K, M_1, M_2, Z)$ is a* B-decomposition *if the following conditions are satisfied (see Figure 3, right).*

1. *$K \cup M_1 \cup M_2$ is non-empty and it separates $(B_1 \cup B_2)$ and Z,*
2. *$|B_1| \geqslant 2$ or $|B_2| \geqslant 2$,*
3. *K induces a reflexive clique and each of B_1, B_2 is an independent set,*
4. *K is complete to $M_1 \cup M_2 \cup B_1 \cup B_2$, M_2 is complete to $M_1 \cup B_1$, and M_1 is complete to B_2,*
5. *B_1 is non-adjacent to M_1 and B_2 is non-adjacent to M_2.*

Observe that a graph H can have more than one decomposition: for example, if (B, P, M, K, Z) is an BP-decomposition of H, but $M = \emptyset$, then $(B \cup P, K, Z)$ is an F-decomposition of H.

For each kind of decomposition, we define its *factors* as the following pair of graphs (H_1, H_2).

for an F-decomposition: $H_1 = H[F]$ and H_2 is obtained from H by contracting F to a vertex f. It has a loop if and only if F is not an independent set.

for a BP-decomposition: $H_1 = H[B \cup P]$ and H_2 is obtained from H by contracting P and B respectively (if they are non-empty), to vertices p and b, such that p has a loop and b does not. Also, $pb \in E(H_2)$ if and only if there is any edge between P and B in H.

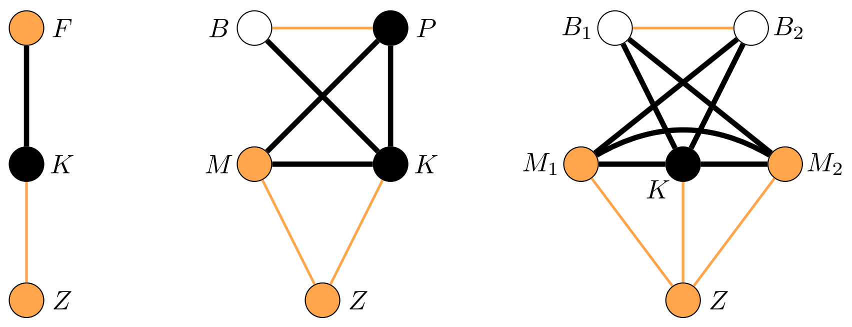


Figure 3 A schematic view of an F-decomposition (left), a BP-decomposition (middle), and a B-decomposition of H (right). Disks correspond to sets of vertices: white ones depict independent sets, black ones depict reflexive cliques, and orange ones depict arbitrary subgraphs. Similarly, thick black lines indicate that all possible edges between two sets exist, and thin orange lines depict edges that might exist, but do not have to. The lack of a line means that there are no edges between two sets.

for a B-decomposition: $H_1 = H[B_1 \cup B_2]$ and H_2 is obtained from H by contracting B_1 and B_2 respectively (if they are non-empty), to vertices b_1 and b_2 (without loops). Also, $b_1b_2 \in E(H_2)$ if and only if there is any edge between B_1 and B_2 in H.

It appears that if H is not a strong split graph, then the three types of decompositions defined above precisely correspond to bipartite decompositions of the associated bipartite graph H^*.

▶ **Lemma 26** (♠). *Let H be be a connected, non-bi-arc graph, which is not a strong split graph. Then H^* admits a bipartite decomposition if and only if H admits a B-, a BP-, or an F-decomposition.*

Let us point out that one application of a BP-decomposition or a B-decomposition corresponds to two consecutive applications of a bipartite decomposition in H^*. Consider the case of a BP-decomposition and the bipartite decomposition $(B' \cup P'', K' \cup M' \cup P' \cup K'', Z' \cup M'' \cup B'' \cup Z'')$ of H'. Note that after contracting B' to b' and P'' to p'', we still have a decomposition $(P' \cup B'', K' \cup K'' \cup M'' \cup \{p''\}, M' \cup \{b'\} \cup Z' \cup Z'')$ of the second factor $(H^*)_2$ of H^*. Similarly, in the case of a B-decomposition, we still have another bipartite decomposition of $(H^*)_2$, where the new set D is $B_1'' \cup B_2'$.

