



List Coloring Ordered Graphs with Forbidden Induced Subgraphs

Marta Piecyk  

CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
Warsaw University of Technology, Poland

Paweł Rzażewski  

Warsaw University of Technology, Poland
University of Warsaw, Poland

Abstract

In the LIST k -COLORING problem we are given a graph whose every vertex is equipped with a list, which is a subset of $\{1, \dots, k\}$. We need to decide if G admits a proper coloring, where every vertex receives a color from its list.

The complexity of the problem in classes defined by forbidding induced subgraphs is a widely studied topic in algorithmic graph theory. Recently, Hajebi, Li, and Spirkl [SIAM J. Discr. Math. 38 (2024)] initiated the study of LIST 3-COLORING in *ordered graphs*, i.e., graphs with fixed linear ordering of vertices. Forbidding ordered induced subgraphs allows us to investigate the boundary of tractability more closely.

We continue this direction of research, focusing mostly on the case of LIST 4-COLORING. We present several algorithmic and hardness results, which altogether provide an almost complete dichotomy for classes defined by forbidding one fixed ordered graph: our investigations leave one minimal open case.

2012 ACM Subject Classification Mathematics of computing → Graph coloring; Mathematics of computing → Graph algorithms

Keywords and phrases coloring, ordered graphs, forbidden induced subgraphs

Digital Object Identifier 10.4230/LIPIcs.STACS.2026.74

Related Version *Full Version*: <https://arxiv.org/abs/2509.22160>

Funding

Marta Piecyk: Supported by Polish National Science Centre, grant no. 2022/45/N/ST6/00237.

Paweł Rzażewski: Supported by the National Science Centre grant 2024/54/E/ST6/00094.

1 Introduction

Coloring is one of the best studied problems in graph theory, both from structural and from algorithmic point of view. In this paper we are interested in the *list* variant of the problem. For a fixed integer k , an instance of the LIST k -COLORING problem is a pair (G, L) , where G is a graph and L is a function mapping each vertex of G to a subset of $\{1, \dots, k\}$ called *list*. We ask if G admits a proper coloring where every vertex v receives a color from its list $L(v)$. Clearly, this problem is NP-hard for all $k \geq 3$ as it generalizes k -COLORING. However, the presence of lists makes it hard even for very restricted instances, like bipartite graphs.

A popular approach for such problem is to explore the complexity on restricted instances, in order to understand the boundary of tractability. One of typical sources of such restricted instances comes from considering graphs excluding certain substructures, most notably, as induced subgraphs. For a graph H , we say that a graph G is H -free if it does not contain H as an induced subgraph.



© Marta Piecyk and Paweł Rzażewski;

licensed under Creative Commons License CC-BY 4.0

43rd International Symposium on Theoretical Aspects of Computer Science (STACS 2026).

Editors: Meena Mahajan, Florin Manea, Annabelle McIver, and Nguyễn Kim Thăng

Article No. 74; pp. 74:1–74:17



Leibniz International Proceedings in Informatics

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



The complexity of LIST k -COLORING in H -free graphs is quite well-understood. Let us introduce some notation. For an integer t , by P_t we denote the t -vertex path. We use “+” to denote disjoint union of graphs, so $H_1 + H_2$ denotes the graph with two components, one isomorphic to H_1 and the other one to H_2 . We also write rH to denote the disjoint union of r copies of H .

Classic hardness reductions imply that if H is not a linear forest (i.e., a forest of paths), then already LIST 3-COLORING is NP-hard for H -free graphs [9, 15, 17]. On the other hand, for every k , the LIST k -COLORING problem in H -free graphs is polynomial-time solvable if (a) H is an induced subgraph of rP_3 , for some r [10, 3, 12], or (b) H is an induced subgraph of $P_5 + rP_1$, for some r [14, 5].

For $k \geq 5$, all remaining cases are NP-hard [11, 16, 5]. The LIST 4-COLORING problem is NP-hard for P_6 -free graphs [11, 16], which leaves a number of open cases for disconnected H .

The case of LIST 3-COLORING is much more elusive. It is known to be in P for P_7 -free [2] and in $(P_6 + rP_3)$ -free graphs [4]. The general belief in the community is that LIST 3-COLORING in H -free graphs should be solvable in polynomial time for any linear forest H . This belief is supported by the existence of a *quasipolynomial-time* algorithm for all that cases [18]. This is a strong indication that the problem is not NP-hard. However, we seem to be very far from improving the quasipolynomial algorithm to a polynomial one.

Motivated by the notorious open case of $k = 3$, Hajebi, Li, and Spirkl [13] considered a slightly different setting. An *ordered graph* is a graph with a fixed linear order of vertices. For ordered graphs G and H , we say that H is an induced subgraph of G if it can be obtained from G by deleting vertices. Thus, the relative ordering of vertices of H must coincide with the relative ordering of their images in G . Now, the notion of H -free (ordered) graphs is a natural one: we forbid H as an induced (ordered) subgraph.

We remark that this setting allows us to understand the distinction between easy and hard cases even better. Indeed, excluding H as an unordered induced subgraph is equivalent to excluding all possible orderings of H . But what if we exclude just one, or few possible orderings?

As a motivating example, consider the case that $H = P_3$. The class of unordered P_3 -free graphs is very simple: every connected component of such a graph is a clique. Consequently, LIST k -COLORING is polynomial-time-solvable for every k .

Now let us look at the ordered setting. Up to isomorphism, there are three possible orderings of P_3 , i.e., \curvearrowright , \curvearrowleft , and \curvearrow . It turns out that forbidding at least one of \curvearrowright , \curvearrowleft leads to a problem that is in P for every fixed k . (As observed by Hajebi et al. [13], the instances of such a problem are chordal, see Proposition 1). However, LIST k -COLORING is NP-hard in \curvearrow -free ordered graphs for all $k \geq 3$ [13].

The results obtained by Hajebi et. al [13] are listed in Table 1. Let us make a few comments about implications between results. First, if H is an unordered graph, then hardness for H -free graphs implies hardness when we exclude any ordering of H . Second, since we consider the list problem, an algorithm for LIST k -COLORING implies an algorithm for LIST $(k - 1)$ -COLORING and the hardness for LIST $(k - 1)$ -COLORING implies the hardness for LIST k -COLORING. Third, if H' is an induced subgraph of H , then the algorithm for H -free graphs implies the algorithm for H' -free graphs, and hardness for H' -free graphs implies hardness for H -free graphs. Finally, if H' is an ordered graph obtained from H by reversing the ordering of vertices, the complexity in H -free and H' -free graphs is the same.

In this paper we consider the complexity of LIST k -COLORING for $k \geq 4$ in ordered H -free graphs. Let us briefly discuss our results, see also Table 1.

■ **Table 1** Complexity of LIST k -COLORING for H -free ordered graphs: state of the art. Green/red cells indicate that the problem is in P/NP-hard. Empty dots indicate vertices that might, but do not have to exist (the number of such vertices is arbitrary).

Forbidden H	$k = 3$	$k = 4$	$k \geq 5$
	[13]	Theorem 3	
	[13]	[13] + folklore	
	[13]	Theorem 5	?
	[13]	Theorem 11	
	?	Theorem 12	
	?		
	?	?	?
other	[13]		

We show that if H has one edge, then LIST k -COLORING is polynomial-time-solvable in H -free graphs, for any fixed k ; see Theorem 3. This extends earlier result of Hajebi et al. [13] for $k = 3$. Even though our algorithm works for every k , it is significantly simpler and has better complexity bound. Our approach is based on branching which allows us to obtain a family of “cleaned” instances that can be efficiently solved by dynamic programming.

We also show that LIST 4-COLORING can be solved in polynomial time for graphs that exclude a graph consisting of a copy of \curvearrowright preceded by an arbitrary number of isolated vertices; see Theorem 5. This algorithm is much more involved than the one from Theorem 3 and is the main algorithmic contribution of the paper. It consists of a few phases of branching that again lead us to a “cleaned” instance. In such an instance we remove certain edges. We remark that in general such an operation is not safe in H -free graphs as it might take us outside the class. However, we argue that in our case we can indeed do it. Finally, we reduce the problem to solving LIST 4-COLORING for a chordal graph, which can be done in polynomial time.

In stark contrast, in Theorem 11 we show that if H is a graph consisting of a copy of \curvearrowright , followed by a single isolated vertex, then LIST 4-COLORING is NP-hard in H -free graphs. We remark that this is also where the complexity of LIST 4-COLORING differs from the complexity of LIST 3-COLORING. Indeed, the latter problem is in P in classes defined by forbidding any graph obtained by adding isolated vertices before and after a copy of \curvearrowright .

