Maximum Weight Independent Set in Hereditary Classes of Ordered Graphs

Paweł Rafał Bieliński
Warsaw University of Technology, Poland

Marta Piecyk
CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Paweł Rzążewski
Warsaw University of Technology, Poland

Abstract

The complexity of classical computational problems in graph classes defined by forbidding induced subgraphs is one of the central topics of algorithmic graph theory. Recently, there has been a growing interest in the complexity of such problems in *ordered graphs*, i.e., graphs with a fixed linear ordering of vertices. Such an approach allows us to investigate the boundary of tractability more closely. However, most results so far concern coloring problems.

In this paper, we focus on the complexity of the Maximum Weight Independent Set (MWIS) problem in classes of ordered graphs. For every ordered graph H, we classify the complexity of MWIS in ordered graphs that exclude H as an induced subgraph into one of the following cases: (1) solvable in polynomial time, (2) solvable in quasipolynomial time, (3) solvable in subexponential time, and (4) NP-hard.

Notably, case (3) contains only one well-structured family of H obtained from two nested edges by adding isolated vertices in a specific way. Thus, our results yield an almost complete complexity dichotomy for MWIS in classes of ordered graphs defined by a single forbidden induced subgraph into cases solvable in quasipolynomial time and those that are NP-hard.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Mathematics of computing → Graph algorithms

Keywords and phrases Max Independent Set, ordered graphs, hereditary classes

Digital Object Identifier 10.4230/LIPIcs.ESA.2026.30

Related Version *Full Version*: `https://arxiv.org/abs/2604.24343` [8]

Funding *Paweł Rafał Bieliński*: Supported by the National Science Centre grant 2024/54/E/ST6/00094.
Paweł Rzążewski: Supported by the National Science Centre grant 2024/54/E/ST6/00094.

Acknowledgements Part of this work was done during the workshop Homonolo 2025. We are grateful to the organizers and other participants for the inspiring atmosphere and fruitful discussions.

1 Introduction

In the Max Independent Set (MIS) problem, we are given a graph G and we ask for a largest *independent set* in G, i.e., a largest set of pairwise nonadjacent vertices. The Max Weight Independent Set (MWIS) problem is a generalization of MIS, where we are given a graph G together with a weight function $\mathfrak{w} : V(G) \to \mathbb{Q}_{>0}$, and we ask for an independent set of maximum total weight. Both problems are among the most fundamental and well-studied problems in algorithmic graph theory. They are also known to be notoriously hard: MIS is among Karp's 21 NP-complete problems [33], cannot be solved in subexponential time under the Exponential-Time Hypothesis (ETH) [32], is a canonical W[1]-hard problem [17], and is hard to approximate [29], even in parameterized settings [12, 35].

A typical approach when dealing with such a hard problem is to consider restricted instances in order to understand the boundary of tractability. Usually, one considers instances from certain *hereditary* graph classes, i.e., classes of graphs closed under vertex deletion. Every such class can be characterized by a family of forbidden induced subgraphs. Thus, as a starting point, it is natural to ask for which graphs H the MWIS problem can be solved efficiently on *H-free graphs*, i.e., graphs that do not contain H as an induced subgraph.

The complexity study of MWIS in H-free graphs is one of the central topics in algorithmic graph theory. It turns out that the crucial role is played by the family of subcubic forests whose every component is a path or a *subdivided claw*, i.e., a tree with at most one vertex of degree 3. Let us denote by $\mathcal{S}$ the family of such forests.

It was already observed by Alekseev [3] that if $H \notin \mathcal{S}$, then MIS (and thus, MWIS) is NP-hard in H-free graphs. Indeed, this observation follows from combining two known results. First, MIS is NP-hard for subcubic graphs [21], and they in particular exclude any graph H with a vertex of degree at least 4. The second ingredient is the so-called "Poljak's subdivision trick" [43]: if we subdivide one edge of a graph G twice, i.e., we replace it by a four-vertex path, then the size of a largest independent set increases by exactly one. Thus, if H has a cycle or two vertices of degree 3 in a single component, then subdividing each edge of any instance of MIS sufficiently many times yields an equivalent instance that is H-free.

It turns out that polynomial-time algorithms for MWIS are only known for small graphs $H \in \mathcal{S}$, most notably, if H is a path on at most 6 vertices [27], a graph obtained from a three-leaf star by subdividing one edge once [37], or a graph whose every component is a three-leaf star [10]. On the other hand, we know no NP-hardness results for MWIS in H-free graphs with $H \in \mathcal{S}$ and the general belief is that the problem is polynomial-time solvable in all such cases. This belief is supported by the fact that for every $H \in \mathcal{S}$, the MWIS problem can be solved in quasipolynomial time when restricted to H-free graphs [22,23,42]. Note that this is a strong indication that the problem is not NP-hard in any of these cases, as otherwise every problem in NP would be solvable in quasipolynomial time, which is considered unlikely.

In this paper we follow the line of research described above, but we focus on hereditary classes of *ordered* graphs, i.e., graphs with a fixed linear order of their vertex set. We aim to understand for which ordered graphs H the MWIS problem can be solved efficiently in H-free ordered graphs (an ordered induced subgraph is an induced subgraph that preserves the relative order of its vertices). Note that the ordered setting can be seen as a refinement of the unordered one. Indeed, forbidding a single *unordered* graph is equivalent to forbidding *all* of its possible orderings as ordered induced subgraphs. Instead, we can forbid only some of the orderings, which allows us to trace the boundary of tractability more precisely.

Let us remark that the idea of looking at classical graph-theoretic problems in an ordered setting is not new. For example, there is a rich line of research concerning Turán-type [9,34,39,45] and Ramsey-type [6,16,19] problems in ordered graphs. There are also some results on the chromatic number [4,11] or structural properties [38,44] of certain classes of ordered graphs. However, the algorithmic results seem to be much scarcer: we are only aware of recent work on the complexity of (List) k-Coloring in classes defined by excluding one induced ordered subgraph [28,41]. In particular, to the best of our knowledge, the complexity of MWIS in this setting was not studied before. In this paper, we fill this gap.[1]

[1] The proofs of statements marked with (♠) are omitted due to the space limit. They can be found in the full version of the paper [8].

▶ **Theorem 1.1.** *Let H be an ordered graph with at least two vertices. The MWIS problem in ordered H-free graphs can be solved in:*

(1) (♠) *polynomial time, if H is an induced subgraph of a graph in* $\bigcup_{k\geq 0}\{ (k)\!\frown\!(k)\,,\ (k)\!\frown\!(k)\,,\ (k)\!\frown\!(k)\,,\ (k)\!\frown\!(k)\!\frown\!(k) \}$,

(2) *quasipolynomial time, if H is an induced subgraph of a graph in* $\bigcup_{k\geq 0}\{ (k)\!\frown\!(k)\!\frown\!(k)\,,\ (k)\!\frown\!(k)\!\frown\!(k)\!\frown\!(k)\!\frown\!(k) \}$,

(3) *subexponential time, if H is an induced subgraph of a graph in* $\bigcup_{k\geq 0}\{ (k)\!\frown\!(k)\!\frown\!(k)\!\frown\!(k) \}$.

(♠) *In all other cases, even the MIS problem is NP-hard.*

The notation used in Theorem 1.1 (and throughout the paper) should be self-explanatory. For example, $\frown$ is the ordered graph on vertices $1, 2, 3$ in natural order and edges 12 and 23. The only non-obvious feature is that by (k) we denote k consecutive isolated vertices, so, e.g., $(k)\!\frown\!(k)\!\frown\!(k)\!\frown\!(k)$ is the graph with vertices $1, \ldots, 4k+4$ in natural order, and edges $(k+1)(3k+4)$ and $(2k+1)(2k+2)$.

