Finding List Homomorphisms from Bounded-treewidth Graphs to Reflexive Graphs: a Complete Complexity Characterization

László Egri
Indiana State University, Department of Mathematics & Computer Science, Terre Haute, USA
Laszlo.Egri@indstate.edu

Dániel Marx
Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary
dmarx@sztaki.mta.hu

Pawel Rzążewski
Faculty of Mathematics and Information Science, Warsaw University of Technology, Warsaw, Poland
p.razzewski@mini.pw.edu.pl

Abstract
In the list homomorphism problem, the input consists of two graphs $G$ and $H$, together with a list $L(v) \subseteq V(H)$ for every vertex $v \in V(G)$. The task is to find a homomorphism $\phi : V(G) \to V(H)$ respecting the lists, that is, we have that $\phi(v) \in L(v)$ for every $v \in V(H)$ and if $u$ and $v$ are adjacent in $G$, then $\phi(u)$ and $\phi(v)$ are adjacent in $H$. If $H$ is a fixed graph, then the problem is denoted by $L\text{Hom}(H)$. We consider the reflexive version of the problem, where we assume that every vertex in $H$ has a self-loop. It is known that reflexive $L\text{Hom}(H)$ is polynomial-time solvable if $H$ is an interval graph and it is NP-complete otherwise [Feder and Hell, JCTB 1998].

We explore the complexity of the problem parameterized by the treewidth $\text{tw}(G)$ of the input graph $G$. If a tree decomposition of $G$ of width $\text{tw}(G)$ is given in the input, then the problem can be solved in time $|V(H)|^{\text{tw}(G)} \cdot n^{O(1)}$ by naive dynamic programming. Our main result completely reveals when and by exactly how much this naive algorithm can be improved. We introduce a simple combinatorial invariant $i^\star(H)$, which is based on the existence of certain decompositions and incomparable sets, and show that this number should appear as the base of the exponent in the best possible running time. Specifically, we prove for every non-interval reflexive graph $H$ that

- If a tree decomposition of width $\text{tw}(G)$ is given in the input, then the problem can be solved in time $i^\star(H)^{\text{tw}(G)} \cdot n^{O(1)}$.
- Assuming the Strong Exponential-Time Hypothesis (SETH), the problem cannot be solved in time $(i^\star(H) - \epsilon)^{\text{tw}(G)} \cdot n^{O(1)}$ for any $\epsilon > 0$.

Thus by matching upper and lower bounds, our result exactly characterizes for every fixed $H$ the complexity of reflexive $L\text{Hom}(H)$ parameterized by treewidth.

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

Keywords and phrases Graph Homomorphism, List Homomorphism, Reflexive Graph, Treewidth

Digital Object Identifier 10.4230/LIPIcs.STACS.2018.27

Funding This work was supported by ERC Starting Grant PARAMTIGHT (No. 280152), ERC Consolidator Grant SYSTEMATICGRAPH (No. 725978) of the European Research Council, and NSF EAGER Grant (No. 1751765).
1 Introduction

It is well known that most NP-hard algorithmic graph problems can be solved significantly more efficiently on graphs of bounded treewidth than on general graphs. A large number of NP-hard problems are known to be fixed-parameter tractable (FPT) parameterized by treewidth, that is, if the input instance contains a tree decomposition of width \( w \) of the graph, then the problem can be solved in time \( f(w) \cdot n^{O(1)} \) for some computable function \( f \) depending only on the width \( w \). In recent years, there have been significant research efforts to understand how complexity depends on treewidth and to determine the best possible function \( f(w) \) that can appear in the running time. On the algorithmic side, new algorithms with improved running times were obtained for a number of problems \([6, 1, 25, 16]\). On the complexity side, conditional lower bounds were given that, in many cases, match the running time of the best known algorithms, thereby giving a tight understanding of the complexity of the problem parameterized by treewidth \([21, 20, 6, 22, 5]\). These lower bounds are usually based on the Exponential-Time Hypothesis (ETH), which can be informally stated as \( n \)-variable 3-Sat cannot be solved in time \( 2^{o(n)} \), or on the Strong Exponential-Time Hypothesis (SETH), which can be informally stated as \( n \)-variable \( m \)-clause \( \text{CNF-Sat} \) cannot be solved in time \( (2 - \epsilon)^n \cdot n^{O(1)} \) for any \( \epsilon > 0 \).

As an exemplary result, let us consider the \( c \)-\textsc{Coloring} problem, where the task is to color the vertices of the graph with \( c \) colors such that adjacent vertices receive distinct colors. Using standard dynamic programming techniques, \( c \)-\textsc{Coloring} can be solved in time \( c^{\text{tw}(G)} \cdot n^{O(1)} \) if a tree decomposition of width \( \text{tw}(G) \) is given in the input. A result of Lokshtanov et al. \([20]\) showed that this running time is essentially optimal.

> **Theorem 1** (Lokshtanov, Marx, and Saurabh \([20]\)). Let \( c \geq 3 \) be a fixed integer. Assuming the SETH, the \( c \)-\textsc{Coloring} problem on a graph \( G \) with \( n \) vertices, given with its tree decomposition of width \( \text{tw}(G) \), cannot be solved in time \( \text{poly}(n) \cdot (c - \epsilon)^{\text{tw}(G)} \) for any \( \epsilon > 0 \).

**Homomorphisms.** Given graphs \( G \) and \( H \), a homomorphism from \( G \) to \( H \) is a mapping \( \phi : V(G) \rightarrow V(H) \) such that if \( uv \) is an edge of \( G \), then \( \phi(u) \phi(v) \) is an edge of \( H \). (In particular, if \( H \) has no loops, then this implies \( \phi(u) \neq \phi(v) \) whenever \( u \) and \( v \) are adjacent.) For every fixed graph \( G \), we can define the \( \text{Hom}(H) \) problem, where, given a graph \( G \), the task is to find a homomorphism from \( G \) to \( H \). Now \( c \)-\textsc{Coloring} is equivalent to \( \text{Hom}(K_c) \), where \( K_c \) is the clique on \( c \) vertices: it is easy to see that \( G \) is \( c \)-colorable if and only if it has a homomorphism to \( K_c \). Thus the Hom(H) family of problems form a far-reaching generalization of the vertex coloring problem. A classic result of Hell and Nešetřil \([17]\) characterized the complexity of \( \text{Hom}(H) \): it is polynomial-time solvable if \( H \) is bipartite and it is NP-complete for every nonbipartite \( H \) (see also \([4, 18]\)).