Finally, in Theorem 12 we show that LIST 4-COLORING is NP-hard for \curvearrowright -free graphs; this is one of the open cases for LIST 3-COLORING.

We also look at the closely related LIST COLORING problem, where the number of colors is not bounded; see Appendix. Unsurprisingly, this problem turns out to be NP-hard in H -free graphs for all graphs H with at least three vertices. If H has two vertices, the problem is actually trivial.

The paper is concluded with several open questions and possible directions for further research.

Proofs of the statements marked with ♠ can be found in the full version of the paper.

2 Preliminaries







For a positive integer n , by $[n]$ we denote the set $\{1, \dots, n\}$. For two positive integers i, j such that $i < j$ by $[i, j]$ we denote $\{i, i + 1, \dots, j\}$. For a set X , by 2^X we denote the set of all subsets of X , and by $\binom{X}{k}$, where $k \in \mathbb{N}$, we denote the set of all k -element subsets of X .

For $k, \ell \in \mathbb{N}$, by $\text{Ram}(k, \ell)$ we denote the *Ramsey number of k and ℓ* , i.e., the minimum number n such that every graph on n vertices contains either a clique on k vertices or an independent set on ℓ vertices. It is known that for every $k, \ell \in \mathbb{N}$, the number $\text{Ram}(k, \ell)$ exists [19].

Ordered graphs. An ordered graph G is a graph given with a linear ordering of its vertices. For two vertices u, v , we write $u \prec v$ if u appears the ordering earlier than v .

We say that u is a *forward neighbor* (resp., *backward neighbor*) of v if $uv \in E(G)$ and $v \prec u$ (resp., $u \prec v$). The set of forward (resp., backward) neighbors of v is denoted by $N^+(v)$ (resp., $N^-(v)$).

Special graphs. Let us define some special ordered graphs that will be crucial for us.

-  For a non-negative integer ℓ , the vertex set of $\overset{\curvearrowright}{\ell}$ consists of $\ell + 2$ vertices $v_1, \dots, v_{\ell+2}$, ordered so that $v_i \prec v_j$ for $i < j$, and the edge set consists of one edge $v_1 v_{\ell+2}$.
-  For a non-negative integer ℓ , the vertex set of $\overset{\curvearrowright}{\ell} \overset{\curvearrowright}{\ell} \overset{\curvearrowright}{\ell}$ consists of $3\ell + 2$ vertices $v_1, \dots, v_{3\ell+2}$, ordered so that $v_i \prec v_j$ for $i < j$, and the edge set consists of one edge $v_{\ell+1} v_{2\ell+2}$.
-  The vertex set of \curvearrowright consists of three vertices v_1, v_2, v_3 ordered so that $v_1 \prec v_2 \prec v_3$, and the edge set consists of two edges $v_1 v_2, v_1 v_3$.
-  The vertex set of \curvearrowright_1 consists of four vertices v_1, v_2, v_3, v_4 ordered so that $v_1 \prec v_2 \prec v_3 \prec v_4$, and the edge set consists of two edges $v_1 v_2, v_1 v_3$.
-  For a non-negative integer ℓ , the vertex set of $\overset{\curvearrowright}{\ell}$ consists of $\ell + 3$ vertices $v_1, \dots, v_{\ell+3}$ ordered so that $v_i \prec v_j$ for $i < j$, and the edge set consists of two edges $v_{\ell+1} v_{\ell+2}, v_{\ell+1} v_{\ell+3}$.
-  The vertex set of \curvearrowright consists of four vertices v_1, v_2, v_3, v_4 , ordered so that $v_1 \prec v_2 \prec v_3 \prec v_4$, and the edge set consists of two edges $v_1 v_4, v_2 v_3$.

A graph G is chordal if it contains no cycle on at least 4 vertices as an induced subgraph. The following observation by Hajebi et al. [13] will be useful.

► **Proposition 1** (Hajebi et al. [13]). *A graph G is chordal if and only if there exists a linear ordering \prec of $V(G)$ such that G (as an ordered graph) is \curvearrowright -free.*

List coloring. The instance of the LIST COLORING problem is (G, L) , where G is a graph and $L : V(G) \rightarrow 2^{\mathbb{N}}$ is a *list function*. We ask whether G admits a proper coloring such that every vertex receives a color from its list. For a fixed integer k , the LIST k -COLORING is a restriction of LIST COLORING, where every list is a subset of $[k]$.

The following result of Edwards [8] will be useful for us.

► **Theorem 2** (Edwards [8]). *For every $k \in \mathbb{N}$, there is a polynomial-time algorithm that solves every instance (G, L) of LIST k -COLORING such that for every $v \in V(G)$, it holds $|L(v)| \leq 2$.*

3 Polynomial-time algorithms

In both algorithm presented in this section we will use two simple reduction rules. Let (G, L) be an instance of LIST k -COLORING for some k .

(R1) If there is $v \in V(G)$ such that $L(v) = \emptyset$, then return NO.

(R2) For $v \in V(G)$ such that $L(v) = \{a\}$, remove a from lists of all neighbors of v , and remove v from G .

Clearly, the reduction rules are safe and their exhaustive application can be performed in polynomial time.

3.1 H -free graphs where $|E(H)| = 1$

In this section we prove the following result.

► **Theorem 3.** *Let H be a fixed ordered graph with one edge. For every k , the LIST k -COLORING problem is polynomial-time solvable in H -free ordered graphs.*

Proof. Observe that every graph H with one edge is an induced subgraph of $\ell \overset{\curvearrowright}{\ell} \ell$, for some $\ell \leq |V(H)|$. Thus, it is sufficient to consider the case that $H = \ell \overset{\curvearrowright}{\ell} \ell$. We can also assume that $\ell \geq 1$, as for $\ell = 0$ the problem is trivial: all instances are edgeless.

We proceed by induction on k ; the case $k \leq 2$ is obvious. Thus suppose that $k \geq 3$ and the claim holds for $k - 1$. Let (G, L) be an n -vertex instance of LIST k -COLORING such that G is an $\ell \overset{\curvearrowright}{\ell} \ell$ -free ordered graph.

Branching. We first verify if there is a list coloring $c : (G, L) \rightarrow [k]$ such that some color $i \in [k]$ is used on at most $2\ell - 1$ vertices; for every $i \in [k]$, and for every $A_i \subseteq V(G)$ of size at most $2\ell - 1$, we create an instance (G_1, L_1) from (G, L) as follows:

1. for every $v \in A_i$, we remove from $L(v)$ all colors but i ,
2. for every $v \in V(G) \setminus A_i$, we set its list to $L(v) \setminus \{i\}$,
3. we exhaustively apply reduction rules (R1) and (R2).

Note that now (G_1, L_1) can be seen as an instance of LIST $(k - 1)$ -COLORING, and thus, it can be solved in polynomial time by the inductive assumption.

So since now we can assume that for every list coloring $c : (G, L) \rightarrow [k]$, every color $i \in [k]$ is used on at least 2ℓ vertices. Now we branch on the choice of first and last ℓ vertices in each color, i.e., for every $(2k)$ -tuple $\mathbf{A} = (A_1, \dots, A_k, B_1, \dots, B_k)$ of pairwise disjoint sets, each of size ℓ , where A_i precedes B_i , we create from (G, L) an instance $(G_{\mathbf{A}}, L_{\mathbf{A}})$ as follows:

1. for every $i \in [k]$, for $v \in A_i \cup B_i$, we remove from $L(v)$ all colors but i ,
2. for every $i \in [k]$, for every $v \in V(G) \setminus (A_i \cup B_i)$, if v precedes the last vertex of A_i or the first vertex of B_i precedes v , then we remove i from $L(v)$,
3. we exhaustively apply reduction rules.

Consider such an instance $(G_{\mathbf{A}}, L_{\mathbf{A}})$. Let $i \in [k]$, and let X_i be the set of vertices v in $G_{\mathbf{A}}$ with $i \in L_{\mathbf{A}}(v)$.

► **Claim 4 (♠).** For every $i \in [k]$, the graph $G_{\mathbf{A}}[X_i]$ is $\ell \overset{\curvearrowright}{\ell}$ -free.

Dynamic programming. So since now, we assume that we are dealing with an instance that satisfies the property from Claim 4. For simplicity, let us denote the current instance (G, L) , omitting the subscript \mathbf{A} .