Let us make a few remarks concerning the statement of Theorem 1.1. First, notice that the case that $H = (k)\!\frown\!(k)\!\frown\!(k)\!\frown\!(k)$ for some k is the only one that we cannot classify either as quasipolynomial-time solvable or NP-hard. Thus, our Theorem 1.1 gives an almost complete complexity dichotomy into cases solvable in quasipolynomial time and those that are NP-hard. Still, $(k)\!\frown\!(k)\!\frown\!(k)\!\frown\!(k)$-free and, in particular, $\frown\!\frown$-free graphs admit some strong structural properties that allow us to solve MWIS in subexponential time.

Interestingly, $\frown\!\frown$-free graphs remain one of few open cases concerning the complexity of List 3-Coloring [28, 41], which is another indication that graphs from this class might be difficult to understand. Their structure does not seem restricted enough to allow for (simple) efficient algorithms, nor rich enough to allow for (simple) hardness results.

Another indication that $\frown\!\frown$-free graphs have complicated structure is the fact that unordered graphs that admit an ordering that avoids $\frown\!\frown$ as a subgraph are precisely graphs of *queue number* 1 [30]. Such graphs are NP-hard to recognize [31] and thus, there is little hope for their simple structural characterization. On the other hand, unordered graphs that admit an ordering that avoids $\frown$ as a subgraph are known as graphs of *stack number* 1 [7] and they are precisely outerplanar graphs – a rather simple class [13]. These two examples already illustrate that graphs defined by forbidding even a very simple ordered substructure might be quite complicated. Furthermore, the ordering of the vertices in a forbidden subgraph can have a huge impact on the structure of the graphs in the corresponding hereditary class.

Technical overview. It is not hard to observe that if H' is obtained from H by adding some isolated vertices at the beginning or at the end of the ordering, then the MWIS problem in H'-free graphs can be reduced to solving the same problem in the more restrictive class of H-free graphs. We note that the above observation does not apply to isolated vertices added in the middle of the ordering, as illustrated by, e.g., cases of $H = \frown$ and $H = \frown\!\frown$. Indeed, MWIS is polynomial-time solvable in $\frown$-free graphs, but NP-hard in $\frown\!\frown$-free graphs.

Thus, from now on let us focus on the case that H has no isolated vertices at the beginning or at the end of the ordering. In what follows, G is always the input graph on n vertices.

The cases listed in Theorem 1.1 (1) are actually quite simple. One can readily verify that $\frown$-free graphs are comparability graphs and $\frown\!\frown$-free and $\frown\!\frown$-free graphs are chordal – in both cases MWIS can be solved in polynomial time by classical algorithms [25, 26]. Interestingly, the above inclusions are in fact equivalences in the sense that every comparability graph admits an ordering that avoids $\frown$ and every chordal graph admits an ordering that avoids $\frown\!\frown$ (and the reverse ordering avoids $\frown\!\frown$). The last case from Theorem 1.1 (1), i.e., $H = (k)$ for fixed k, can be handled by a simple dynamic programming.

So, let us proceed to cases listed in Theorem 1.1 (2). Let us start with $H = \frown\ \frown$. We remark that later we will show an algorithm for the more general case $H = \frown_{(k)}\frown$ for any fixed k, but the algorithm for $H = \frown\ \frown$ is much simpler and has better running time.

Note that $\frown\ \frown$-free graphs contain all bipartite graphs: ordering vertices of any bipartite graph so that all vertices from one part come before all vertices from the other part yields an $\frown\ \frown$-free ordered graph. MWIS in bipartite graphs can be solved in polynomial time by a reduction to the maximum matching problem [36]. Therefore, when working with $\frown\ \frown$-free graphs, we should be aware that we are in particular solving the (unordered) bipartite case and thus, we should not hope for a simple dynamic programming or branching algorithm.

In fact, our approach is to start with an $\frown\ \frown$-free instance of MWIS and reduce it, in quasipolynomial time, to a quasipolynomial number of bipartite instances, each of which can be solved in polynomial time. A *seagull* is a triple of vertices $x \prec y \prec z$ such that y is adjacent to both x and z. We observe that graphs that do not contain any seagull are bipartite. Thus, we employ a branching scheme in order to destroy all seagulls.

First, we observe that every two seagulls in an $\frown\ \frown$-free graph share a vertex or are joined by an edge. Thus, if x, y, z is a seagull, then the closed neighborhood of one of x, y, z intersects at least one third of all seagulls in the graph. We branch on such a vertex v, i.e., we guess whether it is in the solution or not. In the former case we remove v and all its neighbors from the graph (which destroys a constant fraction of all seagulls) and in the latter one we remove just v (which destroys at least one seagull – the one formed by x, y, z). As the number of all seagulls is $\mathcal{O}(n^3)$, this branching scheme produces at most $n^{\mathcal{O}(\log n)}$ instances, each of which has no seagulls and thus, can be solved in polynomial time. Therefore, the overall running time of the algorithm is $n^{\mathcal{O}(\log n)}$.

Now let us discuss the more general case $H = \frown_{(k)}\frown$ for any fixed k. Here we also use a branching scheme, but it is more involved than in the previous case. The algorithm maintains a more structured instance, where the vertex set is partitioned into three consecutive parts X, Y, Z, such that X and Z are independent, while Y is the "workspace" that still requires handling. Initially, we take $X = Z = \emptyset$ and $Y = V(G)$. If $Y = \emptyset$, then the graph is bipartite and the problem can be solved in polynomial time, so the goal is to shrink Y until it is empty.

To this end, we split Y into two halves, Y_L and Y_R, and exhaustively guess the first k vertices of the optimum solution that lie in Y_R. If $X \cup Y_L$ or $Y_R \cup Z$, say the former one, is independent, then we can move Y_L to X, thereby reducing the size of the workspace by at least half. Thus, we may assume that both $X \cup Y_L$ and $Y_R \cup Z$ contain an edge; call such edges *internal*. Now the forbidden pattern $\frown_{(k)}\frown$ yields a key structural property: the closed neighborhood of every internal edge on one side of the split intersects every internal edge on the opposite side. Consequently, if both sides still contain an internal edge, then the neighborhood of one endpoint of any edge on the sparser side (i.e., one with fewer internal edges) intersects a constant fraction of the internal edges on the denser side.

This leads to an internal branching procedure, analogous in spirit to the $\frown\ \frown$-case: we guess whether such a vertex belongs to the solution or not. If we exclude it, we destroy at least one internal edge; if we include it, we delete its closed neighborhood and thus, destroy a constant fraction of all internal edges. We continue until one side of the split becomes independent; the number of instances produced this way is $n^{\mathcal{O}(\log n)}$. At that point, we absorb the independent side into X or Z, thereby obtaining a new structured instance whose workspace has size at most half of the previous one.

Repeating this halving procedure recursively yields quasipolynomially many subinstances, each eventually reduced to a bipartite graph. Thus, the recursion has two levels: the outer one halves the workspace, while the inner one destroys a constant fraction of internal edges. Altogether, this gives running time $n^{\mathcal{O}(\log^2 n)}$.

Now, let us consider the case $H = \overset{(k)}{\cdot}\overset{(k)}{\cdot}\overset{(k)}{\cdot}$ for any fixed k. For simplicity, let us discuss the case $k = 0$, i.e., $H = \cdot\cdot\cdot$ – we just remark that the general case is slightly more complicated. Here the general strategy is similar in spirit to the case of $\cdot\overset{(k)}{\frown}\cdot$, but the relevant structure of the considered instances is different (and the arguments are more involved). We start by partitioning the vertex set into three consecutive blocks A, B, C of roughly equal size and view them as a *chain*, that is, a sequence of consecutive sets (called *links*) in which edges may appear only inside one set, between two consecutive sets, or between the first and the last one. The algorithm repeatedly *refines* such a chain by splitting one selected link into two consecutive sublinks.