What can we say about the complexity of \( \text{Hom}(H) \) parameterized by treewidth? It seems to be a natural goal to try to obtain, for every \( H \), the best possible base \( c_H \) of the exponent that can appear in the running time \( c_H^{\text{tw}(G)} \cdot n^{O(1)} \). If \( H \) is the clique \( K_c \), then we know from Theorem 1 that \( c_H = c \), but what can we say about other graphs \( H \)? While this is a very natural question, it appears to be very difficult and deep as well: while the hardness of \( c \)-\textsc{Coloring} is well understood and can be easily exploited in hardness proofs such as Theorem 1, the hardness of \( \text{Hom}(H) \) for nonbipartite \( H \) comes from a somewhat mysterious combination of combinatorics and algebra \([17, 4, 18, 23]\).

While we are unable at the moment to characterize the exact complexity of \( \text{Hom}(H) \) parameterized by treewidth, we resolve a related question that is still of interest, but apparently more tractable. The problem we study differs from the original question in two
ways. First, we are considering the list version of the problem: in an input instance of the list homomorphism \( \text{LHom}(H) \) problem, each vertex \( v \) of \( G \) is equipped with a list \( L(v) \subseteq V(H) \) and the task is to find a homomorphism \( \phi \) from \( G \) to \( H \) that respects these lists, that is, \( \phi(v) \in L(v) \) for every \( v \in V(G) \). List versions of homomorphism and coloring problems are well studied [24, 7, 8, 13, 15, 12, 11, 14, 10, 9]. Typically, list versions are more robust than the ordinary versions and hardness proofs are simpler to prove for them. Feder et al. [10] characterized the polynomial-time solvable cases of \( \text{LHom}(H) \): now it is not sufficient that \( H \) is bipartite, it has to be the complement of a circular arc graph, otherwise the problem is NP-complete. Second, we consider the reflexive version of the homomorphism problem, which means that we assume that every vertex of \( H \) has a self-loop attached to it. Thus even if \( u \) and \( v \) are adjacent in \( G \), it is still possible that \( \phi(u) = \phi(v) \) in a homomorphism \( \phi \) from \( G \) to \( H \). In particular, now there is always a homomorphism \( \phi \) from every \( G \) to \( H \): let us chose an arbitrary fixed vertex \( u \in V(H) \) and let \( \phi(v) = u \) for every \( v \in V(G) \). However, it remains a nontrivial question whether there is a homomorphism from \( G \) to \( H \) that respects the lists \( L(v) \) of the vertices of \( G \). Feder and Hell [9] showed that reflexive \( \text{LHom}(H) \) is polynomial-time solvable if \( H \) is an interval graph, and NP-complete otherwise. In general, the reflexive problem appears to have simpler structure and cleaner properties than the irreflexive version, where bipartiteness and parity issues introduce technical complications. We believe that it is reasonable to start with the reflexive problem as a prototype result.

Results. Our main result is exactly characterizing, for every fixed \( H \), the complexity of reflexive list homomorphism parameterized by treewidth. Similarly to the \( c\)-\text{COLORING} problem, standard dynamic programming techniques give an algorithm with running time \( |V(H)|^{\text{tw}(G)} \cdot n^{O(1)} \) if a tree decomposition of width \( \text{tw}(G) \) is given (slightly more generally, we can also say that if every list \( L(v) \) has size at most \( c \), then the problem can be solved in time \( c^{\text{tw}(G)} \cdot n^{O(1)} \)). However, unlike in the case of the \( c\)-\text{COLORING} problem, this algorithm is not necessarily optimal: for some \( H \), we can actually do better. We identify two algorithmic ideas that can give improved algorithms:

- **Incomparable sets.** Suppose that the list \( L(v) \) for some \( v \in V(G) \) contains two vertices \( a, b \in V(H) \), such that every neighbor of \( a \) (including \( a \) itself) is also a neighbor of \( b \). It is easy to see that if there is a homomorphism \( \phi \) from \( G \) to \( H \) with \( \phi(v) = a \), then this can be modified to have \( \phi(v) = b \) and it remains a valid homomorphism. In other words, vertex \( a \) in the list \( L(v) \) is not necessary for the solution and can be removed from the list. Thus after a simple preprocessing step, we can assume that every \( L(v) \) is an incomparable set, that is, \( N[a] \subseteq N[b] \) does not hold for any two distinct \( a, b \in L(v) \). This means that if we denote by \( i(H) \) the maximum size of an incomparable set in \( H \), then it can be assumed that every list has size at most \( i(H) \) and hence the problem can be solved in time \( i(H)^{\text{tw}(G)} \cdot n^{O(1)} \). As \( i(H) \) can be much less than \( |V(H)| \), this running time can be significantly faster than \( |V(H)|^{\text{tw}(G)} \cdot n^{O(1)} \).

- **Decompositions.** We identify a certain kind of decomposition that can be used to simplify the problem. Formally, a decomposition is a partition \((S, N, R)\) of the vertices of \( H \) such that \( |S| \geq 2 \), \( |N \cup R| > 0 \), \( N \) separates \( S \) and \( R \), \( N \) induces a clique, and every vertex of \( S \) is adjacent to every vertex of \( N \). As we show later, such a decomposition allows us to reduce \( \text{LHom}(H) \) to instances of \( \text{LHom}(H_1) \) and \( \text{LHom}(H_2) \), where \( H_1 \) and \( H_2 \) are strict induced subgraphs of \( H \).

We show that, in a formal sense, these two algorithmic ideas are sufficient to solve the problem as fast as possible. First, if the graph \( H \) is undecomposable (that is, does not have a decomposition as above), then the best possible running time is indeed of the form \( i(H)^{\text{tw}(G)} \cdot n^{O(1)} \). More generally, we define \( i^*(H) \) to be the maximum of \( i(H^*) \), taken over
every undecomposable, connected, non-interval induced subgraph $H^*$ of $H$. Our main result shows that $i^*(H)^{\text{tw}(G)} \cdot n^{O(1)}$ is the exact complexity of the problem.

**Theorem 2.** Let $H$ be a connected reflexive non-interval graph with $k = i^*(H)$, and $G$ be a graph with $n$ vertices and treewidth $\text{tw}(G)$.

(a) The $\text{LHom}(H)$ problem with instance $(G, L)$ can be solved in time $\text{poly}(n + |H|) \cdot k^{\text{tw}(G)}$ for any lists $L$, provided that $G$ is given with its tree decomposition of width $\text{tw}(G)$.

(b) There is no algorithm that solves $\text{LHom}(H)$ for every $G$ and $L$ in time $f(H) \cdot \text{poly}(n + |H|) \cdot (k - \epsilon)^{\text{tw}(G)}$ for any computable function $f$ and any $\epsilon > 0$, unless the SETH fails.

Note that if $H$ is a reflexive interval graph, then $\text{LHom}(H)$ is polynomial-time solvable [9] and if $H$ is disconnected, then it is easy to reduce the problem to the components of $H$. Thus Theorem 2 gives a complete characterization of the complexity of the problem for every fixed $H$.