Let $v_1 \prec \dots \prec v_n$ denote the vertices of G . Moreover, for $j \in [n]$, we define $V_j = \{v_1, \dots, v_j\}$. Let $j \in [n]$, let \mathcal{C}_j denote the family of all k -tuples of pairwise disjoint independent subsets of V_j , each of size at most ℓ . We say that a list coloring $c : (G[V_j], L) \rightarrow [k]$ is *compatible with* $(C_1, \dots, C_k) \in \mathcal{C}_j$ if, for every $i \in [k]$, the following hold: if $|C_i| \leq \ell - 1$, then $c^{-1}(i) = C_i$, and if $|C_i| = \ell$, then C_i is the set of ℓ last vertices of V_j colored with i .

For every $j \in [n]$, we will construct a table Tab_j of entries, indexed by all elements of \mathcal{C}_j ; note that the total size of all tables is bounded by $n \cdot n^{kw}$. We will set $\text{Tab}_j[(C_1, \dots, C_k)]$ to true if and only if there exists a list coloring of $(G[V_j], L)$ that is compatible with (C_1, \dots, C_k) . Clearly, (G, L) is a yes-instance of LIST k -COLORING if and only if Tab_n contains at least one true entry. So it remains to show how to compute the entries efficiently.

First, we set $\text{Tab}_1[C_1, \dots, C_k]$ to true if and only if there is $i \in L(v_1)$ such that $C_i = \{v_1\}$ and $C_t = \emptyset$ for $t \neq i$. So now assume that $j \geq 2$, we computed all entries for Tab_{j-1} , and let $(C_1, \dots, C_k) \in \mathcal{C}_j$. If there is no $t \in L(v_j)$ such that $v_j \in C_t$, we set $\text{Tab}_j[C_1, \dots, C_k]$ to false. So now assume that such t exists; clearly it is unique. We set $\text{Tab}_j[(C_1, \dots, C_k)]$ to true if and only if at least one of following conditions is satisfied.

1. $\text{Tab}_{j-1}[(C_1, \dots, C_t \setminus \{v_j\}, \dots, C_k)]$ is true.
2. $|C_t| = \ell$ and there exists $x \in V_{j-1}$ such that: (i) x precedes all the vertices of C_t , (ii) x is non-adjacent to C_t , and (iii) for $C'_t = C_t \setminus \{v_j\} \cup \{x\}$, we have that $\text{Tab}_{j-1}[(C_1, \dots, C_{t-1}, C'_t, C_{t+1}, \dots, C_k)]$ is true.

Clearly, all table entries can be computed in polynomial time in n .

Correctness. The full proof of the correctness can be found in the full version of the paper (♠). Let us only mention how we use $\overset{\ell}{\curvearrowright}$ -freeness here. In general, it might happen that for some j , there exists $x \in V_{j-1}$ such that $C_t \cup \{x\}$ is an independent set and there is a coloring c of V_{j-1} compatible with $(C_1, \dots, C_{t-1}, C_t \setminus \{v_j\} \cup \{x\}, C_{t+1}, \dots, C_k)$, but such a coloring cannot be extended to V_j by $c(v_j) = t$ since there is some neighbor y of v_j such that $c(y) = t$. In our case, existence of such a vertex y yields an induced copy of $\overset{\ell}{\curvearrowright}$ on vertices of X_t , which contradicts Claim 4. This completes the proof. ◀

3.2 $\overset{\ell}{\curvearrowright}$ -free graphs

► **Theorem 5.** *For every fixed ℓ , LIST 4-COLORING can be solved in polynomial time on $\overset{\ell}{\curvearrowright}$ -free ordered graphs.*

Proof. Let G be an ordered n -vertex $\overset{\ell}{\curvearrowright}$ -free graph, given with a list function $L : V(G) \rightarrow 2^{[4]}$. We start with checking if G contains a K_5 and, if so, we immediately reject the instance. This can be done in polynomial time by brute force.

First, suppose there is a list 4-coloring of G such that for some $i \in [4]$, at most $\ell - 1$ vertices receive color i . We might exhaustively guess such an i and the set of vertices colored i , remove them from the graph, and remove i from the lists of all remaining vertices. This way we reduced the problem to solving a polynomial number of instances of LIST 3-COLORING, each of which can be solved in polynomial time by using the result of Hajebi et al. [13, Theorem 22], mentioned in Table 1.

So, from now on let us focus on looking for a list 4-coloring, where each color appears at least ℓ times. The algorithm consists of four main phases.

Phase 1. For each $i \in [4]$, we exhaustively guess the first ℓ vertices that will receive color i . More precisely, for every 4-tuple $\mathcal{A} = (A_1, A_2, A_3, A_4)$ of pairwise disjoint independent sets, each of size ℓ , we create an instance as follows.

1. For every $i \in [4]$ and every vertex $v \in A_i$, we set the list of v to $L(v) \cap \{i\}$.
2. For every $i \in [4]$, we remove i from lists of all vertices not in A_i that precede the last vertex of A_i .
3. We exhaustively apply reduction rules.

Observe that as we assumed that in any list 4-coloring of G there are at least ℓ vertices in each color, (G, L) is a yes-instance if and only if at least one of the created instances is a yes-instance. Furthermore, the number of branches in Phase 1 is at most $n^{4\ell}$, which is polynomial. Consider one such branch for $\mathcal{A} = (A_1, A_2, A_3, A_4)$ and let (G_1, L_1) be the current instance. Before we proceed to Phase 2, let us analyze the properties of (G_1, L_1) .

▷ **Claim 6 (♠).** Let $x, y, z \in V(G_1)$ be such that $\{x, y, z\}$ induces a \curvearrowright , where $x \prec y \prec z$. Then $L_1(x) \cap L_1(y) \cap L_1(z) = \emptyset$.

Let us classify forward neighbors of a vertex v . We say that a forward neighbor u of v is *safe* (in (G_1, L_1)) if $L_1(u) \cap L_1(v) \neq \emptyset$, and otherwise it is *dangerous*. Now, for every vertex we can bound the number of its safe forward neighbors.

▷ **Claim 7 (♠).** Let $v \in V(G_1)$ and let $i \in L_1(v)$. Then v has at most 3 forward neighbors whose list contains i . Consequently, v has at most 12 safe forward neighbors.

We say that a vertex v is *bad* if it has at least $3\ell + 4$ dangerous forward neighbors. If v is not bad, then it is *good*. Let **Bad** and **Good** denote, respectively, the sets of bad and good vertices.

Let us make a few comments about bad and good vertices. First, every $v \in \mathbf{Bad}$ has list of size 2. Indeed, the reduction rules assert that there are no vertices with lists of size at most one, and if the list of v has at least three elements, then there are no vertices with lists disjoint with $L_1(v)$ (and in particular v has no dangerous forward neighbors). Second, for analogous reasons, all dangerous forward neighbors of a bad vertex v have the same list, i.e., $[4] \setminus L_1(v)$. Finally, if v is good, Claim 7 implies that v has at most $12 + (3\ell + 3) = 3\ell + 15$ forward neighbors.

▷ **Claim 8 (♠).** Let v be a bad vertex. Let B be the set of all vertices $u \in \mathbf{Bad}$ with $u \prec v$ for which $L_1(u) \neq L_1(v)$. Then v is non-adjacent to at most $\mathbf{Ram}(5, \ell) - 1$ vertices of B .

Phase 2. We proceed further with the instance (G_1, L_1) . We consider two cases depending on the size of **Good**. First, if $|\mathbf{Good}| < \mathbf{Ram}(5, \ell)$, then we can exhaustively guess coloring on **Good**. More precisely, for each $c : \mathbf{Good} \rightarrow [4]$ we create a corresponding instance by setting the list of every $v \in \mathbf{Good}$ to $L_1(v) \cap \{c(v)\}$ and then exhaustively applying reduction rules.

The number of instances created this way is at most $4^{\mathbf{Ram}(5, \ell) - 1}$, which is a constant as ℓ is fixed. Since every bad vertex has list of size 2, in any produced instance, every vertex must have a list of size 2. Here we emphasize that the vertices in the newly created instances might become good, but they were bad before the guessing and their lists could only shrink. Therefore, applying Theorem 2, we can solve each instance in polynomial time. This completes the proof in this case.