The main subroutine is a refinement lemma: for any chosen link, in quasipolynomial time we can branch into quasipolynomially many instances so that this link is replaced by two smaller consecutive links. Again the crucial structural observation is that, since $\cdot\cdot\cdot$ is forbidden, there is always a vertex whose closed neighborhood intersects a constant fraction of the "unwanted" edges. This again shows that in quasipolynomial time we can produce $n^{\mathcal{O}(\log n)}$ branches, in each of which the chosen link can indeed be split into two parts.

Starting from the initial chain (A, B, C), we repeatedly refine the smallest link of *type* A (i.e., one that originates from A), the smallest link of *type* B, and the smallest link of type C. In each refinement step the size of a smallest link of the given type is halved, so after $\mathcal{O}(\log n)$ refinements for each type, we arrive at a chain containing an empty link of each of the three types. These three empty links separate the graph into three pairwise nonadjacent subinstances, each missing one of sets A, B, C and therefore having size at most roughly $2n/3$. We solve these subinstances recursively and sum their optimum values.

Summarizing the above, the algorithm has three nested recursive layers. One application of the refinement lemma costs $n^{\mathcal{O}(\log n)}$ branches. Repeating refinements until an empty link of each type appears creates a tree of depth $\mathcal{O}(\log n)$ and hence the number of instances produced is $n^{\mathcal{O}(\log^2 n)}$. Finally, each leaf instance splits into three pairwise nonadjacent subinstances of size at most roughly $2n/3$ each, which contributes the third logarithmic factor in the exponent. Thus, the overall running time is $n^{\mathcal{O}(\log^3 n)}$.

Now, let us move to Theorem 1.1 (3) and discuss $H = \cdot\overset{(k)}{\frown}\cdot$ for any fixed k. Again, for simplicity, let us focus on the case $k = 0$, i.e., $H = \cdot\frown\cdot$. The first step is a standard branching on high-degree vertices [5]. Setting a threshold $\tau = n^{1/3}$, we exhaustively branch on every vertex of degree at least τ. This costs only subexponential time and from now on we may assume that every vertex has degree at most τ.

The next step is to partition the ordered vertex set into consecutive segments $Z_1, \ldots, Z_t$ so that each segment Z_i comes with an edge $x_i y_i$ spanning it, and edges between different segments may appear only between consecutive ones. Such a partition can be obtained by greedily picking, at each step, an edge leaving the current prefix whose right endpoint is as far to the right as possible. Thus, after this partition, the graph behaves like a path of segments. The forbidden pattern $\cdot\cdot\cdot$ implies that once the neighborhood of $\{x_i, y_i\}$ is removed, the rest of the segment Z_i becomes independent.

Now we look for a place where the instance can be split. If some small segment (i.e., of size at most $n^{2/3}$) appears "in the middle" of the graph, then by branching on its intersection with the optimum solution we separate the graph into two independent subinstances, each of size at most roughly $2n/3$, and recurse on both sides. Otherwise, every segment meeting the middle part is large, so there can be only few of them. In this case we branch on a set of size $\mathcal{O}(n^{2/3})$ containing the two "boundary" small segments together with the neighborhoods of all x_i, y_i for the middle segments Z_i. After removing this set, the middle part becomes bipartite, while the remaining left and right parts are again multiplicatively smaller and can

be solved recursively. Thus, every branch either is solved directly or reduces the problem to instances of size at most about $2n/3$, while paying only a subexponential branching cost per level. The overall running time is $n^{\mathcal{O}(n^{2/3}\log^3 n)}$.

Finally, let us say a few words about the hardness counterpart of Theorem 1.1. We show several hardness reductions, each of which is tailored to a specific family of forbidden ordered graphs H. Typically, we start with an unordered instance of MIS, then we use the "Poljak's subdivision trick" to subdivide certain edges (possibly many times), and then we order the resulting graph in a way that ensures that H is not an induced subgraph. It is possibly worth mentioning that in the vast majority of cases we can actually exclude certain graphs H as a *subgraph*, which is a stronger condition than excluding them as an induced subgraph.

2 Preliminaries

Let G be a graph with a fixed linear ordering of its vertex set. We write $v \prec u$ if v is before u in the ordering, and $v \preceq u$ if $v \prec u$ or $v = u$. For a set of vertices X, we write $X \prec v$ if $x \prec v$ for every $x \in X$, and $v \prec X$ if $v \prec x$ for every $x \in X$. We also write, e.g., $x \prec Y$ for $\{x\} \prec Y$. Analogously we define operators $\succ$ and $\succeq$. A *segment* is a set of consecutive vertices in the ordering. A *prefix* is a segment that contains the first vertex in the ordering.

By $N(v)$ we denote the neighborhood of v in G, and by $N[v]$ we denote the closed neighborhood of v in G, i.e., $N[v] = N(v) \cup \{v\}$. For a vertex-weighted graph $(G, \mathfrak{w})$, by $\alpha(G, \mathfrak{w})$ we denote the maximum weight of an independent set in G.

▶ **Lemma 2.1 (♠).** *Let H be an ordered graph and let $k \geq 1$ be a fixed integer. Let H^* be the graph obtained from H by adding k isolated vertices before all vertices of H, and another k isolated vertices after all vertices of H. Then, MWIS in H-free graphs and MWIS in H^*-free graphs are polynomially equivalent.*

3 Algorithms

In this section we present the algorithms for the positive cases of Theorem 1.1. By Lemma 2.1, we can assume that the first and the last vertex of H have positive degree. We will present the algorithms for the cases that H is  ,  , and  (for some fixed k), which correspond to statements (2) – simplified variant and (3) of Theorem 1.1. The proofs of the remaining cases can be found in the full version of the paper [8].

3.1 Excluding

In this section we prove the following result.

▶ **Theorem 3.1.** *MWIS in n-vertex -free graphs can be solved in time $n^{\mathcal{O}(\log n)}$.*

Let G be an ordered graph. A *seagull* is a triple (x, y, z) of distinct vertices of G such that $x \prec y \prec z$ and $xy, yz \in E(G)$. Thus, $G[\{x, y, z\}]$ is either a triangle or .

▶ **Lemma 3.2 (♠).** *An ordered graph with no seagulls is bipartite.*

▶ **Lemma 3.3 (♠).** *Let G be a -free ordered graph. Then, G has no two seagulls that are disjoint and nonadjacent.*

Combining Lemma 3.2 and Lemma 3.3 we obtain the following.

▶ **Corollary 3.4 (♠).** *Let G be a $\frown\frown$-free ordered graph that is not bipartite. Then there is $v \in V(G)$ that belongs to a seagull and such that $N[v]$ intersects one third of all seagulls.*

Now we are ready to present our algorithm.

Proof of Theorem 3.1. Let $(G, \mathfrak{w})$ be an instance of MWIS, where G has n vertices and is $\frown\frown$-free. Our algorithm is a recursive procedure. The base case is that the instance graph is bipartite. Then, we solve the problem in polynomial time using the Kőnig-Egerváry theorem and flow techniques [36]. Note that this in particular covers the case that $n \leq 2$.

Thus, assume that G is not bipartite. Let v be a vertex given by applying Corollary 3.4 on G. We branch on v, i.e., we call the algorithm recursively for the instances $G - v$ and $G - N[v]$. We return the maximum of the value found in the first branch, and the value found in the second branch plus $\mathfrak{w}(v)$.

The correctness of the algorithm is obvious. Let us argue about the running time. We will measure the potential of the instance by the number μ of seagulls. If $\mu = 0$, the graph is bipartite and thus the instance can be solved in polynomial time. The number of instances created by branching is given by the recursive inequality $F(\mu) \leq F(\mu - 1) + F(2\mu/3)$, which solves to $\mu^{\mathcal{O}(\log \mu)}$. Since $\mu \leq n^3$, overall running time of the algorithm is $n^{\mathcal{O}(\log n)}$. ◀

3.2 Excluding $(k)(k)(k)$

In this section we show the following result.