Let us discuss the significance of a complete classification result such as Theorem 2. As the $\text{LHom}(H)$ problem is an infinite family of problems, it is not clear at all what is the full range of algorithmic ideas that can help solve the problem faster than the naive $|V(H)|^{\text{tw}(G)} \cdot n^{O(1)}$ time algorithm. Even after realizing that this naive algorithm can be beaten in some cases (e.g., by discovering the importance of incomparable sets or some form of decompositions), we cannot be sure that some completely different algorithm cannot solve some cases even faster, or can be applied to an even wider class of target graphs $H$. But in order to prove a complete classification result of the form of Theorem 2, one has to discover each and every relevant algorithmic idea. Our main result not only provides a set of algorithmic tools, but proves in a formal sense (assuming the SETH) that no other algorithmic idea can improve on these results. Thus we completely map the complexity landscape of the $\text{LHom}(H)$ problem, determining the complexity of every case of $\text{LHom}(H)$ with surprising tightness and revealing every combinatorial insight that can be exploited algorithmically.

**Lower bound proofs.** The complexity result of Theorem 2b needs to exploit three properties of the induced subgraph $H^*$: it is not an interval graph, it is undecomposable, and has a large incomparable set. There are well-known characterization results that show that every non-interval graph contains certain obstructions (induced cycles or asteroidal triples) and the NP-hardness proofs of Feder and Hell [9] show how these obstructions can be used to reduce $\text{3-Coloring}$ to $\text{LHom}(H)$. However, here we need something much stronger: if there is an incomparable set $I$ of size $c = i^*(H)$, then we want to reduce $c$-$\text{Coloring}$ to $\text{LHom}(H)$ and use the lower bound in Theorem 1. The natural idea is to represent the $c$ colors of the $c$-$\text{Coloring}$ problem by the $c$ vertices appearing in the incomparable set $I$. Then the main challenge is to construct gadgets that express the $\neq$ relation, that is, ensure that two adjacent vertices are not assigned the same vertex of $I$, but every other combination is allowed. We show with a very delicate and technical proof that the incomparable set can be connected to the interval graph obstruction with a set of walks satisfying certain properties, and these walks, together with the obstruction, can be used to create the required gadgets. It turns out that, surprisingly, the only situation when we cannot find such walks is precisely when a decomposition exists. Thus if we assume that the graph is non-interval, has a large incomparable set, and has no decomposition, then we can construct the gadgets required for the reduction.

**Exploiting decompositions.** We finish the introduction with a brief explanation of how a decomposition $(S, N, R)$ can be exploited (a more detailed algorithm description appears in
Section 3.1). As discussed above, we can assume that every \( L(v) \) is an incomparable set in \( H \). In particular, this means that if some \( a \in S \) appears in \( L(v) \), then \( L(v) \) does not contain any vertex of \( N \) (as every vertex of \( N \) fully contains the neighborhood of every vertex in \( S \)).

Let \( X \subseteq V(G) \) be the set of vertices whose lists contain at least one vertex of \( S \). The set \( X \) induces some number of connected components in \( G \); let \( C \) be a connected component of \( G[X] \).

The first crucial observation is that in every solution \( \phi \), one of the following two cases has to happen on \( C \): either (1) \( \phi(c) \in S \) for every \( c \in C \), or (2) \( \phi(c) \notin S \) for every \( c \in C \). Otherwise, there would be at least two adjacent vertices \( c_1, c_2 \in C \) with \( \phi(c_1) \in S \) and \( \phi(c_2) \notin S \). However, as \( N \) separates \( S \) and \( R \) in \( H \), this is only possible if \( \phi(c_2) \in N \), contradicting our earlier assumption. Thus as a first step, we check if there is a homomorphism \( \phi_C \) from \( G[C] \) to \( H_1 := H[S] \) that respects the list. If there is no such homomorphism, then we can rule out the possibility that case (1) happens on \( C \) and remove the vertices of \( S \) from the lists of the vertices in \( C \). Suppose now that there is such a homomorphism \( \phi_C \).

The second crucial observation is that if case (1) happens on \( C \), then we might as well assume that the solution \( \phi \) restricted to \( C \) is exactly the same as \( \phi_C \): it is easy to see that \( \phi(v) \in N \) should hold for every \( v \in N(C) \), and every vertex of \( N \) is adjacent to every vertex of \( S \), hence no conflict can arise if we change \( \phi \) to be the same as \( \phi_C \) on \( C \). Let us select an arbitrary vertex \( a \in S \) and let us change the list of every \( v \in C \) to be \( L(v) := (L(v) \setminus S) \cup \{a\} \), that is, the single vertex \( a \) will represent the vertices \( L(v) \cap S \). We claim that this modification does not change the solvability of the instance: if the original instance has a solution where case (1) happens on \( C \), then we can modify it to have \( a \)'s on every vertex of \( C \); and if we obtain a solution of the new instance with \( a \)'s on \( C \), then we can obtain a solution of the original instance by using \( \phi_C \) on \( C \).

We repeat these steps for every connected component \( C \) of \( G[X] \). Then we obtain an instance where the selected vertex \( a \in S \) is the only vertex of \( S \) that appears anywhere on the lists. This means that effectively we have an instance where we need to find a homomorphism to \( H_2 := H \setminus (S \setminus \{a\}) \). As \( |S| \geq 2 \), \( H_2 \) has strictly fewer vertices than \( H \). Thus the existence of the decomposition \((S,N,R)\) allowed us to reduce the problem to instances of \( \text{LHom}(H_1) \) and \( \text{LHom}(H_2) \) where \( H_1 \) and \( H_2 \) have fewer vertices than \( H \).

## 2 Preliminaries

Throughout the paper we consider reflexive graphs only, i.e., we assume that for every vertex \( v, uv \) is an edge (a loop). Let \( H = (V,E) \) be a reflexive graph. By \( N[v] \) we denote the set \( \{u: uv \in E\} \). Note that \( v \in N[v] \). By \( N(v) \) we denote \( N[v] \setminus \{v\} \). For a set \( X \) of vertices, by \( N[X] \) we denote \( \bigcup_{v \in X} N[v] \), while \( N(X) \) denotes \( N[X] \setminus X \). For a set \( X \) and a vertex \( v \), by \( N_X[v] \) we denote \( N[v] \cap X \). In an analogous way we define \( N_X(v) \) and \( N_X[Y] \) for a set \( Y \). For two disjoint sets \( A, B \subseteq V \), such that no vertex from \( A \) is adjacent to a vertex from \( B \), an \( A\text{-}B \)-separator is a set \( S \), such that there is no path from any vertex \( a \in A \) to any vertex \( b \in B \) in the graph \( H - S \). An \( A\text{-}B \)-separator \( S \) is minimal if no \( S' \subseteq S \) is an \( A\text{-}B \)-separator. If \( A \) is a singleton, say \( A = \{a\} \), we write \( a\text{-}B \)-separator instead of \( \{a\}\text{-}B \)-separator (analogously if \( B \) is a singleton).