So since now, we can assume that $|\mathbf{Good}| \geq \mathbf{Ram}(5, \ell)$. Let D be the set of first $\mathbf{Ram}(5, \ell)$ good vertices. We exhaustively guess the coloring of $D \cup N^+(D)$, i.e., for every $c : D \cup N^+(D) \rightarrow [4]$, we create a corresponding instance by setting the list of every $v \in D \cup N^+(D)$ to $L_1(v) \cap \{c(v)\}$ and applying reduction rules exhaustively.

Recall that every good vertex has at most $3\ell + 15$ forward neighbors, and thus the size of $D \cup N^+(D)$ is bounded by $|D| + |D| \cdot (3\ell + 15) = \mathbf{Ram}(5, \ell) \cdot (3\ell + 16)$. Therefore, the number of instances created in Phase 2 is at most $4^{\mathbf{Ram}(5, \ell) \cdot (3\ell + 16)}$, which is again constant for fixed ℓ .

Consider one such instance and denote it by (G_2, L_2) . We partition the vertex set of G_2 into sets P, S called *prefix* and *suffix* so that in P are all vertices that in G_1 precede the last vertex of D , and S contains the remaining vertices of G_2 . Observe that all vertices in P were bad at the end of Phase 1, i.e., before we guessed a coloring on $D \cup N^+(D)$. In the instance (G_2, L_2) they do not have to be bad, but we do not care if they are bad now. We will only use the facts that (i) they have lists of size 2 and (ii) Claim 8 holds for all vertices of P . Indeed, note that reduction rules did not increase the size of lists or the number of non-neighbors.

Moreover, observe that the graph induced by S is \curvearrowright -free. Indeed, since all vertices of S were not removed by reduction rules applied after the branching in Phase 2, there are no edges between D and S . Moreover, since G has no K_5 , there is an independent set $D' \subseteq D$ of size ℓ . Consequently, if $G[S]$ contained an induced \curvearrowright , then, together with D' , it would form an induced $\ell\curvearrowright$ in G , a contradiction. We proceed to the third phase.

Phase 3. Let (G_2, L_2) be an instance produced in Phase 2. For $X \in \binom{[4]}{2}$, by P_X we denote the set of vertices of P with list X ; recall that each vertex of P is bad and thus has list of size 2. If, for some X , the set P_X does not induce a bipartite graph, we can immediately terminate the current call and reject. So assume that, for each X , the graph $G[P_X]$ is bipartite. We consider two cases.

If $|P_X| < 2\ell$, then we exhaustively guess the coloring of P_X , i.e., for every $c : P_X \rightarrow X$, we create a corresponding instance by setting the list of every $v \in P_X$ to $L_2(v) \cap \{c(v)\}$ and applying reduction rules exhaustively. The number of instances created in this case is at most $2^{2\ell-1}$, i.e., a constant.

Now let us assume that $|P_X| \geq 2\ell$. Denote $X = \{i, j\}$, and suppose that we are dealing with a yes-instance, i.e., there is a coloring c of G_2 respecting lists L_2 . Observe that one of the following holds.

- (C1) For some $\iota \in \{i, j\}$, at most $\ell - 1$ vertices of P_X receive color ι in c .
- (C2) Each of colors i, j appears at least ℓ times on P_X in c .

We create the following instances, corresponding to the cases above.

- (I1) For every $\iota \in X$ and every set $U \subseteq P_X$ of size at most $\ell - 1$, we create a corresponding instance by setting the list of every vertex $v \in U$ to $L_2(v) \cap \{\iota\}$, and the list of every vertex $v \in P_X \setminus U$ to $L_2(v) \setminus \{\iota\}$.
- (I2) For every pair $(U_{X,i}, U_{X,j})$, where $U_{X,i}, U_{X,j}$ are disjoint independent sets contained in P_X , each of size ℓ , we create a corresponding instance in the following way: (i) for every vertex $v \in U_{X,i}$, we set its list to $L_2(v) \cap \{i\}$, (ii) for every vertex $v \in U_{X,j}$, we set its list to $L_2(v) \cap \{j\}$, and (iii) for every vertex u of $P_X \setminus U_{X,i}$ preceding the last vertex of $U_{X,i}$, we remove i from $L_2(u)$, and (iv) and for every vertex u of $P_X \setminus U_{X,j}$ preceding the last vertex of $U_{X,j}$, we remove j from $L_2(u)$.

We exhaustively apply reduction rules to all created instances. The number of instances created in this case is at most $n^{2\ell}$.

The intended role of the set $U_{X,i}$ (resp., $U_{X,j}$) in (I2) is that it contains the first ℓ vertices of P_X colored i (resp., j). If (G_2, L_2) is a yes-instance, and there is coloring satisfying (C1), then at least one of the instances created in (I1) is a yes-instance, and if there is a coloring satisfying (C2), then at least one of the instances created in (I2) is a yes-instance. Thus, if (G_2, L_2) is a yes-instance, then at least one of the instances created in Phase 3 is a yes-instance. The total number of instances created in this phase is at most $\binom{4}{2} \cdot n^{2\ell}$, i.e., polynomial in n .

Let us analyze an instance (G_3, L_3) created in Phase 3. Recall that all vertices of $P \cap V(G_3)$ have lists of size 2.

▷ **Claim 9** (♠). For any two vertices $v, v' \in P \cap V(G_3)$, either $L_3(v) = L_3(v')$ or $L_3(v) \cap L_3(v') = \emptyset$. In particular, at most two distinct lists might appear among the vertices of P in (G_3, L_3) .

We proceed to Phase 4. We emphasize that sets P and **Good** are not redefined with respect to the current instance.

Phase 4. Let (G_3, L_3) be an instance given by Phase 3. By Claim 9, there are at most two distinct lists on $P \cap V(G_3)$, and by symmetry, we can assume that every vertex of P in (G_3, L_3) has list either $\{1, 2\}$ or $\{3, 4\}$. Furthermore, if for $X = \{i, j\} \in \{\{1, 2\}, \{3, 4\}\}$, we have $P_X \cap V(G_3) \neq \emptyset$, then (G_3, L_3) corresponds to a branch of type (II), i.e., a branch for the sets $U_{X,i}, U_{X,j}$. Let Y be the set of vertices in G_3 that consists of safe forward neighbors of $U_{\{1,2\},1}$ (if $P_{\{1,2\}} \cap V(G_3) \neq \emptyset$) and safe forward neighbors of $U_{\{3,4\},3}$ (if $P_{\{3,4\}} \cap V(G_3) \neq \emptyset$).

For every $c : Y \rightarrow [4]$, we create a corresponding instance as follows:

1. for every $v \in Y$, we set the list of v to $L_3(v) \cap \{c(v)\}$,
2. we exhaustively apply reduction rules,
3. we remove all edges whose one endpoint has list $\{1, 2\}$ and the other $\{3, 4\}$, and at least one endpoint is in P .

By Claim 7, each vertex has at most 12 safe forward neighbors and thus, we guess a coloring on at most $12 \cdot 2\ell = 24\ell$ vertices. Therefore, the number of instances created in this phase is at most $4^{24\ell}$. Note that the edges removed in the final step do not matter for coloring, so we obtain an equivalent instance. Let (G_4, L_4) be an instance obtained in this phase. We point out that in general, if we remove edges in a (ordered) graph which is F -free for some graph F , it does not have to stay F -free. However, in our case, we will show that the resulting graph is even \curvearrowright -free.

▷ **Claim 10** (♠). G_4 is \curvearrowright -free.

Summing up, the instances obtained in Phase 4 are \curvearrowright -free and thus chordal. Chordal graphs of bounded clique number have bounded treewidth, and thus, each obtained instance can be solved in polynomial time by standard dynamic programming on tree decomposition (see for example [6]). This completes the proof. ◀

Let us conclude this section with brief discussion why the approach from Theorem 5 fails for more than 4 colors. In general, in LIST k -COLORING, it is beneficial if adjacent vertices have intersecting lists. This way deciding on a color of one vertex allows to shrink the list of the other, see e.g. [18, 1]. On the other hand, adjacent vertices with disjoint lists of size at least 2 often appear in hardness proofs of LIST k -COLORING for $k \geq 4$, see Section 4 or, e.g., [1].

In the proof of Theorem 5, we called (forward) neighbors of v with list disjoint from $L(v)$ *dangerous* and dealing with them was the most technically involved part of the argument. The crucial property that was very useful here is that every dangerous forward neighbor of v has the same list, i.e., $[4] \setminus L(v)$. This enforces many constraints on possible lists of vertices.

Already for LIST 5-COLORING, a vertex v can have several types of forward neighbors with list disjoint from $L(v)$, and this is the main obstacle in generalizing our approach to more than 4 colors.