▶ **Theorem 3.5.** *For every fixed k, MWIS in n-vertex $(k)(k)(k)$-free graphs can be solved in time $n^{\mathcal{O}(\log^3 n)}$.*

We will work with instances with some additional structure. Let $(G, \mathfrak{w})$ be an instance of MWIS, where $G = (V, E)$ is an ordered $(k)(k)(k)$-free graph. A *chain* (over $(G, \mathfrak{w})$) is a sequence $\mathcal{X} = (X_1, \ldots, X_r)$ of $r \geq 3$ pairwise disjoint subsets of V, called *links*, such that $X_1 \prec X_2 \prec \ldots \prec X_r$ and edges might appear only inside one set of between two cyclically consecutive sets. We often identify chains with the unions of all links, i.e., we write $G[\mathcal{X}]$ for $G[\bigcup \mathcal{X}]$ and, for a set $Y \subseteq V(G)$, we write $\mathcal{X} - Y$ for $(X_1 \setminus Y, \ldots, X_r \setminus Y)$. Furthermore, we define $\alpha(\mathcal{X}) = \alpha(G[\mathcal{X}], \mathfrak{w})$. Note that if $\mathcal{X}$ is a chain, then every edge of $G[\mathcal{X}]$ is either local (i.e., contained in one link), or joins two cyclically consecutive links. We say that a chain $\mathcal{X}' = (X'_1, \ldots, X'_{r+1})$ is a *refinement* of a chain $\mathcal{X} = (X_1, \ldots, X_r)$ *at* $j \in [r]$ if

- for all $i < j$, we have $X'_i \subseteq X_i$,
- $X'_j, X'_{j+1} \subseteq X_j$, and
- for all $i > j$, we have $X'_{i+1} \subseteq X_i$.

In other words, $\mathcal{X}'$ was obtained by splitting X_j into two sets, and possibly removing some vertices. We emphasize that we never insist that links of a chain are non-empty. Actually, our approach is to start with a partition of V into three segments (note that this is a chain), and keep refining it until we obtain some empty links and thus, we disconnect the graph.

The following lemma encapsulates the main technical step of the algorithm that in quasipolynomial time, one can obtain a refinement of a given chain at any given link.

▶ **Lemma 3.6.** *Let $k \geq 1$ be an integer and let $(G, \mathfrak{w})$ be an instance of MWIS such that G is an n-vertex $(k)(k)(k)$-free ordered graph. Let $\mathcal{X} = (X_1, \ldots, X_r)$ be a chain over $(G, \mathfrak{w})$ and let $j \in [r]$. In time $n^{\mathcal{O}(\log n)}$, we can construct a family $\mathcal{Z}$ of $n^{\mathcal{O}(\log n)}$ pairs $(\mathcal{Z}, \ell)$, where $\mathcal{Z}$ is a refinement of $\mathcal{X}$ at j and $\ell \in \mathbb{Q}_{\geq 0}$, such that $\alpha(\mathcal{X}) = \max_{(\mathcal{Z},\ell)\in\mathcal{Z}} \alpha(\mathcal{Z}) + \ell$.*

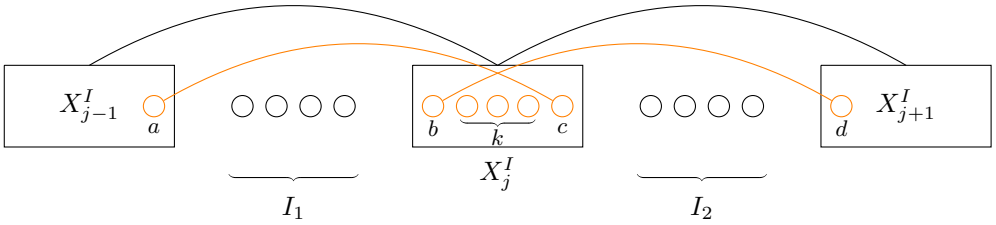


Figure 1 Sets X^I_{j-1}, X^I_j, X^I_{j+1}, and independent sets I_1, I_2 (each of size k) in Claim 3.7. The orange subgraph cannot be induced, as otherwise, together with I_1 and I_2, it creates $(k)(k)(k)$.

In the proof of Lemma 3.6 we distinguish two cases: (1) $1 < j < r$, (2) $j \in \{1, r\}$. The general outline of the argument in both cases is the same, but the details differ. We present the proof for case (1), and the proof for case (2) can be found in the full version of the pape [8].

Proof of Lemma 3.6 for $1 < j < r$. The proof has two main steps.

Construction of $\mathcal{Y}$. Let us first construct an auxiliary family $\mathcal{Y}$, which corresponds to guessing k first and k last vertices of the solution in X_j. More precisely, for each independent set $I \subseteq X_j$ of size at most $2k$, we construct a chain $\mathcal{X}^I$ as follows. Let I_1 be the set of the first k vertices of I, or $I_1 = I$ if $|I| < k$. Let $I_2 = I \setminus I_1$. Let v^*_1 be the last vertex of I_1 (if it exists) and let v^*_2 be the first vertex of I_2 (if it exists). Then, $\mathcal{X}^I$ is obtained from $\mathcal{X}$ by

(i) removing the vertices of $N[I]$, and

(ii) if $|I| = 2k$, then removing $\{x \in X_j \mid x \prec v^*_1 \vee x \succ v^*_2\}$, and otherwise removing all X_j.

That completes the construction of $\mathcal{X}^I$. We emphasize that the links of $\mathcal{X}^I$ are in one-to-one correspondence with the links of $\mathcal{X}$.

We add to $\mathcal{Y}$ the pair $(\mathcal{X}^I, \mathfrak{w}(I))$. Note that $|\mathcal{Y}| = \mathcal{O}(n^{2k})$. Furthermore, as each pair in $\mathcal{Y}$ corresponds to exhaustively guessing (at most) k first and (at most) k last vertices of the solution that are in X_j, it clearly holds that $\alpha(\mathcal{X}) = \max_{(\mathcal{Y},\ell)\in\mathcal{Y}} \alpha(\mathcal{Y}) + \ell$. Now let us show the crucial structural property of elements in $\mathcal{Y}$, see Figure 1.

▷ Claim 3.7 (♠). Let $(\mathcal{X}^I, \ell^I)$, where $\mathcal{X}^I = (X^I_1, \ldots, X^I_r)$, be a pair in $\mathcal{Y}$ constructed for an independent set I. There is no induced copy of (k) with edges ac, bd such that $a \in X^I_{j-1}$ and $b, c \in X^I_j$, and $d \in X^I_{j+1}$.

Construction of $\mathcal{Z}_\mathcal{Y}$. Fix some $(\mathcal{Y}, \ell) \in \mathcal{Y}$. Now in time $n^{\mathcal{O}(\log n)}$ we will construct a family $\mathcal{Z}_\mathcal{Y}$ of pairs $(\mathcal{Z}, \ell')$, where $\mathcal{Z}$ is a refinement of $\mathcal{Y}$ (and thus of $\mathcal{X}$) at j and $\ell' \in \mathbb{Q}_{\geq 0}$, such that $\alpha(\mathcal{Y}) = \max_{(\mathcal{Z},\ell')\in\mathcal{Z}_\mathcal{Y}} \alpha(\mathcal{Z}) + \ell'$. Let us point out that proving this would conclude the proof of the lemma. Indeed, the family $\mathcal{Z} := \bigcup_{(\mathcal{Y},\ell)\in\mathcal{Y}} \bigcup_{(\mathcal{Z},\ell')\in\mathcal{Z}_\mathcal{Y}} (\mathcal{Z}, \ell + \ell')$ satisfies all required properties, and the overall running time and the size of $\mathcal{Z}$ are both bounded by $n^{\mathcal{O}(\log n)}$.