For two graphs \( G \) and \( H \), a mapping \( f: V(G) \to V(H) \) is a homomorphism if for every edge \( xy \) of \( G \) it holds that \( f(x)f(y) \) is an edge of \( H \). If \( f \) is a homomorphism from \( G \) to \( H \), we denote it shortly by \( f: G \to H \). We write \( G \to H \) to say that there exists some homomorphism from \( G \) to \( H \). For a fixed graph \( H \), by \( \text{Hom}(H) \) we consider an algorithmic problem of deciding if there is a homomorphism from a given graph \( G \) to \( H \).
In the list homomorphism problem we are given two graphs $G, H$ and a mapping $L: V(G) \rightarrow 2^{V(H)}$ (where the sets $L(v)$ for $v \in V(G)$ are called $H$-lists or just lists), and we ask for a homomorphism $f: G \rightarrow H$, such that $f(x) \in L(x)$ for every $x \in V(G)$. We denote this by $f: (G, L) \rightarrow H$. We often write $(G, L) \rightarrow H$ to denote that there is a list homomorphisms from $G$ to $H$ with lists $L$. Moreover, as we only deal with list homomorphisms, we write $f: G \rightarrow H$ to denote $f: (G, L) \rightarrow H$, if the lists are clear from the context. For a fixed graph $H$, by $L\text{Hom}(H)$ we denote the algorithmic problem, whose input is a graph $G$ with lists $L$, and we ask whether there exists a list homomorphism $(G, L) \rightarrow H$.

We observe that if $H$ has several connected components, then there is a polynomial-time reduction from $L\text{Hom}(H)$ to the problems $L\text{Hom}(H')$ for the connected components $H'$ of $H$. Thus we always assume that $H$ is connected.

### 2.1 Interval graphs and obstructions

Interval graphs are one of the most studied classes of geometric intersection graphs. A graph $H$ is an interval graph if it admits an interval representation, where each vertex is represented by some closed interval of the real line and two vertices are adjacent if and only if their corresponding intervals intersect. Note that interval graphs are usually defined to be irreflexive, but in our case we consider reflexive graphs.

Before we analyze structural properties of interval graphs, we need a few more definitions. An asteroidal triple is an independent set of three vertices $a, b, c$, such that for every $\{i, j, \ell\} = \{a, b, c\}$ there is an $i$-$j$-path $W_{i,j}$, whose every vertex is non-adjacent to $\ell$. Note that by our convention $W_{j,i}$ is $W_{i,j}$ reversed.

#### Theorem 3 (Lekkeikerker and Boland [19].) A graph is an interval graph if and only if does not contain an asteroidal triple or an induced cycle of length at least 4.

There is a deep connection between the list homomorphism problem and reflexive interval graphs, as shown in the following dichotomy theorem of Feder and Hell [9].

#### Theorem 4 (Feder and Hell [9].) Let $H$ be a reflexive graph. If $H$ is an interval graph, then the $L\text{Hom}(H)$ problem is polynomially solvable, otherwise it is NP-complete.

In this paper we will focus on graphs $H$, for which $L\text{Hom}(H)$ is NP-complete, so we will assume that $H$ is non-interval and thus contains at least one of structures mentioned in Theorem 3.

Observe that for an asteroidal triple $a, b, c$, we may w.l.o.g. assume that each path $W_{a,b}$ is induced. We define the asteroidal subgraph of an asteroidal triple $a, b, c$, as the subgraph of $H$ induced by $W_{a,b} \cup W_{b,c} \cup W_{a,c}$.\footnote{We will often identify graphs with their vertex sets.} For a vertex $a$ ($b, c$, resp.), we say that the path $W_{b,c}$ ($W_{a,c}, W_{a,b}$, resp.) is opposite.

Moreover, note that an induced cycle with at least 6 vertices contains an asteroidal subgraph. So an equivalent statement of Theorem 3 says that every non-interval graph $H$ contains an induced 4-cycle, an induced 5-cycle, or an asteroidal subgraph. An induced subgraph of $H$ isomorphic to one of these three structures is called an obstruction in $H$.

A vertex $o \in \mathcal{O}$ is a corner if:

- $\mathcal{O}$ is isomorphic to a 4-cycle or a 5-cycle, or
- $\mathcal{O}$ is an asteroidal subgraph for an asteroidal triple containing $o$.

Two vertices $o, o'$ of an obstruction $\mathcal{O}$ are opposite if:

1 We will often identify graphs with their vertex sets.
both are corners, or

\( Q \) is an asteroidal subgraph and \( o \) belongs to the path opposite to \( o' \).

### 2.2 Dominating vertices and incomparable sets

For two vertices \( u, v \) of \( H \), we say that \( v \) dominates \( u \) or, equivalently, \( u \) is dominated by \( v \), if \( N[u] \subseteq N(v) \). Observe that this implies that \( u \) and \( v \) are adjacent. We say that a set \( X \) dominates a set \( Y \) if every \( x \in X \) dominates every \( y \in Y \). If \( u \) is not dominated by \( v \), it means that there is a vertex \( u' \in N[u] \) (possibly \( u' = u \)), which is not a neighbor of \( v \). Two vertices \( u \) and \( v \) are incomparable if \( u \) does not dominate \( v \) and \( v \) does not dominate \( u \). A set \( S \) of vertices is incomparable if all its members are pairwise incomparable. By \( i(H) \) we denote the size of the largest incomparable set in \( H \).

### 2.3 Avoiding walks

A walk is a sequence of vertices \( P = p_1, p_2, \ldots, p_\ell \), such that \( p_ip_{i+1} \) is an edge for every \( i = 1, 2, \ldots, \ell - 1 \). For the walk \( P \), its length denotes the number \( \ell - 1 \). For two vertices \( a, b \), we say that \( P = p_1, p_2, \ldots, p_\ell \) is an \( a-b \)-walk if \( p_1 = a \) and \( p_\ell = b \). We denote this shortly by \( P : a \to b \). By \( \bar{P} \) we denote the reversed walk, i.e., \( \bar{P} = p_\ell, p_{\ell-1}, \ldots, p_2, p_1 \).

For two walks \( A = a_1, a_2, \ldots, a_\ell \) and \( B = b_1, b_2, \ldots, b_\ell \) such that \( a_\ell = 1 \), we let \( A \circ B \) denote the concatenation of \( A \) and \( B \), i.e., the walk \( a_1, a_2, \ldots, a_\ell, b_1, b_2, \ldots, b_\ell \). Note that \( |A \circ B| = |A| + |B| - 1 \).