4 Hardness results

The proof of the following theorem can be found in the full version of the paper.

► **Theorem 11** (♠). LIST 4-COLORING is NP-hard on \curvearrowright_1 -free ordered graphs.

74:10 List Coloring Ordered Graphs with Forbidden Induced Subgraphs

In this section we show the following hardness result.

► **Theorem 12.** *LIST 4-COLORING is NP-hard on \curvearrowright -free graphs.*

The construction. First, let us describe the main building blocks in our reduction. For positive integers ℓ, ℓ' , a *link* is a tuple $(F, (x_1, \dots, x_\ell), (y_1, \dots, y_{\ell'}))$, where F is an ordered graph, and (x_1, \dots, x_ℓ) and $(y_1, \dots, y_{\ell'})$ are tuples of vertices of F , such that:

1. x_1, \dots, x_ℓ are, in this order, the first ℓ vertices of F ,
2. $y_1, \dots, y_{\ell'}$ are, in this order, the last ℓ' vertices of F ,
3. sets $\{x_1, \dots, x_\ell\}$ and $\{y_1, \dots, y_{\ell'}\}$ are disjoint and independent,
4. F is \curvearrowright -free.

The vertices x_1, \dots, x_ℓ (resp., $y_1, \dots, y_{\ell'}$) are called *input* (resp., *output*) vertices of the link.

The properties of links allow us combine them, without creating the forbidden subgraph. This operation, that we call *chaining*, is formally described in the following lemma.

► **Lemma 13 (♠).** *Let $(F^1, (x_1^1, \dots, x_{\ell'}^1), (y_1^1, \dots, y_{\ell'}^1))$ and $(F^2, (x_1^2, \dots, x_{\ell'}^2), (y_1^2, \dots, y_{\ell'}^2))$ be two links on disjoint sets of vertices. The ordered graph F obtained from F_1 and F_2 by identifying y_i^1 with x_i^2 for each $i \in [\ell']$ is a link.*

In our reduction we will use gadgets that are links enriched by a list function. We will also speak about chaining gadgets – in such a situation, we always ensure that the lists of vertices that are identified are equal. Thus, the operation is straightforward.

The main gadget used in our hardness proof is a *NAE gadget*.

► **Definition 14 (NAE gadget).** *Let $n \in \mathbb{N}$ and let $I \subseteq \binom{[n]}{3}$ be a set of pairwise disjoint subsets of $[n]$, each of size 3. A NAE gadget (for I) is a tuple $(C, (x_1, \dots, x_n), (y_1, \dots, y_n), L)$, where $(C, (x_1, \dots, x_n), (y_1, \dots, y_n))$ is a link and $L : V(C) \rightarrow 2^{[4]}$ is a list function, such that:*

- (C1) *For every $i \in [n]$, it holds that $L(x_i) = L(y_i) = \{1, 2\}$.*
- (C2) *For every $f : \{x_1, \dots, x_n\} \rightarrow \{1, 2\}$ such that for every $\{i, j, k\} \in I$, it holds that $\{f(x_i), f(x_j), f(x_k)\} = \{1, 2\}$, we can extend f to a coloring of (C, L) .*
- (C3) *For every coloring f of (C, L) , it holds that:*
 - a. *for every $i \in [n]$, $f(x_i) = f(y_i)$,*
 - b. *for every $\{i, j, k\} \in I$, it holds that $\{f(x_i), f(x_j), f(x_k)\} = \{1, 2\}$*

For simplicity of notation, we will denote the gadget by C .

Intuitively, the gadget plays two roles. First, it transfers the coloring of input vertices to their corresponding output vertices. Second, it makes sure that for every triple in I , the input/output vertices corresponding to that triple are not monochromatic or, in other words, *not all equal* (NAE). The following lemma is the main technical ingredient of our hardness proof.

► **Lemma 15.** *Let $n \in \mathbb{N}$ and let $I \subseteq \binom{[n]}{3}$ be a set of pairwise disjoint subsets of $[n]$, each of size 3. In time polynomial in n , we can construct a NAE gadget for I .*

Let us postpone the proof of Lemma 15, and first let us show Theorem 12 assuming that Lemma 15 holds.

Proof of Theorem 12. We reduce from POSITIVE NAE-3-SAT. An instance of this problem consists of a set of boolean variables and a set of clauses, each containing exactly three variables (none of them is negated). We ask if there exists a truth assignment in which each clause contains a true and a false variable, i.e., for each clause, the variables in it are not all equal.

Let Φ be an instance of POSITIVE NAE-3-SAT with variables v_1, \dots, v_n and clauses C_1, \dots, C_m . We can assume that every variable appears at most four times [7]. In polynomial time, we will construct an instance (G, L) of LIST 4-COLORING such that:

1. G is an ordered \curvearrowright -free graph,
2. (G, L) admits a proper coloring if and only if Φ is satisfiable.

As each clause has three variables and each variable appears in at most four clauses, we can greedily partition the set of clauses into 10 pairwise disjoint subsets; denote them by $\mathcal{C}_1, \dots, \mathcal{C}_{10}$ (we can think of this as a greedy edge coloring of a hypergraph: for each hyperedge, each its vertex blocks at most three colors). For $s \in [10]$, we define $I_s \subseteq \binom{[n]}{3}$ so that $\{i, j, k\} \in I_s$, if and only if there is a clause $\{v_i, v_j, v_k\}$ in \mathcal{C}_s .

Now we are ready to construct the instance (G, L) . We start with introducing NAE gadgets $C_{I_1}, \dots, C_{I_{10}}$, for, respectively, I_1, \dots, I_{10} . For $s \in [10]$, let us denote the input (resp., output) vertices of C_{I_s} by (x_1^s, \dots, x_n^s) (resp., (y_1^s, \dots, y_n^s)). Next, we chain gadgets $C_{I_1}, \dots, C_{I_{10}}$, i.e., for every $s \in [9]$, we identify the output vertices of C_{I_s} with input vertices of $C_{I_{s+1}}$. This completes the construction of (G, L) .

As each gadget is in particular a link, a repeated application of Lemma 13 yields that G is \curvearrowright -free. Thus, we are left with proving the equivalence of instances.

Suppose that Φ is satisfiable and let $\psi : \{v_1, \dots, v_n\} \rightarrow \{\text{true}, \text{false}\}$ be a satisfying assignment. For every $i \in [n]$, we set $f(x_i^1) = 1$ if $\psi(v_i) = \text{true}$, and $f(x_i^1) = 2$ if $\psi(v_i) = \text{false}$. Since for every clause $\{v_i, v_j, v_k\} \in \mathcal{C}_1$, we have $\{\psi(v_i), \psi(v_j), \psi(v_k)\} = \{\text{true}, \text{false}\}$, and thus, for every $\{i, j, k\} \in I_1$ we have $\{f(x_i^1), f(x_j^1), f(x_k^1)\} = \{1, 2\}$, we can extend this coloring to a coloring of the whole gadget C_{I_1} . Since for every $i \in [n]$, we have that $f(x_i^1) = f(y_i^1) = f(x_i^2)$, we can repeat the reasoning inductively for every $s \in [10]$ and extend f to all vertices of G .

Now suppose that there is a proper coloring f of (G, L) . We define $\psi : \{v_1, \dots, v_n\} \rightarrow \{\text{true}, \text{false}\}$ so that $\psi(v_i) = \text{true}$ if $f(x_i^1) = 1$ and $\psi(v_i) = \text{false}$ if $f(x_i^1) = 2$. Let us verify that ψ satisfies Φ . Suppose that there is a clause $C_q = \{v_i, v_j, v_k\}$ which is not satisfied, i.e., all v_i, v_j, v_k are either set **true** or set **false** by ψ . By property (C3) (a), this means that for every $s \in [10]$, we have $f(x_i^s) = f(x_j^s) = f(x_k^s)$. As $C_q \in \mathcal{C}_s$ for some $s \in [10]$, we obtain a contradiction with property (C3) (b) of a NAE gadget. This completes the proof. \blacktriangleleft

From NOT-ccc gadgets to NAE gadgets. We will construct NAE gadgets in several steps. First, let us show that in order to construct a NAE gadget, it is sufficient to construct its restricted variant, called *NOT-ccc gadget*.