Thus, let us show how to construct $\mathcal{Z}_\mathcal{Y}$ for $(\mathcal{Y}, \ell) \in \mathcal{Y}$. The process can be again described as a recursion tree T. We start with the root node which is labeled with the pair $(\mathcal{Y}, 0)$. Next, we proceed exhaustively as follows. Let d be a node of T that was not processed yet, and let it be labeled with $(\mathcal{Y}', \ell')$, where $\mathcal{Y}' = (Y'_1, \ldots, Y'_r)$. If there are no edges from Y'_{j-1} to Y'_j or from Y'_j to Y'_{j+1}, then we stop the recursion, i.e., d will be a leaf of T.

Otherwise, let u^* be the first vertex of Y'_j that is adjacent to some vertex in Y'_{j+1}, and let v^* be the last vertex of Y'_j that is adjacent to some vertex in Y'_{j-1}. If $v^* \prec u^*$, then we again stop the recursion and d becomes a leaf.

So now assume that $u^* \preceq v^*$, we emphasize that it is possible that $u^* = v^*$. Let S be the set of vertices $y \in Y'_j$ such that $u^* \preceq y \preceq v^*$, and let E_1 (resp., E_2) be the set of edges with one endpoint in Y'_{j-1} (resp., Y'_{j+1}), and the other in S.

▷ Claim 3.8 (♠). There exists a vertex w which is an endpoint of some edge in $E_1 \cup E_2$ and such that the set $N[w]$ intersects at least $\frac{1}{2k+4}$-fraction of the edges of $E_1 \cup E_2$.

Let us remark that the vertex w from Claim 3.8 can be found in polynomial time. Now we are ready to define child nodes for d; the nodes will correspond to taking w to the solution or not, i.e., we add two child nodes of d that are labeled respectively with: $(\mathcal{Y}' - N[w], \ell' + \mathfrak{w}(w))$ and $(\mathcal{Y}' - \{w\}, \ell')$. Clearly, the introduced pairs satisfy the following: $\alpha(\mathcal{Y}') = \max\big(\alpha(\mathcal{Y}' - N[w]) + \mathfrak{w}(w), \alpha(\mathcal{Y}' - \{w\})\big)$. This concludes the construction of T.

Now let us bound the number of leaves. Let $\mu = |E_1| + |E_2|$. We point out that the vertices u^* and v^* might change, but always the sets S, E_1, E_2 are subsets of the corresponding sets for the parent node. By Claim 3.8, in the branch corresponding to taking w to the solution, the number of edges in $E_1 \cup E_2$ decreases at least by $\frac{\mu}{2k+4}$, and in the branch corresponding to not taking w to the solution, we remove w which is an endpoint of at least one edge of $E_1 \cup E_2$. Therefore, the bound on the number of leaves in T is given by the recursive inequality: $F(\mu) \leq F\left(\frac{2k+3}{2k+4}\,\mu\right) + F(\mu - 1)$, which solves to $\mu^{\mathcal{O}(\log \mu)}$ (recall that k is a constant). Applying $\mu \leq n^2$, the number of leaves in T is bounded by $(n^2)^{\mathcal{O}(\log n^2)} = n^{\mathcal{O}(\log n)}$.

Now we are ready to define the family $\mathcal{Z}_{\mathcal{Y}}$. Consider a leaf node of T, let it be labeled with $(\mathcal{Y}', \ell')$, and let $\mathcal{Y}' = (Y'_1, \ldots, Y'_r)$. Then one of the following holds:

- **(i)** there are no edges between Y'_{j-1} and Y'_j,
- **(ii)** there are no edges between Y'_j and Y'_{j+1},
- **(iii)** for the first vertex $u^* \in Y'_j$ that is adjacent to some vertex in Y'_{j+1}, and for the last vertex $v^* \in Y'_j$ that is adjacent to some vertex in Y'_{j-1}, it holds that $v^* \prec u^*$.

In each of these cases, we add to $\mathcal{Z}_{\mathcal{Y}}$ the pair $(\mathcal{Z}, \ell')$, where $\mathcal{Z}$ is defined as follows, depending on the case:

- **(i)** $\mathcal{Z} = (Y'_1, \ldots, Y'_{j-1}, \emptyset, Y'_j, Y'_{j+1}, \ldots, Y'_r)$,
- **(ii)** $\mathcal{Z} = (Y'_1, \ldots, Y'_{j-1}, Y'_j, \emptyset, Y'_{j+1}, \ldots, Y'_r)$,
- **(iii)** $\mathcal{Z} = (Y'_1, \ldots, Y'_{j-1}, Y'_{j,\leftarrow}, Y'_{j,\rightarrow}, Y'_{j+1}, \ldots, Y'_r)$, where $Y'_{j,\leftarrow} = \{x \in Y'_j \mid x \preceq v^*\}$ and $Y'_{j,\rightarrow} = Y'_j \setminus Y'_{j,\leftarrow}$.

We repeat this for every leaf node of T. It is straightforward to verify that $\mathcal{Z}_{\mathcal{Y}}$ satisfies all required properties. This completes the proof of the lemma in the case if $1 < j < r$. ◀

Equipped with Lemma 3.6, we are ready to prove Theorem 3.5.

Proof of Theorem 3.5. Without loss of generality assume that $k \geq 1$. Let $(G, \mathfrak{w})$ be an n-vertex instance of MWIS such that $G = (V, E)$ is a $\overbrace{(k)(k)(k)}$-free ordered graph. The algorithm is recursive, with the trivial base case that $n \leq 2$. So assume that $n \geq 3$ and let $V = \{v_1, \ldots, v_n\}$ where $v_1 \prec v_2 \prec \ldots \prec v_n$. Define:

$$A = \{v_1, \ldots, v_{\lceil \frac{n}{3} \rceil}\} \quad B = \{v_{\lceil \frac{n}{3} \rceil + 1}, \ldots, v_{\lceil \frac{2n}{3} \rceil}\} \quad C = \{v_{\lceil \frac{2n}{3} \rceil + 1}, \ldots, v_n\}.$$

Note that $\mathcal{Q} = (A, B, C)$ is a chain over $(G, \mathfrak{w})$ and each of its links is of size at most $\lceil \frac{n}{3} \rceil$.

We will keep refining $\mathcal{Q}$, obtaining new chains. Note that each link of such a chain originates from one of A, B, or C. We will call such links, respectively, of type A, B, or C.

We recursively construct a labeled rooted tree T as follows. We start with a root labeled with $(\mathcal{Q}, 0)$. Now, consider a node d of T that was not processed yet. Let d be labeled with $(\mathcal{Q}', \ell')$, where $\mathcal{Q}' = (Q'_1, \ldots, Q'_r)$ is a chain and $\ell \in \mathbb{Q}_{\geq 0}$.

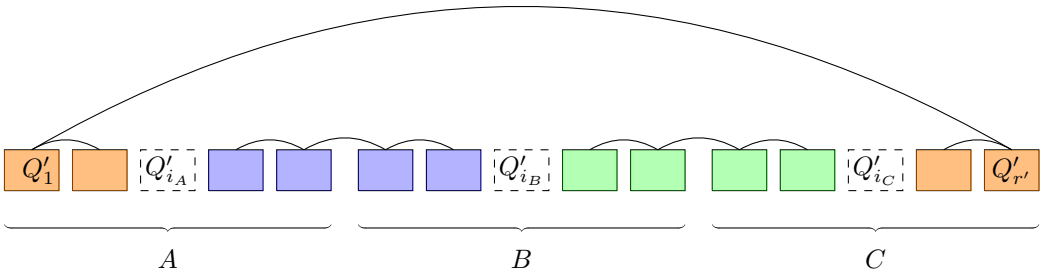


Figure 2 Chain $Q' = (Q'_1, \ldots, Q'_r)$ in the proof of Theorem 3.5. Dashed blocks denote empty sets. Blocks in orange, blue, and green denote, respectively, the sets that belong to V_1, V_2, and V_3.