For two walks \( P = p_1, p_2, \ldots, p_\ell \) and \( Q = q_1, q_2, \ldots, q_\ell \) of equal length, we say that \( P \) avoids \( Q \) if \( p_i \) is non-adjacent to \( q_{i+1} \) for every \( i = 1, 2, \ldots, \ell - 1 \). We conclude this section with two simple observations concerning walks and avoidance.

- **Observation 5.** For walks \( A : a \to b \), \( B : b \to c \) and \( A' : a' \to b' \), \( B' : b' \to c' \), if \( A \) avoids \( A' \) and \( B \) avoids \( B' \), then \( A \circ B \) avoids \( A' \circ B' \).
- **Observation 6.** Let \( P = p_1, p_2, \ldots, p_\ell \) and \( Q = q_1, q_2, \ldots, q_\ell \) be two walks, such that \( P \) avoids \( Q \). Then \( Q \) avoids \( P \).

### 3 Algorithm

In this section we prove the algorithmic part of our main result, i.e., Theorem 2a). Let us start with the following simple observation.

- **Observation 7.** Let \( u, v \) be vertices of \( H \), such that \( v \) dominates \( u \). Let \( f : G \to H \) be a homomorphism, such that \( f(x) = u \) for some vertex \( x \) of \( G \). Then \( f' \) defined by \( f'(x) := v \) and \( f'(y) := f(y) \) for every \( y \in V(G) \setminus \{x\} \) is also a homomorphism from \( G \) to \( H \).

Thus we can assume that in our instance \((G, L)\) of \( \text{LHom}(H) \) the set \( L(x) \) is incomparable for every vertex \( x \) of \( G \) (otherwise we can safely remove a dominated vertex).

### 3.1 Decomposition

For a graph \( H \), let \( T(H, n, t) \) denote an upper bound for the time complexity of an algorithm solving the \( \text{LHom}(H) \) problem on a graph with \( n \) vertices and treewidth \( t \). The following lemma is the main tool in the proof of Theorem 2a). The proof is omitted in this extended abstract.
Lemma 8 (Decomposition lemma). Let \( H = (V, E) \) be a reflexive graph, whose vertex set can be partitioned into three subsets \( S, N, R \), such that:
1. \( |S| \geq 2 \),
2. \( N \) is a clique with at least one vertex,
3. \( N \) separates \( S \) and \( R \),
4. all edges between \( S \) and \( N \) are present in \( H \).

Let \( H_1 \) be the subgraph of \( H \) induced by \( S \), and \( H_2 \) be the subgraph of \( H \) obtained by contracting \( S \) to a single vertex. Moreover, suppose that there are constants \( c, d \), such that \( T(H_1, n, t) = O(c^t \cdot n^d) \) and \( T(H_2, n, t) = O(c^t \cdot n^d) \). Then \( T(H, n, t) = O(c^t \cdot n^d) \).

A graph \( H \) which satisfies the assumptions of Lemma 8 is called decomposable and we say that \( (S, N, R) \) is a decomposition of \( H \), or that \( H \) decomposes into \( H_1 \) and \( H_2 \). We refer to \( S \) as dominated part and the set \( N \) as dominating clique separator. A graph which is not decomposable is called undecomposable.

Observe that with \( H \) we can associate a decomposition tree \( T \), 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, then the decomposition tree has just one node. If \( H \) decomposes into \( H_1 \) and \( H_2 \), then node(\( H \)) has two children, node(\( H_1 \)) and node(\( H_2 \)), respectively. We construct a decomposition tree recursively. Clearly, each leaf of the decomposition tree is an undecomposable induced subgraph of \( H \). Note that a decomposition tree may not be unique, as a graph may have more than one decomposition. However, the number of leaves is always \( O(|H|) \), so the total number of nodes is also \( O(|H|) \).

3.2 Solving \( L\text{Hom}(H) \) problem

Now we are ready to present an algorithm for determining if \((G, L) \rightarrow H\).

Proof of Theorem 2a). We assume that the graph \( G \) has \( n \) vertices and is given along with its tree decomposition of width \( tw(G) \). We also define

\[
i^*(H) := \max \{ i(H') : H' \text{ is undecomposable connected non-interval induced subgraph of } H \}.
\]

Observe that if \( H' \) is an induced subgraph of \( H \), then \( i^*(H') \leq i^*(H) \), and thus \( i(H) = i^*(H) \) for undecomposable \( H \).

It can be shown that in time polynomial in \( H \) we can check if \( H \) is undecomposable, or find a decomposition. If \( H \) is undecomposable, we run a standard dynamic programming on a tree decomposition of \( G \) (see [3, 2]). For each bag of the tree decomposition we store all partial list homomorphisms from the graph induced by this bag to \( H \). By Observation 7, the size of each list \( L(x) \) for \( x \in V(G) \) is at most \( i(H) \), thus the complexity of the dynamic programming algorithm is bounded by \( O(n^d \cdot i^*(H)^{tw(G)}) = O(n^d \cdot i^*(H)^{tw(G)}) \) for some constant \( d \).

So suppose \( H \) is decomposable. Let \( T \) be a decomposition tree of \( H \), note that it can be constructed in polynomial time, has \( O(|H|) \) nodes, and its every leaf corresponds to an induced subgraph of \( H \) with strictly fewer vertices. Indeed, if \( H \) decomposes into \( H_1 \) and \( H_2 \), then they are both induced subgraphs of \( H \) and \( |H_1|, |H_2| < |H| \). Therefore, for any leaf \( H' \) of \( T \), we can solve every instance of \( L\text{Hom}(H') \) with \( n \) vertices and treewidth at most \( tw(G) \) in time \( O(n^d \cdot i^*(H)^{tw(G)}) \) (note that this is also true if \( H' \) is an interval graph, as then we can use a polynomial algorithm). Now, applying Lemma 8 in a bottom-up fashion, we conclude that we can solve \( L\text{Hom}(H) \) in time \( O(n^d \cdot i^*(H)^{tw(G)}) \), which completes the proof of Theorem 2a).


4 Hardness

In this section we prove Theorem 2b), i.e., the lower bound for an algorithm deciding the existence of a list homomorphism $(G, L) \rightarrow H$. We will prove the following theorem.

- **Theorem 9.** Let $H$ be a connected, reflexive, undecomposable graph with $\iota(H) \geq 3$. Assuming the SETH, there is no algorithm that solves $L\text{HOM}(H)$ for every $G$ and $L$ in time $f(H) \cdot \text{poly}(|G| + |H|) \cdot (\iota(H) - \epsilon)^{\text{tw}(G)}$ for any $\epsilon > 0$ and any computable function $f$.

Let us first show that Theorem 9 is equivalent to Theorem 2b).