► **Definition 16** (NOT-ccc gadget). *Let $c \in \{1, 2\}$. Let $n \in \mathbb{N}$ and let $I \subseteq \binom{[n]}{3}$ be a set of pairwise disjoint subsets of $[n]$, each of size 3. A NOT-ccc gadget (for I) is a tuple $(C^c, (x_1, \dots, x_n), (y_1, \dots, y_n), L)$, where $(C^c, (x_1, \dots, x_n), (y_1, \dots, y_n))$ is a link and $L : V(C^c) \rightarrow 2^{[4]}$ is a list function, such that:*

- (C1) *For every $i \in [n]$, it holds that $L(x_i) = L(y_i) = \{1, 2\}$.*
- (C2) *For every $f : \{x_1, \dots, x_n\} \rightarrow \{1, 2\}$ such that for every $\{i, j, k\} \in I$, it holds that $\{f(x_i), f(x_j), f(x_k)\} \neq \{c\}$, we can extend f to a coloring of (C^c, L) .*
- (C3) *For every coloring f of (C^c, L) , it holds that:*
 - a. *for every $i \in [n]$, $f(x_i) = f(y_i)$,*
 - b. *for every $\{i, j, k\} \in I$, it holds that $\{f(x_i), f(x_j), f(x_k)\} \neq \{c\}$*

As usual, we will simply denote the gadget by C^c .

74:12 List Coloring Ordered Graphs with Forbidden Induced Subgraphs

Let us emphasize the difference between a NAE gadget and a NOT-*ccc* gadget. The first one is responsible for ensuring that for each triple in I , the colors assigned to the corresponding vertices are *not all 1* and *not all 2*. The role of a NOT-*ccc* gadget is to ensure that the vertices corresponding to each triple in I are not all colored c , where $c \in \{1, 2\}$. Thus, a NAE gadget is simultaneously a NOT-111 gadget and a NOT-222 gadget.

As the gadget not only forbids some colorings, but also copies the coloring of the input to the output, we immediately obtain the following observation.

► **Observation 17.** *Chaining a NOT-111 gadget and a NOT-222 yields a NAE gadget.*

Thus, in order to prove Lemma 15, it is sufficient to build a NOT-*ccc* gadget for $c \in \{1, 2\}$.

Permutation gadgets. On the way to the proof of Lemma 15, we will first construct another type of a gadget.

► **Definition 18** (Permutation gadget). *Let $\ell \in \mathbb{N}$, and let $\sigma : [\ell] \rightarrow [\ell]$ be a permutation. A permutation gadget (for σ) is a tuple $(P_\sigma, (x_1, \dots, x_\ell), (y_1, \dots, y_\ell), L)$, where $(P_\sigma, (x_1, \dots, x_\ell), (y_1, \dots, y_\ell))$ is a link and $L : V(P_\sigma) \rightarrow 2^{[4]}$ is a list function, such that:*

(P1) *For every $i \in [\ell]$, it holds that $L(x_i) = L(y_i) = \{1, 2\}$.*

(P2) *Every mapping $f : \{x_1, \dots, x_\ell\} \rightarrow \{1, 2\}$ can be extended to a coloring of (P_σ, L) .*

(P3) *For every coloring f of (P_σ, L) , for every $i \in [\ell]$, it holds that $f(x_i) = f(y_{\sigma(i)})$.*

Again, we will shortly denote a permutation gadget for σ by P_σ . Intuitively, the role of a permutation gadget is to transfer the coloring of the input to the output, but after applying σ on it. We will show that we can efficiently construct permutation gadgets.

► **Lemma 19.** *Let $\ell \in \mathbb{N}$, and let $\sigma : [\ell] \rightarrow [\ell]$ be a permutation. In time polynomial in ℓ , we can construct a permutation gadget for σ .*

Let us break the proof of Lemma 19 in three steps.

First, we observe that chaining permutation gadgets yields a permutation gadget for the composition of permutations: this follows directly from Lemma 13 and the definition of a permutation gadget.

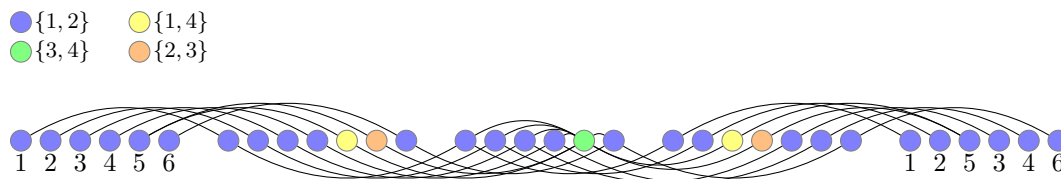
► **Observation 20.** *For $\ell \in \mathbb{N}$, let $\sigma, \sigma' : [\ell] \rightarrow [\ell]$ be two permutations and let $P_\sigma, P_{\sigma'}$ be their corresponding permutation gadgets. Let P be obtained by chaining P_σ with $P_{\sigma'}$. Then, P is a permutation gadget for $\sigma' \circ \sigma$.*

Thus, in order to show Lemma 19, it is enough to construct permutation gadgets for some basic permutations that can be composed into an arbitrary permutation. These basic permutations are *rotations*. For integers $j \leq k \leq \ell$, a rotation, denoted by $\langle \ell; j, k \rangle$ is a permutation of ℓ such that:

$$\langle \ell; j, k \rangle(i) = \begin{cases} i & \text{if } i < j \text{ or } i > k \\ i + 1 & \text{if } j \leq i < k \\ j & \text{if } i = k. \end{cases}$$

In other words, $\langle \ell; j, k \rangle$ performs a cyclic shift on elements $\{j, \dots, k\}$, leaving the remaining ones untouched. In particular, if $j = k$, then $\langle \ell; j, k \rangle$ is an identity.

► **Observation 21** (♠). *Every permutation $\sigma : [\ell] \rightarrow [\ell]$ can be written as the composition of $\ell - 1$ rotations.*



■ **Figure 1** A permutation gadget for the rotation $\langle \ell; 3, 5 \rangle$.

Thus, in order to prove Lemma 19, it is sufficient to show how to construct permutation gadgets for rotations. We do this in the next lemma, consult also Figure 1.

▶ **Lemma 22.** *Let $j < k \leq \ell \in \mathbb{N}$ be integers. In time polynomial in ℓ , we can construct a permutation gadget for $\langle \ell; j, k \rangle$.*

Proof. The gadget has $5\ell + 2$ vertices. We partition them into five sets S_1, \dots, S_5 such that $|S_1| = |S_3| = |S_5| = \ell$ and $|S_2| = |S_4| = \ell + 1$ in a natural way: the first ℓ vertices belong to S_1 , next $\ell + 1$ vertices belong to S_2 and so on. For $p \in [5]$, the i -th vertex of S_p is denoted by s_i^p .

The lists of vertices of the gadget are as follows:

$$L(s) = \begin{cases} \{1, 4\} & \text{if } s \in \{s_k^2, s_j^4\} \\ \{2, 3\} & \text{if } s \in \{s_{k+1}^2, s_{j+1}^4\} \\ \{3, 4\} & \text{if } s = s_k^3 \\ \{1, 2\} & \text{otherwise.} \end{cases}$$

The edge set of the gadget consists of the following elements:

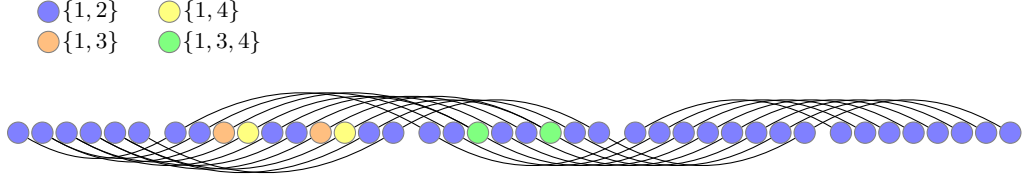
- Between S_1 and S_2 :** $s_i^1 s_i^2$ for $i \in [k]$ and $s_i^1 s_{i+1}^2$ for $i \in [k, \ell]$,
- Between S_2 and S_3 :** $s_i^2 s_i^3$ for $i \in [k]$ and $s_i^2 s_{i-1}^3$ for $i \in [k + 1, \ell + 1]$,
- Inside S_3 :** $s_k^3 s_i^3$ for $i \in [\ell] \setminus \{k\}$,
- Between S_3 and S_4 :** $s_i^3 s_i^4$ for $i \in [j - 1]$ and $s_i^3 s_{i+2}^4$ for $i \in [j, k - 1]$ and $s_k^3 s_j^4, s_k^3 s_{j+1}^4$, and $s_i^3 s_{i+1}^4$ for $i \in [k + 1, \ell]$,
- Between S_4 and S_5 :** $s_i^4 s_i^5$ for $i \in [j]$ and $s_i^4 s_{i-1}^5$ for $i \in [j + 1, \ell + 1]$.