Note that in a BP-decomposition, a B-decomposition, and an F-decomposition, when F is an independent set or contains a vertex with a loop, the factors are always induced subgraphs of H. Indeed, we can equivalently obtain H_2 by removing certain vertices from H. In the last case of an F-decomposition, when F contains at least one edge and has only vertices without loops, H_2 is not an induced subgraph of H. We can equivalently define H_2 as the graph obtained by removing from F all but two adjacent vertices, and then replacing them with a vertex with a loop.

In the next lemma we consider a graph H' that was obtained from H by a series of decompositions (i.e., it is a factor of H, or a factor of a factor of H etc.). We show that even if H' is not an induced subgraph of H, the associated bipartite graph H'^* is still an induced subgraph of H^*.

▶ **Lemma 27** (♠). *Let H' be a graph obtained from H by a series of decompositions. Then H'^* is an induced subgraph of H^*.*

Finally, let us show an analogue of Lemma 11 for general graphs. Recall that $T(H,n,t)$ denotes an upper bound for the complexity of LHom(H) on instances with n vertices, given along with a tree decomposition of width t. Note that we do not assume that $|H|$ is a constant [2].

▶ **Lemma 28** (♠ General decomposition lemma). *Let H be a connected, non-bi-arc graph, and let Γ be a decomposition of H (i.e., Γ is either an F-, a BP-, or a B-decomposition) with factors H_1, H_2. If there exist constants $c \geqslant 1$ and $d > 2$ such that $T(H_1,n,t) = \mathcal{O}\left(c^t \cdot (n \cdot |H_1|)^d\right)$ and $T(H_2,n,t) = \mathcal{O}\left(c^t \cdot (n \cdot |H_2|)^d\right)$, then $T(H,n,t) = \mathcal{O}\left(c^t \cdot (n \cdot (|H|+2))^d\right)$.*

4.3 Solving LHom(H) for general target graphs

By Lemma 26, it is straightforward to observe the following.

▶ **Observation 29.** *If H is a connected, undecomposable, non-bi-arc graph, then $i^*(H)$ is the size of the largest incomparable set in H.*

In this section we sketch the proof of the following, slightly stronger version of Theorem 5 a), where the input tree decomposition is not assumed to be optimal.

▶ **Theorem 5' a).** *Let H be non-bi-arc graph. Even if H is given as an input, the LHom(H) problem with instance (G,L) can be solved in time $\mathcal{O}\left(i^*(H)^t \cdot (n \cdot |H|)^{\mathcal{O}(1)}\right)$ for any lists L, provided that G is given with its tree decomposition of width t.*

The main idea is similar to the one in the proof of Theorem 3' a): given an instance of LHom(H), we recursively decompose H into smaller subgraphs and reduce the initial instance to a number of instances of list homomorphism to these smaller subgraphs. Finally, we solve the problem for leaves of the recursion tree, and then, using Lemma 28 in a bottom-up fashion, we will compute the solution to the original instance. The only thing missing is how to solve the instances corresponding to leaves of the recursion tree. We describe this in the following results.

▶ **Corollary 30** (♠). *Let $\widehat{H}$ be a graph and let H be a strong split graph, which was obtained from $\widehat{H}$ by a series of decompositions. The LHom(H) problem with instance (G,L) with n vertices, given along with a tree decomposition of width t, can be solved in time $\mathcal{O}(i^*(\widehat{H})^t \cdot (n \cdot |H|)^{\mathcal{O}(1)})$.*

▶ **Lemma 31.** *For any graph H, any n-vertex instance (G,L) of LHom(H) can be solved in time $\mathcal{O}\left(i(H^*)^t \cdot (n \cdot |H|)^{\mathcal{O}(1)}\right)$, assuming a tree decomposition of G with width at most t is given.*

Proof. Let (G^*,L^*) be the associated instance of LHom(H^*). By Proposition 19, we know that $(G,L) \to H$ if and only if there is a clean homomorphism $(G^*,L^*) \to H^*$. We will focus on finding such a clean homomorphism.