**Theorem 9 \rightarrow Theorem 2b.** Suppose Theorem 9 holds and Theorem 2b) fails. So there is a graph $H$ (may be decomposable) and an algorithm $A$ that solves $L\text{HOM}(H)$ in time $f(H) \cdot \text{poly}(|G| + |H|) \cdot (\iota(H) - \epsilon)^{\text{tw}(G)}$ for every input $G$, $L$. Let $H'$ be an undecomposable connected non-interval induced subgraph of $H$, such that $\iota(H') = \iota(H)$. As every instance of $L\text{HOM}(H')$ can be seen as an instance of $L\text{HOM}(H)$, the algorithm $A$ can be used to solve $L\text{HOM}(H')$ in time $f'(H') \cdot \text{poly}(|G| + |H'|) \cdot (\iota(H') - \epsilon)^{\text{tw}(G)}$, thus contradicting Theorem 9.

**Theorem 2b) \rightarrow Theorem 9.** Suppose Theorem 2b) holds and Theorem 9 fails. So there is an undecomposable graph $H$ and an algorithm $A$ that solves $L\text{HOM}(H)$ in time $f(H) \cdot \text{poly}(|G| + |H|) \cdot (\iota(H) - \epsilon)^{\text{tw}(G)}$ for every input $G$, $L$. But since $H$ is undecomposable, we have $\iota(H) = \iota(H)$, so algorithm $A$ contradicts Theorem 2b).

4.1 Using an obstruction to express basic relations

Let $\mathcal{O}$ be an obstruction in $H$ with non-adjacent corners $\alpha, \beta$ and let $k \geq 2$ be an integer.

First, we show how express $k$-wise relations $OR_k = \{\alpha, \beta\}^k \setminus \alpha \beta$ and NAND$_k = \{\alpha, \beta\}^k \setminus \beta \alpha$.

More formally, we define a graph $F(OR_k)$ ($F($NAND$_k)$, resp.), called an $OR_k$-gadget (NAND$_k$-gadget, resp.) with $H$-lists $L$ and $k$ specified vertices $x_1, x_2, \ldots, x_k$, such that:

- for every $i \in [k]$ it holds that $L(x_i) = \{\alpha, \beta\}$,
- the relation $\bigcup f: F(OR_k) \rightarrow H\{f(x_1)f(x_2) \cdots f(x_k)\}$ is exactly $OR_k$ (respectively, $\bigcup f: F($NAND$_k) \rightarrow H\{f(x_1)f(x_2) \cdots f(x_k)\}$ is NAND$_k$).

The construction of these gadgets is simple and it is omitted in this extended abstract. Another useful property of obstructions is shown in the following lemma.

- **Lemma 10 (Moving inside the obstruction).** Let $\mathcal{O}$ be an obstruction with distinct corners $a, c$. Moreover, let $b, d$ be distinct vertices of $\mathcal{O}$, such that $b$ is a corner and $d$ is either a corner, or a vertex non-adjacent to $b$. Then there are walks $A_{a,b}, A'_{a,b}: a \rightarrow b$ and $B_{c,d}, B'_{c,d}: c \rightarrow d$, such that $A_{a,b}$ avoids $B_{c,d}$, and $B'_{c,d}$ avoids $A'_{a,b}$. Moreover, all four walks use only vertices of $\mathcal{O}$ and can be constructed in polynomial time.

**Proof.** If $\mathcal{O}$ is an induced 4-cycle or an induced 5-cycle, the walks are easy to construct. So consider the case that $\mathcal{O}$ is an asteroidal subgraph for an asteroidal triple $a_1, a_2, a_3$.

**Case 1.** First, let us deal with the case when both $b$, $d$ are corners. Then we have $\{a, b, c, d\} \subseteq \{a_1, a_2, a_3\}$. If $a = b$ and $c = d$ then the problem is trivial. If $a = b$ and $c \neq d$, then we set $A_{a,b} = A'_{a,b} = a, a, \ldots, a$ and $B_{c,d} = B'_{c,d} = W_{c,d}$ (the walk opposite to a). The case when $a \neq b$ and $c = d$ is similar. So we assume that $a \neq b$ and $c \neq d$. If $a = d$ and $c \neq b$ (the case when $a \neq d$ and $c = b$ is similar), we set $A_{a,b} = A'_{a,b} = W_{a,b} \circ b, b, \ldots, b$ and $B_{c,d} = B'_{c,d} = c, c, \ldots, c \circ W_{c,d}$. If $a = d$ and $c \neq b$, we set $A_{a,b} = A'_{a,b} = a, a, \ldots, a \circ W_{a,c} \circ c, c, \ldots, c$ and $B_{c,d} = B'_{c,d} = W_{c,b} \circ b, b, \ldots, b \circ W_{b,a}$.
Case 2. Next, consider the case when $b$ is a corner and $d$ is not. We know that $d \in W'_b,b''$, where $b',b''$ are corners and $b' \neq b$. Let $W'$ be the subpath of $W'_b,b''$, starting in $b'$ and ending in $d$. Recall that $W'_b,b''$ is induced, so even if $b = b''$, there is no edge from $W'$ to $b$. We set

$$A_{a,b} = C_{a,b} \circ b,b,\ldots,b, \quad A'_{a,b} = C'_{a,b} \circ b,b,\ldots,b,$$

$$B_{c,d} = D_{c,b'} \circ W', \quad B'_{c,d} = D'_{c,b'} \circ W',$$

where $C_{a,b}, C'_{a,b} : a \to b$ and $D_{c,b'}, D'_{c,b'} : c \to b'$ are appropriate walks given by Case 1. ▶

4.2 Constructing distinguishing walks

As we have seen, we can easily use the structure of an obstruction to enforce non-trivial relations that could be used to show hardness. For the rest of the proof we will show that we can attach vertices of an incomparable set to the vertices of an obstruction using walks with certain avoidance properties, which will later be exploited to prove hardness.

For a walk $P = v_1, v_2, \ldots, v_n$, by $\tilde{P}$ we denote the walk $P$ with its first vertex removed, i.e., $\tilde{P} = v_2, \ldots, v_n$. The following structural lemmas will be later used to obtain the main gadget used in our hardness proof.

▶ Lemma 11. Let $H$ be a connected undecomposable non-interval reflexive graph, and $O$ be an obstruction in $H$ with non-adjacent corners $\alpha, \beta$. Let $S$ be a set of incomparable vertices in $H$ such that $|S| \geq 2$. Let $a$ and $b$ be arbitrary distinct vertices in $S$. Then there is a partition $(X,Y)$ of $S$ such that vertices $a$ and $b$ are in $X$, and there are walks $D_v$ for each $v \in S$ of length at least 1, satisfying the following properties:

1. For each $v \in S$, the first vertex of $D_v$ is $v$, and its last vertex is either $\alpha$ ($D_v$ is said to be an $\alpha$-walk) or $\beta$ ($D_v$ is said to be a $\beta$-walk).
2. $D_a$ is an $\alpha$-walk and $D_b$ is a $\beta$-walk.
3. Let $u, v \in S$ such that $D_u$ is an $\alpha$-walk and $D_v$ is a $\beta$-walk. Then
   a. if $u, v \in X$, or if $u \in Y$ and $v \in X \cup Y$, then $D_u$ avoids $D_v$,
   b. if $u \in X$, $v \in Y$, then $D_u$ avoids $D_v$.
4. For any $v \in Y$ and $u \in X$, there is no edge joining $v$ and the second vertex of $D_u$.
5. For every $v \in Y$, the second vertex of $D_v$ is in $Y$.

