Double-Exponential and Triple-Exponential Bounds for Choosability Problems Parameterized by Treewidth*

Dániel Marx[1] and Valia Mitsou[2]

1 **Institute for Computer Science and Control, Hungarian Academy of Sciences, Budapest, Hungary**
dmarx@cs.bme.hu

2 **Institute for Computer Science and Control, Hungarian Academy of Sciences, Budapest, Hungary**
vmitsou@sztaki.hu

Abstract

Choosability, introduced by Erdős, Rubin, and Taylor [*Congr. Number.* 1979], is a well-studied concept in graph theory: we say that a graph is c-choosable if for any assignment of a list of c colors to each vertex, there is a proper coloring where each vertex uses a color from its list. We study the complexity of deciding choosability on graphs of bounded treewidth. It follows from earlier work that 3-choosability can be decided in time $2^{2^{O(w)}} \cdot n^{O(1)}$ on graphs of treewidth w. We complement this result by a matching lower bound giving evidence that double-exponential dependence on treewidth may be necessary for the problem: we show that an algorithm with running time $2^{2^{o(w)}} \cdot n^{O(1)}$ would violate the Exponential-Time Hypothesis (ETH). We consider also the optimization problem where the task is to delete the minimum number of vertices to make the graph 4-choosable, and demonstrate that dependence on treewidth becomes triple-exponential for this problem: it can be solved in time $2^{2^{2^{O(w)}}} \cdot n^{O(1)}$ on graphs of treewidth w, but an algorithm with running time $2^{2^{2^{o(w)}}} \cdot n^{O(1)}$ would violate ETH.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.2.2 Graph Theory

Keywords and phrases Parameterized Complexity, List coloring, Treewidth, Lower bounds under ETH

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.28

1 Introduction

Most NP-hard algorithmic problems become significantly easier when restricted to bounded-treewidth graphs. There are notable exceptions that remain NP-hard on graphs of constant treewidth (e.g., Steiner Forest [1, 20], List Edge Coloring [37], Edge-disjoint Paths [41]), but most of the natural combinatorial problems can be solved in polynomial time (or even in linear time) if treewidth is bounded. Courcelle's Theorem [11] is a meta-result showing that if a problem can be expressed in the language of monadic second order logic (MSOL), then it can be solved in linear-time on bounded-treewidth graphs: there is an algorithm with running time $f(w) \cdot n$, where w is the treewidth of the graph. While this result

* This work was supported by ERC Starting Grant PARAMTIGHT (No. 280152) and OTKA grant NK105645.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 28; pp. 28:1–28:15

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

immediately gives algorithms for a large number of combinatorial problems, focus has shifted in recent years towards trying to obtain a tighter understanding of how the running time has to depend on treewidth, that is, understanding what the best possible $f(w)$ in the running time can be. On the one hand, recent algorithm advances, such as *Cut & Count* [14] and fast subset convolution [3, 47], resulted in improved dependence on treewidth for various problems. On the other hand, conditional results based on the Exponential-Time Hypothesis (ETH) [25, 26, 34] give lower bounds on the best possible dependence that can be achieved, and in many cases these lower bounds tightly match the known algorithms [13, 14, 33, 35, 43]. (ETH can be informally stated as n-variable 3SAT cannot be solved in time $2^{o(n)}$.) Most of these tight bounds are of the form $2^{O(w)}$ (e.g., 3-Coloring, Hamiltonian Cycle, Triangle Packing, etc.), but there are surprising exceptions where the best possible dependence on treewidth is $2^{O(w \log w)}$ [14, 35] or even $2^{O(w^c)}$ for some constant $c > 1$ [13]. A result of Frick and Grohe [19] shows that, assuming $\mathsf{P} \neq \mathsf{NP}$, the dependence on treewidth can be really bad for some problems: Courcelle's Theorem does not remain true if we impose any *elementary* bound on the function $f(w)$. That is, for every $h \geq 1$, there are MSOL sentences and corresponding model-checking problems for which the dependence on treewidth is an exponential tower of height h. Pan and Vardi [42] gave strong evidence that this bad performance is expected for problems high up in the polynomial hierarchy: they showed that, under ETH, there is a strict hierarchy of lower bounds of the form $2^{2^{\cdots^{2^{o(w)}}}}$ inside PSPACE. In this paper, we present two fairly standard graph-theoretic problems that require double-exponential and triple-exponential dependence on treewidth, respectively.