First, recall that by Observation 18 (2) and Observation 21 (1), the instance (G^*,L^*) is consistent. So, by Corollary 9, the size of each list in L^* is at most $i(H^*)$. Moreover, by Observation 21 (1), for every $x \in V(G)$, the vertices in $L^*(x')$ are precisely the twins

[2] Let us point out that in the full version of the paper we actually prove three variants of Lemma 28, one for each type of decomposition. Furthermore, in the statements of these three lemmas we are more explicit about the constants, to make sure that they do not increase multiplicatively. This is necessary to avoid introducing factors exponential in $|H|$ in the proof of Theorem 5 a).

of vertices in $L^*(x'')$. Finally, by Observation 21 (2), in polynomial time we can obtain a tree decomposition $\mathcal{T}^*$ of G^* with width at most $2t$, in which vertices of G^* come in pairs: whenever any bag contains x', it also contains x''.

Consider the straightforward dynamic programming algorithm for $\textsc{LHom}(H^*)$, using the tree decomposition $\mathcal{T}^*$ of G^*. We observe that since we are looking for a clean homomorphism, we do not need to remember partial solutions, in which the colors of twins do not agree. Thus, even though the size of each bag of $\mathcal{T}^*$ is at most $2t$, the number of partial colorings we need to consider is bounded by $(\max_{x \in V(G^*)} |L^*(x)|)^t \leqslant i(H^*)^t$. So the claim follows. ◀

Finally, let us wrap everything up and prove Theorem 5' a).

Proof of Theorem 5' a). Let (G, L) be an instance of $\textsc{LHom}(H)$, where G has n vertices and is given with its tree decomposition of width at most t. Again, we can assume that n is sufficiently large, as otherwise we can solve the problem by brute-force. We proceed as in the Theorem 3' a). We consider a recursion tree $\mathcal{R}$, obtained by decomposing H recursively. For each node corresponding to some graph H', we construct its children recursively, unless none of the following happens (i) H' is a bi-arc graph, (ii) H' is bipartite, (iii) H' is a strong split graph, or (iv) H' is undecomposable.

We compute the solutions in a bottom-up fashion. First, consider a leaf of the recursion tree, let the corresponding target graph for this node of $\mathcal{R}$ be H'. If H' is a bi-arc graph, we can solve the problem in polynomial time. If H' is bipartite, we solve the problem in time $\beta \cdot i^*(H') \cdot (n \cdot |H'|)^{d_1} \leqslant \beta \cdot i^*(H) \cdot (n \cdot |H'|)^{d_1}$ for some constants β and d_1, using the algorithm from Theorem 3' a). If H' is a strong split graph, we can solve the problem in time $\gamma \cdot i^*(H) \cdot (n \cdot |H'|)^{d_2}$, for constants γ, d_2, using the algorithm from Corollary 30.

So finally consider the remaining case, i.e., that H' is connected, non-bi-arc, non-bipartite, undecomposable, which is not a strong split graph. Furthermore we know that H' was obtained from H by a sequence of decompositions. Recall that by Lemma 31, we can solve the instances of $\textsc{LHom}(H')$ in time $\delta \cdot i(H'^*)^t \cdot (n \cdot |H'|)^{d_3}$, for some constants δ, d_3. Let us consider the graph H'^*, by Lemma 27 we know that H'^* is an induced subgraph of H^*. Also, H'^* is either connected (if H' is non-bipartite), or consists of two disjoint copies of H' (if H' is bipartite). Moreover, by Lemma 26, we observe that H'^* is undecomposable. Thus, by the definition of $i^*(H)$, we observe that

$$\begin{aligned} i(H'^*) \leqslant \max\{i(H'')\colon & H'' \text{ is an undecomposable, connected, induced subgraph of } H^*, \\ & \text{whose complement is not a circular-arc graph}\} = i^*(H). \end{aligned}$$