Recall that it is possible that we have two walks $D_x, D_y$ constructed in Lemma 11, such that $D_x$ is an $\alpha$-walk, $D_y$ is a $\beta$-walk, but $D_x$ does not avoid $D_y$ (this may happen for $x \in X$ and $y \in Y$). This is an undesired situation for us, but luckily such walks have a well-defined structure. In the next lemma we will construct a small gadget to patch this situation, and then we will combine them to construct the main tool in our hardness proof, i.e., a distinguisher gadget.

▶ Lemma 12. Let $H, S, X, Y$, and $D_v$, where $v \in S$, be as in Lemma 11. Let $N_X = \{d^y_x : x \in X\}$, i.e., the set of vertices that appear as a second vertex of a walk $D_x$ where $x \in X$ and $N_Y = \{d^x_y : y \in Y\}$. Then there is a graph $F$ with $H$-lists and two specified vertices $p_1, p_2 \in V(F)$ such that

1. $L(p_1) = S$ and $L(p_2) = N_X \cup N_Y$,
2. for any list homomorphism $\varphi : F \to H$, if $\varphi(p_1) \in X$, then $\varphi(p_2) \notin Y$,
3. for every $v \in S$, there is $\psi : F \to H$, such that $\psi(p_1) = v$ and $\psi(p_2) = d^y_x$. 


### 4.3 Constructing a distinguisher gadget

The main tool in our hardness proof is a gadget called a **distinguisher**. Let $H$ be an undecomposable reflexive non-interval graph with obstruction $\emptyset$ with non-adjacent corners $\alpha, \beta$. For an incomparable set $S$ of $H$ and two vertices $a, b \in S$, a distinguisher is a graph $D_{a/b}$ with two specified vertices $x, y$ and $H$-lists $L$, such that:

1. $L(x) = S$ and $L(y) = \{\alpha, \beta\}$,
2. there is a list homomorphism $\phi_a: D_{a/b} \to H$, such that $\phi_a(x) = a$ and $\phi_a(y) = \alpha$,
3. there is a list homomorphism $\phi_b: D_{a/b} \to H$, such that $\phi_b(x) = b$ and $\phi_b(y) = \beta$,
4. for any $c \in S \setminus \{a, b\}$ there is $\phi_c: D_{a/b} \to H$, such that $\phi_c(x) = c$ and $\phi_c(y) \in \{\alpha, \beta\}$,
5. there is no list homomorphism $\phi: D_{a/b} \to H$, such that $\phi(x) = a$ and $\phi(y) = \beta$.

**Lemma 13 (Construction of distinguisher).** Let $H = (V, E)$ be an undecomposable reflexive non-interval graph with obstruction $\emptyset$ with two non-adjacent corners $\alpha, \beta$. Let $S$ be a maximum incomparable set in $H$. Then for every ordered pair $(a, b)$ of distinct elements of $S$ there exists a distinguisher $D_{a/b}$.

**Proof.** Call Lemma 11 for $H, S, a, b$ to obtain a partition $(X, Y)$ of $S$ and walks $D_v$ for every $v \in S$. Let $s$ be the length of each of these walks. By $d^v_j$ we denote the $j$-th vertex of $D_v$.

Let $P$ be a path with $s$ vertices $p_1, p_2, \ldots, p_s$. We set $L(p_i) = \bigcup_{v \in S}\{d^v_j\}$. Observe that by Lemma 11 we have $s \geq 2$.

Next, call Lemma 12 to obtain a graph $F_P$ and unify its $p_1$-vertex with $p_1$ of $P$ and its $p_2$-vertex with $p_2$ of $P$. Observe that this unification preserves lists. Finally, we set $x = p_1$ and $y = p_s$. Let us verify that the graph constructed in such a way is indeed a distinguisher.

The first property holds by the definition of the walks $D_v$ for $v \in S$. To show properties 2, 3, and 4, consider $v \in S$ and set $\phi_v(p_i) = d^v_j$ for all $i \in [s]$. This mapping can be extended to the vertices of $F_P$ by property 3 of Lemma 12.

Finally, let us show that property 5 holds as well. Assume for the sake of contradiction that a list homomorphism $\phi: D_{a/b} \to H$, such that $\phi(x) = a$ and $\phi(y) = \beta$, exists. Observe that $\phi(p_1), \phi(p_2), \ldots, \phi(p_s)$ is an $\alpha$-$\beta$ walk of length $s$ in $H$, such that for every $i \in [s]$ we have $\phi(p_i) \in \bigcup_{v \in S}\{d^v_j\}$. For all $i \in [s]$, let $D^i$ denote a walk from $\{D_v: v \in S\}$, whose $i$-th vertex is $\phi(p_i)$ (if there is more than one such a walk, choose an arbitrary one). Observe that $D^1 = D_v$ is an $\alpha$-walk. Let $i$ be a minimum integer, such that $D^i$ is a $\beta$-walk. This value is well-defined, as $D^s$ is a $\beta$-walk. Thus there is an edge between the $(i - 1)$th vertex of the $\alpha$-walk $D^{i-1}$ and the $i$-th vertex of the $\beta$-walk $D^i$, so $D^{i-1}$ does no avoid $D_v$. Let $u, v$ be vertices such that $D^{i-1} = D_u$ and $D^i = D_v$. If $u, v \in X$, or $u \in Y$ and $v \in X \cup Y$, then we have a contradiction with property 3a in Lemma 11.

Thus assume that $u \in X$ and $v \in Y$. If $i \geq 3$, then again we have a contradiction with property 3b in Lemma 11. Thus the only case left is $i = 2$. By property 2, in Lemma 12 we observe that $d^v_2 \notin Y$, and thus, by property 5 in Lemma 11, we conclude that $v \notin Y$, a contradiction. This completes the proof.

### 4.4 Hardness proof

We are ready to prove Theorem 9. Recall that Theorem 9 implies Theorem 2b).

**Proof of Theorem 9.** Let $\emptyset$ be an obstruction in $H$ and let $\alpha, \beta$ be non-adjacent corners of $\emptyset$. Let $S = \{v_1, v_2, \ldots, v_k\}$ be a maximum incomparable set in $H$. Note that we can assume that $k \geq 3$, since the corners of $\emptyset$ are pairwise incomparable.