1.1 k-Choosability

A proper k-coloring of a graph G is a mapping $f : V(G) \to [k]$ such that $f(u) \neq f(v)$ for any two adjacent $u, v \in V(G)$. List coloring is the generalization where instead of having the same set $[k]$ of colors available at each vertex, each vertex v has its own list $L(v)$ of available colors and the question is whether there is a proper coloring f that assigns a color $f(v) \in L(v)$ to each vertex v. The algorithmic aspects of list coloring have received significant interest [2, 10, 23, 24, 29, 32, 37, 38, 46].

Erdős, Rubin, and Taylor [16] defined a combinatorial property related to list colorings: we say that a graph G is k-choosable if it has a coloring for any list assignment L that has size k at each vertex. Clearly, k-choosability implies k-colorability. One may feel that the converse implication could also be true: after all, it may seem safe to guess that the "worst case" of list coloring is when every vertex has the same list $[k]$. But this intuition is wrong: for example, for any $k \geq 1$, there are 2-colorable graphs that are not k-choosable.

From the computational complexity point of view, deciding k-choosability is a much harder algorithmic problem than deciding k-colorability. Deciding k-choosability does not seem to belong to the class NP (a witness for k-choosability would need to prove colorability for *every* list assignment L) or to the class coNP (an uncolorable list assignment would be a good witness for non-k-choosability, but it cannot be verified in polynomial time). In fact, the k-choosability problem is known to lie higher in the polynomial hierarchy.

▶ **Theorem 1** (from [21]). *k-Choosability for $k \geq 3$ is Π_2^p-complete.*

We observe a similar gap in complexity between the two problems when looking at algorithms parameterized by treewidth. 3-colorability can be decided in time $3^w \cdot n^{O(1)}$ using standard techniques [12]. Fellows et al. [17] showed that deciding 3-choosability is fixed-parameter tractable parameterized by treewidth. One can make this result quantitative

by observing that the running time is actually $2^{2^{O(w)}} \cdot n^{O(1)}$ for graphs of treewidth w. Our first result shows that this double exponential dependence on treewidth is best possible, assuming ETH.

▶ **Theorem 2.** *For every fixed $k \geq 3$:*

1. *There is an algorithm for k-Choosability with running time $2^{2^{O(w)}} \cdot n^{O(1)}$ on graphs of treewidth w.*
2. *Assuming ETH, there is no algorithm for k-Choosability with running time $2^{2^{o(w)}} \cdot n^{O(1)}$ on graphs of treewidth w.*

1.2 k-Choosability Deletion

Given any class $\mathcal{C}$ of graphs, one can define various graph modification problems where the task is to transform a given graph G into a member of $\mathcal{C}$ with the minimum number of vertex deletions/edge deletions/edge additions. The parameterized algorithms literature is especially rich in this type of problems, where the goal is, for example, to make the graph acyclic [9, 15], bipartite [27, 36, 44, 45], planar [28, 40], chordal [5, 7, 18, 30, 39], or interval [4, 6, 8]. Investigating these problems on graphs of bounded treewidth is an interesting question on its own right, but additional motivation comes from the fact that some of the algorithms on *general graphs* first reduce the treewidth and then invoke an algorithm exploiting bounded treewidth [28, 39, 40].

If we look at the vertex deletion versions of coloring and choosability problems, then we can observe an even larger gap than in the decision version. For technical reasons, we give a proof only for $k \geq 4$ colors. It is not difficult to show (we leave it as an exercise to the reader) that there is an algorithm with running time $2^{O(w)} \cdot n^{O(1)}$ for 4-Coloring Deletion that, given a graph G of treewidth w, computes the minimum number of vertices that needs to be deleted to make the graph 4-colorable. On the other hand, if we consider the 4-Choosability Deletion problem, which asks for the minimum number of vertices that need to be deleted to make a given graph G 4-choosable, then we need *triple-exponential* dependence on treewidth.

▶ **Theorem 3.** *For every fixed $k \geq 4$:*

1. *There is an algorithm for k-Choosability Deletion with running time $2^{2^{2^{O(w)}}} \cdot n^{O(1)}$ on graphs of treewidth w.*
2. *Assuming ETH, there is no algorithm for k-Choosability Deletion with running time $2^{2^{2^{o(w)}}} \cdot n^{O(1)}$ on graphs of treewidth w.*

As a side result, we show that k-Choosability Deletion lies one level higher than k-Choosability in the polynomial hierarchy (compare this with the fact that k-Coloring and k-Coloring Deletion are both in NP).