So the algorithm from Lemma 31 solves the instances corresponding to leaves of $\mathcal{R}$ in time $\delta \cdot i^*(H)^t \cdot (n \cdot |H'|)^{d_3}$. Define $\alpha := \max(2\beta, \gamma, \delta)$ and $d := \max(d_1, d_2, d_3, 3)$ [3]. By applying Lemma 28 for every non-leaf node of $\mathcal{R}$ in a bottom-up fashion, we conclude that we can solve $\textsc{LHom}(H)$ in time $\alpha \cdot i^*(H)^t \cdot (n \cdot |H|)^d = \mathcal{O}\left(i^*(H)^t \cdot (n \cdot |H|)^{\mathcal{O}(1)}\right)$, which completes the proof. ◀

[3] The choice of these constants follows from the full statement of Lemma 28, refined for each of the decompositions.

5 Conclusion

Let us conclude the paper with some side remarks and pointing out several open problems.

5.1 Special cases: reflexive and irreflexive graphs

Recall that the crucial tool for our algorithmic results were the decompositions of a connected graph H, introduced in Section 2.1 (for bipartite graphs H) and in Section 4.2 (for general graphs H). Let us analyze how the general decompositions behave in two natural special cases: if H is either a reflexive or an irreflexive graph. We use the notation introduced in Definition 23, Definition 24, and Definition 25.

First we consider the case that H is reflexive, i.e., every vertex of H has a loop. Let us point out that a B-decomposition cannot occur in this case, as the sets B_1, B_2 are empty. In case of an F-decomposition we obtain exactly the decomposition defined by Egri et al. [15, Lemma 8]. Finally, in the case of a BP-decomposition, note that the set B is empty and therefore each vertex in P has exactly the same neighborhood. Thus the total contribution of the vertices in P to $i^*(H)$ is at most 1. Therefore the only type of decomposition that can be algorithmically exploited in reflexive graph is the F-decomposition, as observed by Egri et al. [15].

Now let us consider the case that H is irreflexive, i.e., no vertex of H has a loop. Observe that the sets K and P are reflexive cliques, so they are empty in our case. Thus BP-decompositions and F-decompositions do not occur in this case (recall that H is connected). Therefore the only possibility left is a B-decomposition, in which the set K is empty. Let us point out that this decomposition is very similar to the bipartite decomposition, in particular, the graph H_1 is bipartite (while H_2 might be non-bipartite). This gives even more evidence that the case of bipartite graphs H is a crucial step to understanding the complexity of the LHom(H) problem. Actually, if H is bipartite, then the B-decomposition turns out to be equivalent to the bipartite decomposition introduced in Section 2.1.

5.2 Generalized algorithm

We believe that the decompositions that we discovered can be used for many problems concerning the complexity of variants of the LHom(H) problem, e.g., for various other parameterizations. Let us point out that in the proofs of Lemma 11 and Lemma 28, we did not really require that the running time is of the form $\mathcal{O}(c^{\text{tw}(G)} \cdot (n \cdot |H|)^d)$, for a constant c.

Moreover, recall that in each variant of the decomposition lemma, all instances for which we computed partial results were induced subgraphs of G, the original instance. This motivates the following, generalized statement. For a graph G, and a graph H, let $T(G, H, n)$ be an upper bound for the complexity of the LHom(H) problem for instances of size n, which are induced subgraphs of G.