This completes the construction of the gadget $P_{\langle \ell; j, k \rangle}$: the input vertices are S_1 and the output vertices are S_5 . It can be shown that $P_{\langle \ell; j, k \rangle}$ satisfies the desired properties (♠), which completes the proof. ◀

Now, Lemma 19 follows directly from Observation 20, Observation 21, and Lemma 22,

Indicator gadgets. The next type of a gadget that we will need is an *indicator*.

Throughout this paragraph, let $n \in \mathbb{N}$ and let $I \subseteq \binom{[n]}{2}$ be a set of pairwise disjoint pairs from $[n]$, each of the form $\{i, i + 1\}$, and let $N = n + |I|$. Let us define a bijection $\gamma : [n] \cup I \rightarrow [n + |I|]$. We will do it by specifying the relative order of images of particular elements of $[n] \cup I$. The images of elements in $[n]$ are in the natural order, i.e., $\gamma(1) < \gamma(2) < \dots < \gamma(n)$. For each $\{i, i + 1\} \in I$, the image of this pair is placed between $\gamma(i)$ and $\gamma(i + 1)$. Note that, as the pairs in I are of the form $\{i, i + 1\}$, γ is well-defined.



■ **Figure 2** Gadget Ind_1 for $n = 6$ and $I = \{\{2, 3\}, \{4, 5\}\}$.

► **Definition 23** (Indicator gadget). Let $c \in \{1, 2\}$. An indicator gadget (for c, n and I) is a tuple $(\text{Ind}_c, (x_1, \dots, x_n), (y_1, \dots, y_N), L)$, where $(\text{Ind}_c, (x_1, \dots, x_n), (y_1, \dots, y_N))$ is a link and $L : V(\text{Ind}_c) \rightarrow 2^{[4]}$ is a list function, such that:

- (11) For every $i \in [n]$ and $j \in [N]$, it holds that $L(x_i) = L(y_j) = \{1, 2\}$.
- (12) Every $f : \{x_1, \dots, x_n\} \rightarrow \{1, 2\}$ can be extended to a coloring of the gadget so that the following holds. For every $\{i, i+1\} \in I$, if $\{f(x_i), f(x_{i+1})\} \neq \{c\}$, then $f(y_{\gamma(\{i, i+1\})}) \neq c$.
- (13) For every coloring f of the gadget, the following holds. For every $i \in [n]$, we have $f(x_i) = f(y_{\gamma(i)})$, and for every $\{i, i+1\} \in I$, if $f(x_i) = f(x_{i+1}) = c$, then $f(y_{\gamma(\{i, i+1\})}) = c$.

Note that each output vertex of the gadget corresponds either to an input vertex, or to a pair in I ; this correspondence is given by γ . The gadget plays two roles. First, the coloring of input vertices is copied to their corresponding output vertices. Second, for each pair in I , the color of its corresponding output vertex *indicates* if both input vertices representing this pair are colored c .

In the following lemma we show how to construct indicator gadgets; consult also Figure 2.

► **Lemma 24.** Let $c \in \{1, 2\}$, let $n \in \mathbb{N}$ and let $I \subseteq \binom{[n]}{2}$ be a set of pairwise disjoint pairs from of $[n]$, each of the form $\{i, i+1\}$. In time polynomial in n , we can construct an indicator gadget for c, n , and I .

Proof. By symmetry, assume that $c = 1$. Let γ be the function defined for n and I as before.

The vertex set of the gadget is partitioned into five sets V_1, \dots, V_5 , where, for all $i \in [4]$, all vertices of V_i precede all vertices from V_{i+1} . We will construct the sets one by one.

Set V_1 : We start with introducing the vertices v_1^1, \dots, v_n^1 (in that order), all with list $\{1, 2\}$.

Set V_2 : Next, we introduce vertices v_1^2, \dots, v_n^2 (in that order), all with list $\{1, 2\}$. Next, for every $\{i, i+1\} \in I$, we add vertices u_i^2, w_i^2 , with lists, respectively, $\{1, 3\}$ and $\{1, 4\}$. We position them in the order so that $v_i^2 \prec u_i^2 \prec w_i^2 \prec v_{i+1}^2$. For every $i \in [n]$, we add the edge $v_i^1 v_i^2$, and for every $\{i, i+1\} \in I$, we add edges $v_i^1 u_i^2$ and $v_{i+1}^1 w_i^2$.

Set V_3 : We introduce vertices v_1^3, \dots, v_n^3 (in that order), all with list $\{1, 2\}$. For every $\{i, i+1\} \in I$, we add a vertex z_i^3 , with list $\{1, 3, 4\}$. We insert z_i^3 between v_i^3 and v_{i+1}^3 . For every $i \in [n]$, we add the edge $v_i^2 v_i^3$, and for every $\{i, i+1\} \in I$, we add edges $u_i^2 z_i^3$ and $w_i^2 z_i^3$.

Set V_4 : We introduce vertices v_1^4, \dots, v_n^4 (in that order), all with list $\{1, 2\}$. For every $\{i, i+1\} \in I$, we add vertex z_i^4 with list $\{1, 2\}$. We insert z_i^4 between v_i^4 and v_{i+1}^4 . For every $i \in [n]$, we add the edge $v_i^3 v_i^4$, and for every $\{i, i+1\} \in I$, we add the edge $z_i^3 z_i^4$.

Set V_5 : We introduce vertices v_1^5, \dots, v_n^5 (in that order), all with list $\{1, 2\}$. For every $\{i, i+1\} \in I$, we add a vertex z_i^5 with list $\{1, 2\}$. We insert z_i^5 between v_i^5 and v_{i+1}^5 . For every $i \in [n]$, we add the edge $v_i^4 v_i^5$, and for every $\{i, i+1\} \in I$, we add the edge $z_i^4 z_i^5$.

This completes the construction of Ind_1 ; the input vertices are V_1 and the output vertices are V_5 . Note that, for each $i \in [n]$, the $\gamma(i)$ -th output vertex is v_i^5 , and for each $\{i, i+1\} \in I$, the $\gamma(\{i, i+1\})$ -th output vertex is z_i^5 . It can be verified that Ind_1 satisfies the desired properties (\spadesuit), which completes the proof. \blacktriangleleft

NOT-cc gadgets. The last type of a gadget is a NOT-cc gadget, for $c \in \{1, 2\}$.

Throughout this paragraph, let $N \in \mathbb{N}$ and let $I \subseteq \binom{[N]}{2}$ be a set of pairwise disjoint pairs from of $[N]$, each of the form $\{i, i+1\}$. Let $n = N - |I|$; note that $n > 0$. Let $\delta : [n] \rightarrow [N]$ be an injection defined as follows. The image of this function is the set of these i , for which $\{i, i+1\} \notin I$. Furthermore, we have $\delta(1) < \delta(2) < \dots < \delta(n)$. Note that δ is well-defined.

► **Definition 25 (NOT-cc).** Let $c \in \{1, 2\}$. An NOT-cc gadget (for N and I) is a tuple $(B_c, (x_1, \dots, x_N), (y_1, \dots, y_n), L)$, where $(B_c, (x_1, \dots, x_N), (y_1, \dots, y_n))$ is a link and $L : V(B_c) \rightarrow 2^{[4]}$ is a list function, such that:

- (B1) For every $i \in [N]$ and $j \in [n]$, it holds that $L(x_i) = L(y_j) = \{1, 2\}$.
- (B2) Every $f : \{x_1, \dots, x_N\} \rightarrow \{1, 2\}$ such that for every $\{i, i+1\} \in I$ it holds that $\{f(x_i), f(x_{i+1})\} \neq \{c\}$, can be extended to a coloring of the gadget.
- (B3) For every coloring f of the gadget, the following holds. For every $i \in [n]$, we have $f(y_i) = f(x_{\delta(i)})$, and for every $\{i, i+1\} \in I$ we have $\{f(x_i), f(x_{i+1})\} \neq \{c\}$.

Note that there are two types of input vertices: those x_i for which $\{i, i+1\} \notin I$, and the remaining ones. The former ones have corresponding output vertices; this correspondence is given by δ . The remaining input vertices do not have corresponding output vertices.