▶ **Theorem 4.** *For every $k \geq 3$, k-Choosability Deletion is Σ_3^p-complete.*

The reader may feel that the Π_2^p- and Σ_3^p-completeness of these problems already give sufficient explanation why double- or triple-exponential dependence on treewidth is needed. This is true in some sense: the quantifier alternations in the problem definitions are the common underlying reasons for being in the higher levels of the polynomial hierarchy and for requiring unusually large dependence on treewidth. But let us point out that these two types of complexity results require very different proof structures. In Π_2^p- or Σ_3^p-completeness proofs, we start with a canonical Π_2^p- or Σ_3^p-complete quantified satisfiability problem and we use the alternations inherent in the definitions of k-Choosability or k-Choosability

Deletion to express the alternations in quantified satisfiability. On the other hand, in the proofs of Theorems 2(2) and 3(2), we start with problems in NP and use the alternations inherent in k-Choosability or k-Choosability Deletion for *compression*: we want to express the original instance by a graph having treewidth only $O(\log n)$ or $O(\log \log n)$. Thus the main theme of our proofs is trading alternation for compression: we want to use alternation to allow the succinct encoding and verification of information. Note also that both k-Choosability and k-Choosability Deletion can be solved in time $2^{n^{O(1)}}$, hence the exponential explosion appears only in the context of bounded-treewidth graphs.

2 Preliminaries

▶ **Definition 5** (List coloring)**.** Given graph G together with sets $\mathcal{L}(v) \subset \mathbb{N}$, one for every vertex v (we shall call $\mathcal{L}(v)$ the *list* of v), then G is *$\mathcal{L}$-colorable* if there exists a coloring $c : V(G) \to \mathbb{N}$, which is proper and for which $\forall v \in V(G), c(u) \in \mathcal{L}(v)$. In that case, c is called a *proper $\mathcal{L}$-coloring*.

▶ **Definition 6** (Choice number)**.** Given G and some $k \in \mathbb{N}$, G is called *k-choosable* if for any list assignment $\mathcal{L} : V(G) \to 2^{\mathbb{N}}$ with $|\mathcal{L}(v)| = k$ for all $v \in V(G)$, G is $\mathcal{L}$-colorable. The *choice number* (or *list-chromatic number*) of G, denoted by $\chi_\ell(G)$, is the minimum number k such that G is k-choosable.

The notion of f-choosability defined below generalizes k-choosability, but for arbitrary list sizes for each vertex.

▶ **Definition 7** (f-choosable graphs)**.** A graph G is called *f-choosable* for some function $f : V(G) \to \mathbb{N}$ (called *list-capacity function*) if G is $\mathcal{L}$-colorable for any list assignment $\mathcal{L} : V(G) \to 2^{\mathbb{N}}$ where $|\mathcal{L}(v)| = f(v)$ for all $v \in V(G)$.

As a direct consequence of the definition, we get that the null graph K_0 with no vertices is f-choosable for any list-capacity function f. The reason is the vacuous truth that, for any list assignment $\mathcal{L}$, the empty function $e : \emptyset \to \mathbb{N}$ is a proper $\mathcal{L}$-coloring of K_0.

▶ **Definition 8** (Color Compatibility)**.** Given a graph G, a list assignment $\mathcal{L}$, an ordered k-tuple $(u_1, u_2, \ldots, u_k) \in V(G)^k$ and an ordered k-tuple of colors $(c_1, c_2, \ldots, c_k)$ with $c_i \in \mathcal{L}(u_i)$ for $i \in [k]$, we say that $(c_1, c_2, \ldots, c_k)$ is *compatible* on $(u_1, u_2, \ldots, u_k)$ if there exists a proper $\mathcal{L}$-coloring g of G with $g(u_i) = c_i$. We may omit G and $\mathcal{L}$ if they are clear from the context.

We define the following algorithmic problems:

- (i, j)-Choosability: Given a graph G, integers $i, j \in \mathbb{N}$ where $i \leq j$, and a list-capacity function f for which $f(v) \in \{i, \ldots, j\}$ for any vertex $v \in V$, decide whether G is *f-choosable*.
- k-Choosability:= (k, k)-Choosability.
- (i, j)-Choosability Deletion: Given a graph G, integers $i, j, r \in \mathbb{N}$ with $i \leq j$, and a list-capacity function f for which $f(v) \in \{i, \ldots, j\}$ for any vertex $v \in V$, decide whether there exists $U \subset V$ with $|U| \leq r$ such that $G - U$ is *f-choosable*.
- k-Choosability Deletion:= (k, k)-Choosability Deletion.