▶ **Corollary 32.** *Let H be a connected, non-bi-arc graph. Let $\mathcal{R}$ be its recursion tree and let $\mathcal{H}$ be the set of graphs associated with leaves of $\mathcal{R}$. Consider an instance (G, L) of LHom(H) with n vertices. Suppose that that for each $H' \in \mathcal{H}$ it holds that $T(G, H', n') \leqslant f(n', H')$, where f is superadditive with respect to its first argument. Then (G, L) can be solved in time $\mathcal{O}\left(\sum_{H' \in \mathcal{H}} f(n, H') + n^2 \cdot |H|^3\right)$.*

Sketch of proof. The proof is analogous to the proofs of Theorem 3' a) and Theorem 5' a). Recall that we can compute the instances associated with the leaves of $\mathcal{R}$, and then proceed in a bottom-up fashion. Even though there might be more than one instance associated with a leaf node, we observe that their numbers of vertices sum up to n , so by the superadditivity of f we can bound the running time related to each leaf node, associated with H', by $f(n, H')$.

Now let us consider an internal node of $\mathcal{R}$, it is associated with some subgraph H' of H. Recall that the computation for this node consists of the computations for child nodes and $\mathcal{O}(n^2|H'|^2)$ additional steps. Since the total number of nodes of $\mathcal{R}$ is $\mathcal{O}(|H|)$, we can bound the total running time for the root node (i.e., time needed to solve (G, L)) by $\mathcal{O}\left(\sum_{H' \in \mathcal{H}} f(n, H') + n^2 \cdot |H|^3\right)$. ◀

5.3 Further research directions

In this paper we have shown tight complexity bounds for the list homomorphism problem, parameterized by the treewidth of the instance graph.

A very natural question, mentioned also in [15], is to provide analogous results for the non-list variant of the problem, denoted by Hom(H). As we already mentioned, Okrasa and Rzążewski were able to provide tight bounds for the complexity of Hom(H), assuming two conjectures from algebraic graph theory from early 2000s [35]. It would be very interesting to strengthen these results, either by proving the mentioned two conjectures, or by providing a different reduction.

Another interesting direction is to consider one of many other variants of the graph homomorphism problem. Let us mention one, i.e., *locally surjectve homomorphism*, denoted by LSHom(H). In this problem we ask for a homomorphism from an instance graph G to the target graph H, which is surjective on each neighborhood. In other words, if we map some vertex $x \in V(G)$ to some vertex $v \in V(H)$, then every neighbor of v must appear as a color of some neighbor of x [21, 19, 20, 36]. We believe that it is interesting to show tight complexity bounds for this problem. One of the reasons why this problem is challenging is that the natural dynamic programming runs in time $2^{\mathcal{O}(|H| \cdot \mathrm{tw}(G))} \cdot (n + |H|)^{\mathcal{O}(1)}$. Thus in order to show that this bound is tight, it is not sufficient to design edge gadgets encoding inequality and substitute all edges of the instance of k-Coloring with these edge gadgets, as we did in this paper. Since the number of colors needs to be exponential in H, one should also design some *vertex gadgets*, which will encode the exponential number of possible states.

Finally, instead of changing the problem, we can consider changing the parameter. We believe that an exciting question is to find tight bounds for Hom(H) and LHom(H), parameterized by the *cutwidth* of the instance graph, denoted by $\mathrm{cw}(G)$. Quite recently Jansen and Nederlof showed that the chromatic number of a graph can be found in time $2^{\mathcal{O}(\mathrm{cw}(G))} \cdot n^{\mathcal{O}(1)}$ [30], i.e., the base of the exponential factor does not depend on the number of colors. Jansen [29] asked whether the same is possible for Hom(H) and LHom(H), if the target H is not complete? Note that while the chromatic number of a graph can be found in time $2^n \cdot n^{\mathcal{O}(1)}$ [3], for the Hom(H) and LHom(H) problems the $|H|^{\mathcal{O}(n)}$-time algorithm is essentially best possible, assuming the ETH [12]. We believe that a similar phenomenon might occur if the cutwidth is a parameter.