For $\lambda \in \{A, B, C\}$, let i_λ be an index of a smallest link of type λ, where ties are resolved arbitrarily. If $|Q'_{i_A}| = |Q'_{i_B}| = |Q'_{i_C}| = 0$, we stop the recursion and d becomes a leaf of T. So suppose there is some $\lambda \in \{A, B, C\}$ such that $Q'_{i_\lambda} \neq \emptyset$. We call Lemma 3.6 for $\mathcal{Q}'$ and $j = i_\lambda$, obtaining the family $\mathcal{Z}$. For each member $(\mathcal{Z}, \ell)$ of this family, we create a child node for d and label it with $(\mathcal{Z}, \ell)$. This concludes the construction of T.

Let us bound the number of leaves in the tree. First note that each node of T has at most $n^{\mathcal{O}(\log n)}$ children. Furthermore, note that if d' is a child node of d, obtained by refining the chain at i_λ for some $\lambda \in \{A, B, C\}$, then the smallest link of type λ in the instance corresponding to d' has size at most half of the size of the smallest link of type λ in the instance corresponding to d. This means that on each root-to-leaf path in T, the same value of λ is chosen at most $\log n$ times, and thus, the depth of T is at most $3 \log n$. Consequently, the number of leaves of T is at most $n^{\mathcal{O}(\log^2 n)}$.

Finally, by Lemma 3.6, we can compute $\alpha(G, \mathfrak{w}) = \alpha(\mathcal{Q})$ by solving the instances corresponding to leaves of T.

So let us focus on analyzing the structure of such an instance $\mathcal{Q}' = (Q'_1, \ldots, Q'_{r'})$. Recall that since we are at a leaf of T, for each $\lambda \in \{A, B, C\}$ there is i_λ such that Q'_{i_λ} is of type λ and $Q'_{i_\lambda} = \emptyset$. Clearly $i_A < i_B < i_C$. Let us partition $\bigcup \mathcal{Q}'$ into three sets (see Figure 2):

$$V_1 = Q'_1 \cup \ldots \cup Q'_{i_A - 1} \cup Q'_{i_C + 1} \ldots \cup Q'_{r'}$$
$$V_2 = Q'_{i_A + 1} \cup \ldots \cup Q'_{i_B - 1}$$
$$V_3 = Q'_{i_B + 1} \cup \ldots \cup Q'_{i_C - 1}.$$

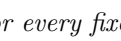
Note that since $\mathcal{Q}'$ is a chain, these three sets are pairwise nonadjacent. Thus, we obtain

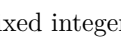
$$\alpha(\mathcal{Q}') = \alpha(G[V_1], \mathfrak{w}) + \alpha(G[V_2], \mathfrak{w}) + \alpha(G[V_3], \mathfrak{w}),$$

and each of the summands can be computed independently by calling the algorithm recursively the corresponding induced subgraph of G. Furthermore, note that V_1 is disjoint with B, V_2 is disjoint with C, and V_3 is disjoint with A. Consequently, each of these sets is of size at most $2\lceil n/3 \rceil$. This means that the depth of the recursion is bounded by $\mathcal{O}(\log n)$. Summing up, the overall running time is upper-bounded by $n^{\mathcal{O}(\log^3 n)}$. This completes the proof. ◀

3.3 Excluding $(k) \curvearrowright (k)$

In this section we prove Theorem 1.1 (3): for every fixed k, there is a *subexponential-time* algorithm for MWIS in $(k) \curvearrowright (k)$-free graphs. More precisely, we show the following result.

▶ **Theorem 3.9.** *For every fixed k, MWIS in n-vertex*  *-free graphs can be solved in time $n^{\mathcal{O}(n^{2/3}\cdot\log^3 n)}$.*

Proof. Let k be a fixed integer and let $(G, \mathfrak{w})$ be an instance of MWIS, where G is  -free and has n vertices. Note that we do not update the value of n throughout the run of the algorithm. Let $\tau = n^{1/3}$.

Step 1. Branching on high-degree vertices. First, we perform a standard branching on high-degree vertices. If there is a vertex v of degree at least τ, we branch on including v into the solution or not. The number of instances created in this phase is given by the recursion $F(n) \leq F(n-1) + F(n-\tau)$, which is solved by $n^{\mathcal{O}(n/\tau)} = 2^{\mathcal{O}(n^{2/3}\log n)}$. Each instance $(G', \mathfrak{w})$ created in this step has maximum degree at most τ.

Step 2. Partitioning $V(G')$ into segments. Consider one instance $(G', \mathfrak{w})$ created in Step 1 and let $n' = |V(G')|$. If $n' = 1$, the problem is obvious, so let us assume that $n' \geq 2$. We can also assume that G' is connected, as otherwise we can solve each component of G' separately.

In the next claim we will define some sequence $Z_1, \ldots, Z_t$ of subsets of $V(G')$. For $i \in [t]$, we introduce a notation $Z_{\leq i} := \bigcup_{i' \leq i} Z_{i'}$. In analogous way we define $Z_{<i}$ and $Z_{>i}$.

▷ Claim 3.10 (♠). There is a sequence $(x_i, y_i, Z_i)_{i=1}^t$, for some $t \geq 1$, such that each $x_i, y_i \in V(G')$ and $Z_i \subseteq V(G')$, that satisfies the following properties.

- **(P1)** for each $i \in [t]$, the set $Z_{\leq i}$ is a prefix of $V(G')$;
- **(P2)** for each $i \in [t]$, we have $x_i y_i \in E(G')$,
- **(P3)** for each $i \in [t]$, it holds that $x_i \preceq Z_i \preceq y_i$ and $y_i \in Z_i$,
- **(P4)** for each $i \in [t]$, the only vertices of $\bigcup_{i' \leq i} Z_{i'}$ that have edges to $V(G') \setminus Z_{\leq i}$ are in Z_i,
- **(P5)** sets $Z_1, Z_2, \ldots, Z_t$ form a partition of $V(G')$.

Let $(x_i, y_i, Z_i)_{i=1}^t$ be as in Claim 3.10. Note that property (P4) implies that edges between sets Z_i, Z_j, for $i < j$, might exist only if $j = i + 1$. We also denote $\mathsf{Roof} = \bigcup_{i=1}^t \{x_i, y_i\}$.

Step 3. Picking independent sets inside Z_i. For each $i \in [t]$, we proceed as follows. We greedily construct a set First_i: processing vertices from left to right, we add to First_i the first vertex in $Z_i \setminus N[\mathsf{Roof} \cup \mathsf{First}_i]$. We finish once $|\mathsf{First}_i| = k$ or when there are no more vertices in $Z_i \setminus N[\mathsf{Roof} \cup \mathsf{First}_i]$. Next, we construct a set Last_i: processing vertices from right to left, we add to Last_i the first vertex in $Z_i \setminus N[\mathsf{Roof} \cup \mathsf{First}_i \cup \mathsf{Last}_i]$. Again, we stop once we pick k vertices or there are no more possible vertices to choose. Summing up, $\mathsf{First}_i \cup \mathsf{Last}_i$ is an independent set of size at most $2k$, and there are no edges between this set and $\{x_i, y_i\}$.

▷ Claim 3.11 (♠). For each $i \in [t]$, the set $Z_i \setminus N[\{x_i, y_i\} \cup \mathsf{First}_i \cup \mathsf{Last}_i]$ is independent.

Step 4. Some more branching to split the instance. For $i \in [t]$, we say that Z_i is *large* if it has at least $n'/\tau \leq n^{2/3}$ vertices; otherwise Z_i is *small*. Let Left (resp., Right) consist of the first (resp., last) $\lfloor n'/3 \rfloor$ vertices of G', and let $\mathsf{Middle} = V(G') \setminus (\mathsf{Left} \cup \mathsf{Right})$.

Case 1: There is a small set Z_i intersecting Middle. Recall that there are no edges between $Z_{<i}$ and $Z_{>i}$. Furthermore $|Z_{<i}| \leq \frac{2}{3}n'$ and $|Z_{>i}| \leq \frac{2}{3}n'$.