Again, the gadget plays two roles. First, it transfers the coloring of input vertices of the first type to their corresponding output vertices. Second, for each pair in I , the gadget ensures that input vertices corresponding to that pair are not both colored c .

The construction of NOT-cc gadgets is quite similar to the one for indicator gadget, so we omit here the proof of the following lemma.

► **Lemma 26 (\spadesuit).** Let $c \in \{1, 2\}$, let $N \in \mathbb{N}$ and let $I \subseteq \binom{[N]}{2}$ be a set of pairwise disjoint pairs from of $[N]$, each of the form $\{i, i+1\}$. Let $n = N - |I|$. In time polynomial in N , we can construct a NOT-cc gadget for I .

Constructing NOT-ccc gadgets. Finally, we can prove the following statement, which would finish the proof of Lemma 15 and thus, of Theorem 12.

► **Lemma 27 (\spadesuit).** Let $c \in \{1, 2\}$, let $n \in \mathbb{N}$ and let $I \subseteq \binom{[n]}{3}$ be a set of pairwise disjoint subsets of $[n]$, each of size 3. In time polynomial in n , we can construct a NOT-ccc gadget for I .

The proof of Lemma 27 can be found in the full version of the paper. Let us just present here the general idea of our approach. Let x_1, \dots, x_n be the input vertices of the gadget. The gadget is supposed to forbid some triples of vertices to be colored only with color 1. However, doing this directly seems difficult, while keeping \curvearrowright -freeness. Thus, we do it in two steps.

For each triple $\{i, j, k\} \in I$, we create a new vertex $x_{i,j}$, so that if both vertices x_i and x_j are colored c , then the color of $x_{i,j}$ is forced to be c , and otherwise $x_{i,j}$ can be colored with the other color in $\{1, 2\}$. This is precisely the behavior that can be forced by an indicator gadget. Then we make sure that $x_{i,j}$ and x_k (actually, a vertex whose color is equal to the color of x_k ; here we use the second role of an indicator) are not both colored c . This can be done using the NOT-cc gadget.

There is one last thing to do: recall that the indicator and the NOT-*cc* gadgets can only operate on pairs of consecutive vertices, and $\{i, j, k\}$ do not have to be consecutive. However, this can be easily obtained by reshuffling the vertices using appropriate permutation gadgets.

Summing up, the NOT-*cc* is obtained by chaining a permutation gadget, an indicator gadget, another permutation gadget, a NOT-*cc* gadget, and, finally, yet another permutation gadget, to restore the original ordering of vertices.

Now, Lemma 15 follows directly by combining Lemma 27 and Observation 17. This completes the proof of Theorem 12.

5 Conclusion

Let us conclude the paper with pointing out some possible directions for future research. An obvious one is to fill the gaps in Table 1. We believe obtaining a full dichotomy here is much easier than the analogous question for unordered graphs. In particular, we believe that understanding the complexity of LIST k -COLORING for $k \geq 3$ in \curvearrowright -free ordered graphs is a specific and intriguing problem.

Second, we believe it is interesting to investigate the complexity of the (non-list) k -COLORING problem in hereditary classes of ordered graphs. While algorithms for LIST k -COLORING clearly carry over to k -COLORING this is not the case for hardness reductions. Note that our proofs crucially use lists, and a standard way of simulating lists by adding a k -clique and using the edges to this clique to forbid certain colors introduces new induced subgraphs.

On the other hand, the reduction in Theorem 12 is *quadratic*, and thus, it only excludes algorithms with running time $2^{o(\sqrt{n})}$, where n is the number of vertices of the input. The bottleneck is the construction of a permutation gadget, which has $\Theta(n^2)$ vertices. It would be interesting to improve this construction to a linear one, or provide a subexponential-time algorithm for the problem.

References

- 1 Édouard Bonnet and Paweł Rzażewski. Optimality program in segment and string graphs. *Algorithmica*, 81(7):3047–3073, 2019. doi:10.1007/S00453-019-00568-7.
- 2 Flavia Bonomo, Maria Chudnovsky, Peter Maceli, Oliver Schaudt, Maya Stein, and Mingxian Zhong. Three-coloring and list three-coloring of graphs without induced paths on seven vertices. *Comb.*, 38(4):779–801, 2018. doi:10.1007/S00493-017-3553-8.
- 3 Maria Chudnovsky, Sepehr Hajebi, and Sophie Spirkl. List- k -Coloring H -Free Graphs for All $k > 4$. *Comb.*, 44(5):1063–1068, 2024. doi:10.1007/S00493-024-00106-2.
- 4 Maria Chudnovsky, Shenwei Huang, Sophie Spirkl, and Mingxian Zhong. List 3-coloring graphs with no induced $P_6 + rP_3$. *Algorithmica*, 83(1):216–251, 2021. doi:10.1007/S00453-020-00754-Y.
- 5 Jean-François Couturier, Petr A. Golovach, Dieter Kratsch, and Daniël Paulusma. List coloring in the absence of a linear forest. *Algorithmica*, 71(1):21–35, 2015. doi:10.1007/S00453-013-9777-0.
- 6 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
- 7 Andreas Darmann and Janosch Döcker. On simplified NP-complete variants of Not-All-Equal 3-Sat and 3-Sat. *CoRR*, abs/1908.04198, 2019. arXiv:1908.04198.
- 8 Keith Edwards. The complexity of colouring problems on dense graphs. *Theor. Comput. Sci.*, 43:337–343, 1986. doi:10.1016/0304-3975(86)90184-2.

- 9 Thomas Emden-Weinert, Stefan Hougardy, and Bernd Kreuter. Uniquely colourable graphs and the hardness of colouring graphs of large girth. *Comb. Probab. Comput.*, 7(4):375–386, 1998. URL: <http://journals.cambridge.org/action/displayAbstract?aid=46667>.
- 10 Esther Galby, Paloma T. Lima, Andrea Munaro, and Amir Nikabadi. Maximum list r -colorable induced subgraphs in kP_3 -free graphs. In Anne Benoit, Haim Kaplan, Sebastian Wild, and Grzegorz Herman, editors, *33rd Annual European Symposium on Algorithms, ESA 2025, Warsaw, Poland, September 15-17, 2025*, volume 351 of *LIPICs*, pages 40:1–40:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPICs.ESA.2025.40.
- 11 Petr A. Golovach, Daniël Paulusma, and Jian Song. Closing complexity gaps for coloring problems on H -free graphs. *Inf. Comput.*, 237:204–214, 2014. doi:10.1016/J.IC.2014.02.004.
- 12 Sepehr Hajebi, Yanjia Li, and Sophie Spirkl. Complexity dichotomy for list-5-coloring with a forbidden induced subgraph. *SIAM J. Discret. Math.*, 36(3):2004–2027, 2022. doi:10.1137/21M1443352.
- 13 Sepehr Hajebi, Yanjia Li, and Sophie Spirkl. List-3-Coloring ordered graphs with a forbidden induced subgraph. *SIAM J. Discret. Math.*, 38(1):1158–1190, 2024. doi:10.1137/22M1515768.
- 14 Chinh T. Hoàng, Marcin Kamiński, Vadim V. Lozin, Joe Sawada, and Xiao Shu. Deciding k -colorability of P_5 -free graphs in polynomial time. *Algorithmica*, 57(1):74–81, 2010. doi:10.1007/S00453-008-9197-8.
- 15 Ian Holyer. The NP-completeness of edge-coloring. *SIAM J. Comput.*, 10(4):718–720, 1981. doi:10.1137/0210055.
- 16 Shenwei Huang. Improved complexity results on k -coloring P_t -free graphs. *Eur. J. Comb.*, 51:336–346, 2016. doi:10.1016/J.EJC.2015.06.005.
- 17 Daniel Leven and Zvi Galil. NP completeness of finding the chromatic index of regular graphs. *J. Algorithms*, 4(1):35–44, 1983. doi:10.1016/0196-6774(83)90032-9.
- 18 Marcin Pilipczuk, Michał Pilipczuk, and Paweł Rzażewski. Quasi-polynomial-time algorithm for independent set in P_t -free graphs via shrinking the space of induced paths. In Hung Viet Le and Valerie King, editors, *4th Symposium on Simplicity in Algorithms, SOSA 2021, Virtual Conference, January 11-12, 2021*, pages 204–209. SIAM, 2021. doi:10.1137/1.9781611976496.23.
- 19 Frank Plumpton Ramsey. On a problem of formal logic. *Proceedings of the London Mathematical Society*, 30(1):264–286, 1930. doi:10.1112/plms/s2-30.1.264.