References

1 Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Methods*, 8(2):277–284, April 1987. doi:10.1137/0608024.

2 Stefan Arnborg and Andrzej Proskurowski. Linear time algorithms for NP-hard problems restricted to partial k-trees. *Discrete Applied Mathematics*, 23(1):11–24, 1989. doi:10.1016/0166-218X(89)90031-0.

3 Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. Set partitioning via inclusion-exclusion. *SIAM J. Comput.*, 39(2):546–563, 2009. doi:10.1137/070683933.

4 Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996. doi:10.1137/S0097539793251219.

5 Hans L. Bodlaender, Paul S. Bonsma, and Daniel Lokshtanov. The fine details of fast dynamic programming over tree decompositions. In *Parameterized and Exact Computation - 8th International Symposium, IPEC 2013, Sophia Antipolis, France, September 4-6, 2013, Revised Selected Papers*, pages 41–53, 2013. doi:10.1007/978-3-319-03898-8_5.

6 Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Inf. Comput.*, 243:86–111, 2015. doi:10.1016/j.ic.2014.12.008.

7 Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michal Pilipczuk. A $c^k n$ 5-approximation algorithm for treewidth. *SIAM J. Comput.*, 45(2):317–378, 2016. doi:10.1137/130947374.

8 Hans L. Bodlaender and Arie M. C. A. Koster. Combinatorial optimization on graphs of bounded treewidth. *Comput. J.*, 51(3):255–269, May 2008. doi:10.1093/comjnl/bxm037.

9 Andrei A. Bulatov. Constraint satisfaction problems: Complexity and algorithms. In Shmuel Tomi Klein, Carlos Martín-Vide, and Dana Shapira, editors, *Language and Automata Theory and Applications*, pages 1–25, Cham, 2018. Springer International Publishing.

10 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.

11 Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.

12 Marek Cygan, Fedor V. Fomin, Alexander Golovnev, Alexander S. Kulikov, Ivan Mihajlin, Jakub Pachocki, and Arkadiusz Socala. Tight lower bounds on graph embedding problems. *J. ACM*, 64(3):18:1–18:22, 2017. doi:10.1145/3051094.

13 Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*. Springer, 2015.

14 László Egri, Andrei A. Krokhin, Benoît Larose, and Pascal Tesson. The complexity of the list homomorphism problem for graphs. *Theory Comput. Syst.*, 51(2):143–178, 2012. doi:10.1007/s00224-011-9333-8.

15 László Egri, Dániel Marx, and Paweł Rzążewski. 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 fuer Informatik, 2018. doi:10.4230/LIPIcs.STACS.2018.27.

16 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.

17 Tomás Feder, Pavol Hell, and Jing Huang. List homomorphisms and circular arc graphs. *Combinatorica*, 19(4):487–505, 1999. doi:10.1007/s004939970003.

18 Tomás Feder, Pavol Hell, and Jing Huang. Bi-arc graphs and the complexity of list homomorphisms. *Journal of Graph Theory*, 42(1):61–80, 2003. doi:10.1002/jgt.10073.

19 Jirí Fiala and Jana Maxová. Cantor-Bernstein type theorem for locally constrained graph homomorphisms. *Eur. J. Comb.*, 27(7):1111–1116, 2006. doi:10.1016/j.ejc.2006.06.003.

20 Jirí Fiala and Daniël Paulusma. A complete complexity classification of the role assignment problem. *Theor. Comput. Sci.*, 349(1):67–81, 2005. doi:10.1016/j.tcs.2005.09.029.

21 Jirí Fiala, Daniël Paulusma, and Jan Arne Telle. Matrix and graph orders derived from locally constrained graph homomorphisms. In Joanna Jedrzejowicz and Andrzej Szepietowski, editors, *Mathematical Foundations of Computer Science 2005, 30th International Symposium, MFCS 2005, Gdansk, Poland, August 29 - September 2, 2005, Proceedings*, volume 3618 of *Lecture Notes in Computer Science*, pages 340–351. Springer, 2005. doi:10.1007/11549345_30.