We exhaustively guess the intersection of a fixed optimum solution with Z_i; in each branch we remove Z_i and all neighbors of vertices that are chosen to be in the solution.

This breaks the instance into two independent subinstances, one contained in $Z_{<i}$ and another contained in $Z_{>i}$, each of size at most $\frac{2}{3}n'$. Thus, on each of these subinstances, we can call the algorithm recursively.

The running time bound in this case is given by recursive inequality

$$T(n') \leq 2^{\mathcal{O}(n'/\tau)}T(2n'/3), \text{ which is solved by } T(n') \leq 2^{\mathcal{O}(n' \log n'/\tau)} = 2^{\mathcal{O}(n^{2/3}\log n)}.$$

Case 2: All small sets (if any) are contained in Left $\cup$ Right. Let ℓ be the largest value of i such that Z_i is small and $Z_i \subseteq \mathsf{Left}$, if such a value exists. Similarly, r be the smallest value of i such that Z_i is small and $Z_i \subseteq \mathsf{Right}$, if such a value exists. In what follows, let us assume that both ℓ and r as defined; the other cases are analogous (and slightly simpler).

Note that removing Z_ℓ and Z_r splits the graph into three parts: $Z_{<\ell}$, $\bigcup_{i=\ell+1}^{r-1} Z_i$, and $Z_{>r}$. Again, there are no edges between distinct parts.

The first and the last part are multiplicatively smaller than the original instance – each has at most $n'/3$ vertices. On the other hand, the middle part can be arbitrarily large. In fact, it might contain almost all vertices of G'. However, by the choice of ℓ and r, all sets Z_i that end up in the middle part are large. Let $t' = (r-1)-(\ell+1)+1$, i.e., the number of sets Z_i that belong to the middle part. Since $n' \geq \sum_{i=\ell+1}^{r-1} |Z_i| \geq t'n'/\tau$, we conclude that $t' \leq \tau$.

Let $S = Z_\ell \cup Z_r \cup N_{G'}\left[\bigcup_{i=\ell+1}^{r-1} (\{x_i, y_i\} \cup \mathsf{First}_i \cup \mathsf{Last}_i)\right]$. Note that $|S| \leq 2 \cdot n/\tau + (2+2k)\tau \cdot \tau = \mathcal{O}(n^{2/3})$. Again, we exhaustively guess the intersection of a fixed optimum solution with S. This results in at most $2^{|S|} = 2^{\mathcal{O}(n^{2/3})}$ branches, in each of which we remove S and the neighbors of vertices chosen to be included in the solution.

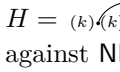
Now, we obtain three instances that can be solved independently: two of them – one contained in Left and the other contained in Right – are small, i.e., have at most $2n'/3$ vertices. Let G'' be the middle instance. By Claim 3.11 and (P4), G'' consists of independent sets, where only possible edges between distinct sets might appear if sets are consecutive. This means that G'' is bipartite, and thus MWIS on G'' can be solved in polynomial time.

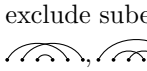
Summing up, the running time bound in this case is given by recursive inequality

$$T(n') \leq 2^{\mathcal{O}(n^{2/3})}\left(2\,T(2n'/3) + n'^{\mathcal{O}(1)}\right) \leq 2^{\mathcal{O}(n^{2/3})}T(2/3 \cdot n'),$$

which is solved by $T(n') \leq 2^{\mathcal{O}(n^{2/3}\log n')} = 2^{\mathcal{O}(n^{2/3}\log n)}$. This completes the proof. ◀

4 Conclusion

Let us conclude the paper by pointing out some directions for future research. Note that the hardness part of Theorem 1.1 is close to a dichotomy into cases solvable in quasipolynomial time and those that are NP-hard. Indeed, the (quasi)polynomial-time algorithms given by Theorem 1.1 (1) and (2) are a strong indication that the corresponding cases of MWIS in H-free ordered graphs are not NP-hard, since otherwise every problem in NP would admit a (quasi)polynomial-time algorithm. By contrast, the subexponential-time algorithm for $H = $  (for $k \geq 0$) from Theorem 1.1 (3) does not provide comparable evidence against NP-hardness. Thus, it would be very interesting to decide whether MWIS can be solved in (quasi)polynomial time in  -free ordered graphs, or whether the problem is NP-hard in this case. We conjecture that the algorithmic side holds.

Another natural direction is to strengthen the hardness side to ETH-lower bounds that exclude subexponential-time algorithms. At present, the reductions for $H \in \{$  $\}$ introduce a super-linear blow-up in the size of the constructed instance. It would be interesting either to replace them with tighter reductions or to design subexponential-time algorithms for those cases. We conjecture that the former is possible.

Finally, we believe that the ordered setting is promising also for problems beyond independent set and coloring. To the best of our knowledge, apart from the current paper and recent work on (List) k-Coloring [28, 41] such algorithmic questions in hereditary classes of ordered graphs have not been considered so far. In particular, it would be interesting to investigate Feedback Vertex Set and Odd Cycle Transversal in hereditary classes of ordered graphs. Both problems can be seen as generalizations of MWIS. Feedback Vertex Set can be equivalently stated as finding a maximum size (or weight) induced forest, i.e., an induced subgraph of treewidth at most 1 (while an independent set is an induced subgraph of treewidth 0). Odd Cycle Transversal is equivalent to finding a maximum size (or weight) induced bipartite subgraph, i.e., an induced subgraph of chromatic number at most 2 (while an independent set is an induced subgraph of chromatic number 1). Both problems are among the best studied in hereditary classes of unordered graphs [1, 2, 14, 15, 18, 20, 24, 40], so they seem like natural next targets in the ordered setting.

References

1 Tara Abrishami, Maria Chudnovsky, Marcin Pilipczuk, Paweł Rzążewski, and Paul D. Seymour. Induced subgraphs of bounded treewidth and the container method. *SIAM J. Comput.*, 53(3):624–647, 2024. doi:10.1137/20M1383732.

2 Akanksha Agrawal, Paloma T. Lima, Daniel Lokshtanov, Paweł Rzążewski, Saket Saurabh, and Roohani Sharma. Odd Cycle Transversal on P_5-free graphs in polynomial time. *ACM Trans. Algorithms*, 21(2):16:1–16:14, 2025. doi:10.1145/3708544.

3 Vladimir E. Alekseev. The effect of local constraints on the complexity of determination of the graph independence number. *Combinatorial-algebraic methods in applied mathematics*, pages 3–13, 1982.

4 Maria Axenovich, Jonathan Rollin, and Torsten Ueckerdt. Chromatic number of ordered graphs with forbidden ordered subgraphs. *Comb.*, 38(5):1021–1043, 2018. doi:10.1007/S00493-017-3593-0.

5 Gábor Bacsó, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Zsolt Tuza, and Erik Jan van Leeuwen. Subexponential-time algorithms for maximum independent set in P_t-free and broom-free graphs. *Algorithmica*, 81(2):421–438, 2019. doi:10.1007/S00453-018-0479-5.

6 Martin Balko, Josef Cibulka, Karel Král, and Jan Kyncl. Ramsey numbers of ordered graphs. *Electron. J. Comb.*, 27(1):1, 2020. doi:10.37236/7816.

7 Frank R. Bernhart and Paul C. Kainen. The book thickness of a graph. *Journal of Combinatorial Theory, Series B*, 27(3):320–331, 1979. doi:10.1016/0095-8956(79)90021-2.

8 Paweł Rafal Bieliński, Marta Piecyk, and Paweł Rzążewski. Maximum Weight Independent Set in hereditary classes of ordered graphs. *CoRR*, abs/2604.24343, 2026. doi:10.48550/arXiv.2604.24343.

9 Vladimir Bošković and Balázs Keszegh. Saturation of ordered graphs. *SIAM J. Discret. Math.*, 37(2):1118–1141, 2023. doi:10.1137/22M1485735.