Suppose we are given a graph $G$ along with its tree decomposition of width $tw(G)$. The main idea of our hardness proof is to construct a graph $G^*$ with $H$-lists $L$ such that:

---

**STACS 2018**
Finding List Homomorphisms from Bounded-treewidth Graphs to Reflexive Graphs

- \((G^*, L) \rightarrow H\) if and only if \(G\) is \(k\)-colorable (in our construction the colors used on \(G\) will correspond to vertices from \(S\))
- the number of vertices of \(G^*\) is \(g(H) \cdot (|V(G)| + |E(G)|)\) for some function \(g\) of \(H\)
- the treewidth of \(G^*\) is at most \(g(H) + tw(G)\)
- \(G^*\) can be constructed in time \(poly(|V(G)|) \cdot g'(H)\) for some function \(g'\).

Invoking Theorem 1, this will prove Theorem 9. The construction is performed in four steps.

**Step 1. Constructing an indicator gadget.** Fix \(i \in [k]\). For \(j \in [k] \setminus \{i\}\), we use Lemma 13 to construct a distinguisher gadget \(D_{v_i/v_j}\) with two specified vertices \(x_{i,j}\) and \(y_{i,j}\). The number of constructed gadgets is thus \(k-1\). We identify vertices \(x_{i,j}\) for all \(j \in [k]\), let us call this identified vertex \(x_i\). Moreover, introduce a new vertex \(c_i\). Now, using the construction from Section 4.1 we introduce an \(OR_k\) gadget and identify its specified vertices with distinct vertices from \(X_i := \{c_i\} \cup \bigcup_{j \in [k] \setminus \{i\}} \{v_{i,j}\}\) (there are \(k\) vertices in this set). Let us call this graph \(I_i\) (\(I\)' stands for indicator).

The construction forces that in every list homomorphism \(f: I_i \rightarrow H\), at least one vertex from \(X_i\) is mapped to \(\beta\). Observe that:
- for every \(f: I_i \rightarrow H\), if \(f(x_i) = v_i\), then \(f(c_i) = \beta\).
- for every \(j \neq i\), there exist \(f', f'': I_i \rightarrow H\), such that \(f'(x_i) = f''(x_i) = v_j\) and \(f'(c_i) = \alpha\) and \(f''(c_i) = \beta\).

**Step 2. Constructing a half-edge gadget.** Let us construct \(k\) indicator gadgets \(I_1, I_2, \ldots, I_k\). We identify the vertices \(x_1, x_2, \ldots, x_k\), and call this vertex \(x\). Call the resulting gadget a half-edge. By the construction of indicators, we observe that for a half-edge \(F\), the following hold:
- for every \(f: F \rightarrow H\), if \(f(x) = v_i\), then \(f(c_i) = \beta\).
- for every \(i \in [k]\) and every tuple \(X \in \{\alpha, \beta\}^k\), such that \(X_i = \beta\), there exists \(f: F \rightarrow H\) such that \(f(x) = v_i\) and \(f(c_j) = X_j\).

**Step 3. Constructing an edge gadget.** An edge gadget consists of two half-edge gadgets \(F, F'\) (we will use primes to denote the vertices in \(F'\)). Moreover, for every \(i \in [k]\), we introduce a \(NAND_2\) gadget on vertices \(c_i\) and \(c_i'\), which enforces that at least one of them is mapped to \(\alpha\). We call the resulting graph an edge gadget. For an edge gadget \(FF\) we observe the following:
- for any \(f: FF \rightarrow H\), if \(f(x) = v_i\), then \(f(x') \neq v_i\). Assume for contradiction that \(f(x) = f(x') = v_i\). Then by the construction of a half-edge, we observe that \(f(c_i) = f(c_i') = \beta\). However, this is impossible by the definition of \(NAND_2\) gadget.
- for any distinct \(i, j \in [k]\) there is \(g: FF \rightarrow H\) such that \(g(x) = v_i\), and \(g(x') = v_j\). By the construction of a half-edge gadget, there is \(f: F \rightarrow H\), such that \(f(x) = v_i\), \(f(c_i) = \beta\), and \(f(c_i') = \alpha\) for every \(i' \neq i\) (in particular, for \(i' = j\)). Analogously, there is \(f': F' \rightarrow H\), such that \(f'(x') = v_j\), \(f'(c_j) = \beta\), and \(f'(c_j') = \alpha\) for every \(j' \neq j\). We obtain \(g\) by combining \(f\) and \(f'\), and extending this partial homomorphism to vertices of \(NAND_2\)-gadgets. By the definition of these gadgets, it is possible, as for every \(i'\) we have \(f(c_{i'}) = \alpha\) or \(f'(c_{i'}) = \alpha\).

Observe that the construction so far was performed for \(H\) only. Let \(g(H)\) be the number of vertices in an edge gadget.
Step 4. Constructing $G^*$ and $H$-lists $L$. We start constructing $G^*$ by including the vertex set $V(G)$ of $G$ to $G^*$ (initially they are isolated vertices). For every edge $uv$ of $G$, we introduce an edge gadget, where $x$ is unified with $u$ and $x'$ is unified with $v$. By the construction of edge gadget, we observe that for every $v \in V(G)$ we have $L(v) = S$. Moreover, $(G^*, L)$ is a No-instance of $LHom(H)$ if and only if $G$ is $k$-colorable (we interpret mapping $u$ to $v \in S$ as coloring $u$ with color $i$). Recall that the size of each edge gadget is $g(H)$, thus the number of vertices of $G^*$ is at most $(|V(G)| + |E(G)|) \cdot g(H)$.

To see that the treewidth of $G^*$ is at most $\text{tw}(G) + g(H)$, consider a tree decomposition $\mathcal{T}$ of $G$ with width $\text{tw}(G)$. For every edge $uv$ of $G$, we choose one bag $X_{uv}$ of $\mathcal{T}$, such that $u, v \in X_{uv}$. Define a set $X'_{uv}$ as the union of $X_{uv}$ and the set of vertices of the edge gadget corresponding to the edge $uv$. We extend $\mathcal{T}$ to a tree decomposition $\mathcal{T}^*$ of $G^*$, by introducing a bag $X'_{uv}$ for every edge $uv$ of $G$ and making it adjacent (in $\mathcal{T}^*$) to $X_{uv}$ only. It is straightforward to verify that $\mathcal{T}^*$ is a tree decomposition of $G^*$ with width at most $\text{tw}(G) + g(H)$. Moreover, it is clear that $G^*$ and $L$ can be constructed in time $g'(H) \cdot \text{poly}(|V(G)|)$ for some function $g'$. Thus, by Theorem 1, our claim holds. △

References


Finding List Homomorphisms from Bounded-treewidth Graphs to Reflexive Graphs