22 Fedor V. Fomin, Pinar Heggernes, and Dieter Kratsch. Exact algorithms for graph homomorphisms. *Theory Comput. Syst.*, 41(2):381–393, 2007. doi:10.1007/s00224-007-2007-x.

23 Markus Frick and Martin Grohe. The complexity of first-order and monadic second-order logic revisited. *Ann. Pure Appl. Logic*, 130(1-3):3–31, 2004. doi:10.1016/j.apal.2004.01.007.

24 Richard Hammack, Wilfried Imrich, and Sandi Klavžar. *Handbook of product graphs*. Discrete Mathematics and its Applications (Boca Raton). CRC Press, Boca Raton, FL, second edition, 2011. With a foreword by Peter Winkler.

25 Pavol Hell and Jaroslav Nešetřil. *Graphs and homomorphisms*. Oxford University Press, 2004.

26 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.

27 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

28 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

29 Bart M. P. Jansen. personal communication.

30 Bart M. P. Jansen and Jesper Nederlof. Computing the chromatic number using graph decompositions via matrix rank. *Theor. Comput. Sci.*, 795:520–539, 2019. doi:10.1016/j.tcs.2019.08.006.

31 Tomasz Kociumaka and Marcin Pilipczuk. Deleting vertices to graphs of bounded genus. *Algorithmica*, 81(9):3655–3691, 2019. doi:10.1007/s00453-019-00592-7.

32 Stefan Kratsch, Daniel Lokshtanov, Dániel Marx, and Peter Rossmanith. Optimality and tight results in parameterized complexity (dagstuhl seminar 14451). *Dagstuhl Reports*, 4(11):1–21, 2014. doi:10.4230/DagRep.4.11.1.

33 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.

34 Karolina Okrasa, Marta Piecyk, and Paweł Rzążewski. Full complexity classification of the list homomorphism problem for bounded-treewidth graphs. *CoRR*, abs/2006.11155, 2020. arXiv:2006.11155.

35 Karolina Okrasa and Paweł Rzążewski. Fine-grained complexity of graph homomorphism problem for bounded-treewidth graphs. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms*, pages 1578–1590, 2020. doi:10.1137/1.9781611975994.97.

36 Karolina Okrasa and Paweł Rzążewski. Subexponential algorithms for variants of the homomorphism problem in string graphs. *J. Comput. Syst. Sci.*, 109:126–144, 2020. doi:10.1016/j.jcss.2019.12.004.

37 Michał Pilipczuk. Problems parameterized by treewidth tractable in single exponential time: A logical approach. In *MFCS 2011*, volume 6907, pages 520–531. Springer, 2011.

38 Paweł Rzążewski. Exact algorithm for graph homomorphism and locally injective graph homomorphism. *Inf. Process. Lett.*, 114(7):387–391, 2014. doi:10.1016/j.ipl.2014.02.012.

39 William T. Trotter and John I. Moore. Characterization problems for graphs, partially ordered sets, lattices, and families of sets. *Discrete Mathematics*, 16(4):361–381, 1976. doi:10.1016/S0012-365X(76)80011-8.

40 Johan M. M. van Rooij, Hans L. Bodlaender, and Peter Rossmanith. Dynamic programming on tree decompositions using generalised fast subset convolution. In Amos Fiat and Peter Sanders, editors, *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, volume 5757 of *Lecture Notes in Computer Science*, pages 566–577. Springer, 2009. doi:10.1007/978-3-642-04128-0_51.

41 Magnus Wahlström. New plain-exponential time classes for graph homomorphism. *Theory Comput. Syst.*, 49(2):273–282, 2011. doi:10.1007/s00224-010-9261-z.