10 Andreas Brandstädt and Raffaele Mosca. Maximum weight independent set for ℓclaw-free graphs in polynomial time. *Discret. Appl. Math.*, 237:57–64, 2018. doi:10.1016/J.DAM.2017.11.029.

11 Marcin Briański, James Davies, and Bartosz Walczak. Coloring ordered graphs with excluded induced ordered matchings. Banff International Research Station. 22w5009: Extremal Combinatorics and Geometry. https://www.birs.ca/events/2022/5-day-workshops/22w5009/videos/watch/202208161436-Walczak.html, 2022.

12 Parinya Chalermsook, Marek Cygan, Guy Kortsarz, Bundit Laekhanukit, Pasin Manurangsi, Danupon Nanongkai, and Luca Trevisan. From Gap-Exponential Time Hypothesis to fixed parameter tractable inapproximability: Clique, dominating set, and more. *SIAM J. Comput.*, 49(4):772–810, 2020. doi:10.1137/18M1166869.

13 Gary Chartrand and Frank Harary. Planar permutation graphs. *Annales de l'institut Henri Poincaré*, 6(4):433–438, 1967.

14 Maria Chudnovsky, Jason King, Michal Pilipczuk, Paweł Rzążewski, and Sophie Spirkl. Finding large H-colorable subgraphs in hereditary graph classes. *SIAM J. Discret. Math.*, 35(4):2357–2386, 2021. doi:10.1137/20M1367660.

15 Maria Chudnovsky, Rose McCarty, Marcin Pilipczuk, Michał Pilipczuk, and Paweł Rzążewski. Sparse induced subgraphs in P_6-free graphs. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 5291–5299. SIAM, 2024. doi:10.1137/1.9781611977912.190.

16 David Conlon, Jacob Fox, Choongbum Lee, and Benny Sudakov. Ordered Ramsey numbers. *J. Comb. Theory B*, 122:353–383, 2017. doi:10.1016/J.JCTB.2016.06.007.

17 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.

18 Konrad K. Dabrowski, Carl Feghali, Matthew Johnson, Giacomo Paesani, Daniël Paulusma, and Paweł Rzążewski. On cycle transversals and their connected variants in the absence of a small linear forest. *Algorithmica*, 82(10):2841–2866, 2020. doi:10.1007/S00453-020-00706-6.

19 Andrzej Dudek, Jarosław Grytczuk, and Andrzej Ruciński. Ordered unavoidable sub-structures in matchings and random matchings. *Electron. J. Comb.*, 31(2), 2024. doi:10.37236/11932.

20 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*, LIPIcs, pages 40:1–40:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.ESA.2025.40.

21 Michael R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976. doi:10.1016/0304-3975(76)90059-1.

22 Peter Gartland and Daniel Lokshtanov. Independent set on P_k-free graphs in quasi-polynomial time. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 613–624. IEEE, 2020. doi:10.1109/FOCS46700.2020.00063.

23 Peter Gartland, Daniel Lokshtanov, Tomás Masařík, Marcin Pilipczuk, Michał Pilipczuk, and Paweł Rzążewski. Maximum weight independent set in graphs with no long claws in quasi-polynomial time. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 683–691. ACM, 2024. doi:10.1145/3618260.3649791.

24 Peter Gartland, Daniel Lokshtanov, Marcin Pilipczuk, Michał Pilipczuk, and Paweł Rzążewski. Finding large induced sparse subgraphs in $C_{>t}$-free graphs in quasipolynomial time. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 330–341. ACM, 2021. doi:10.1145/3406325.3451034.

25 Fanica Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. *SIAM J. Comput.*, 1(2):180–187, 1972. doi:10.1137/0201013.

26 Martin C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*, volume 57 of *Annals of Discrete Mathematics*. Elsevier/North-Holland, 2nd edition, 2004.

27 Andrzej Grzesik, Tereza Klimošová, Marcin Pilipczuk, and Michał Pilipczuk. Polynomial-time algorithm for maximum weight independent set on P_6-free graphs. *ACM Trans. Algorithms*, 18(1):4:1–4:57, 2022. doi:10.1145/3414473.

28 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.

29 Johan Håstad. Clique is hard to approximate within $n^{(1-\varepsilon)}$. *Electron. Colloquium Comput. Complex.*, TR97, 1997. URL: https://eccc.weizmann.ac.il/eccc-reports/1997/TR97-038/index.html.

30 Lenwood S. Heath, F. Thomson Leighton, and Arnold L. Rosenberg. Comparing queues and stacks as mechanisms for laying out graphs. *SIAM Journal on Discrete Mathematics*, 5(3):398–412, 1992. doi:10.1137/0405031.

31 Lenwood S. Heath and Arnold L. Rosenberg. Laying out graphs using queues. *SIAM J. Comput.*, 21(5):927–958, 1992. doi:10.1137/0221055.

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

33 Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. doi:10.1007/978-1-4684-2001-2_9.

34 Dániel Korándi, Gábor Tardos, István Tomon, and Craig Weidert. On the Turán number of ordered forests. *J. Comb. Theory A*, 165:32–43, 2019. doi:10.1016/J.JCTA.2019.01.006.

35 Bingkai Lin, Xuandi Ren, Yican Sun, and Xiuhan Wang. Improved hardness of approximating k-Clique under ETH. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 285–306. IEEE, 2023. doi:10.1109/FOCS57990.2023.00025.

36 László Lovász and Michael D. Plummer. *Matching Theory*, volume 121 of *North–Holland Mathematics Studies*. North–Holland, Amsterdam – New York – Oxford – Tokyo, 1986.

37 Vadim V. Lozin and Martin Milanič. A polynomial algorithm to find an independent set of maximum weight in a fork-free graph. *J. Discrete Algorithms*, 6(4):595–604, 2008. doi:10.1016/J.JDA.2008.04.001.

38 János Pach and István Tomon. Erdős-Hajnal-type results for monotone paths. *J. Comb. Theory B*, 151:21–37, 2021. doi:10.1016/J.JCTB.2021.05.004.

39 János Pach and Gábor Tardos. Forbidden paths and cycles in ordered graphs and matrices. *Israel Journal of Mathematics*, 155(1):359–380, December 2006. doi:10.1007/bf02773960.

40 Giacomo Paesani, Daniël Paulusma, and Paweł Rzążewski. Feedback Vertex Set and Even Cycle Transversal for H-free graphs: Finding large block graphs. *SIAM J. Discret. Math.*, 36(4):2453–2472, 2022. doi:10.1137/22m1468864.

41 Marta Piecyk and Paweł Rzążewski. List coloring ordered graphs with forbidden induced subgraphs. In Meena Mahajan, Florin Manea, Annabelle McIver, and Kim Thang Nguyen, editors, *43rd International Symposium on Theoretical Aspects of Computer Science, STACS 2026, Grenoble, France, March 9-13, 2026*, LIPIcs, pages 74:1–74:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2026. doi:10.4230/LIPIcs.STACS.2026.74.

42 Marcin Pilipczuk, Michał Pilipczuk, and Paweł Rząẽwski. 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.

43 Svatopluk Poljak. A note on stable sets and colorings of graphs. *Commentationes Mathematicae Universitatis Carolinae*, 15:307–309, 1974.

44 Alex Scott, Paul D. Seymour, and Sophie Spirkl. Pure pairs VI: Excluding an ordered tree. *SIAM J. Discret. Math.*, 36(1):170–187, 2022. doi:10.1137/20M1368331.

45 Gábor Tardos. Extremal theory of vertex or edge ordered graphs. In Allan Lo, Richard Mycroft, Guillem Perarnau, and Andrew Treglown, editors, *Surveys in Combinatorics, 2019: Invited lectures from the 27th British Combinatorial Conference, Birmingham, UK, July 29 - August 2, 2019*, pages 221–236. Cambridge University Press, 2019. doi:10.1017/9781108649094.008.