Below are some known results about Choosability.

▶ **Theorem 9** (from [16])**.** $(2, 3)$-Choosability *is Π_2^p-complete.*

▶ **Theorem 10** (from [21])**.** k-Choosability *for $k \geq 3$ is Π_2^p-complete.*

▶ **Theorem 11** (from [17])**.** k-Choosability *parameterized by the treewidth of the input graph is in* FPT.

3 Double-exponential lower bound for k-Choosability

The topic of this section is proving Theorem 2(2), the double-exponential lower bound on 3-Choosability parameterized by treewidth. For the proof, we will find it convenient to start the reduction from Edge 3-Coloring, where given a graph G, the task is to decide if G has a proper edge 3-coloring (that is, whether there is an assignment $c : E(G) \to \{1, 2, 3\}$ such that $c(e_1) \neq c(e_2)$ for any two edges e_1 and e_2 sharing an endpoint). Holyer's proof [22] for the NP-hardness of Edge 3-Coloring on 3-regular graphs can be observed to give a tight lower bound under ETH. Note that a 3-regular graph on n vertices has exactly $3n/2 = O(n)$ edges. We state the lower bound in a slightly awkward way, but this type of bound is what we exactly need.

▶ **Theorem 12** (follows from [22]). *Assuming ETH,* Edge 3-Coloring *cannot be decided in time* $2^{2^{o(\log n)}}$*, where* n *is the number of vertices of the graph. Moreover, this remains true even if we consider only 3-regular graphs whose number of vertices is an integer power of* 2.

That is, if we reduce Edge 3-Coloring to some other problem by creating an equivalent instance with treewidth $O(\log n)$, then an algorithm of the target problem with running time $2^{2^{o(w)}} \cdot n^{O(1)}$ for graphs of treewidth w would yield an algorithm with running time $2^{2^{o(\log n)}} \cdot n^{O(1)}$ for Edge 3-Coloring, contradicting Theorem 12.

The reason why reducing from this problem is convenient for our proof is that Edge 3-Coloring involves constraints of the form "the three edges e_1, e_2, e_3 incident to a vertex u use all three colors from $\{1, 2, 3\}$," and this type of constraints will be easy to express in our reduction. However, there is an unfortunate presentation issue: when talking about colors, vertices, and edges, it may not be immediately clear if we refer to the source Edge 3-Coloring instance or to the target 3-Choosability instance, and this may cause confusion. Therefore, we prefer to treat Edge 3-Coloring as a constraint satisfaction problem and use terminology appropriate to that. Formally, an instance of Edge 3-Coloring can be interpreted as a problem where we have a set X of n variables (corresponding to the edges of G), each variable has the domain $\{1, 2, 3\}$, and we have a set Y of m constraints (corresponding to the vertices of G). Each constraint contains exactly three distinct variables, and the constraint is satisfied if these variables are assigned three different values. Then the proper edge 3-colorings of G are in one-to-one correspondence with the satisfying assignments of this constraint satisfaction problem. Note that each variable appears in exactly two constraints.

Given an instance H of the constraint satisfaction problem described above, we create an instance of $(2, 3)$-Choosability, i.e, a graph G and a list-capacity function $f : V \to \{2, 3\}$. Then we show that there is a satisfying assignment of I if and only if G is *not* $(2, 3)$-choosable. We will further make sure that $\mathrm{tw}(G) = O(\log m)$, which will imply the desired time lower bound. For the forward direction, all we need to do is, given a satisfying assignment $h : X \to \{1, 2, 3\}$ of H, use h in order to define a list assignment $\mathcal{L}$ for which G is not $\mathcal{L}$-colorable. The converse direction is somewhat more involved, as we need to start from the assumption that G is not f-choosable and, given some uncolorable list assignment $\mathcal{L}'$, show that H is satisfiable. However, since we do not know exactly how the list assignment $\mathcal{L}'$ which fails to proper $\mathcal{L}'$-color G looks like, we need to consider more general properties which apply to any such potential list assignment.

3.1 Gadgets

Before presenting the main reduction, we introduce three different types of gadgets and prove their properties. Corresponding to the two directions of the proof of correctness of the

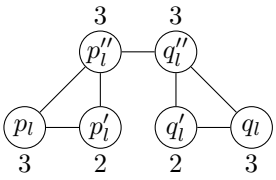

Figure 1 The *Bit-chooser* gadget B_l of the l^{th} bit.

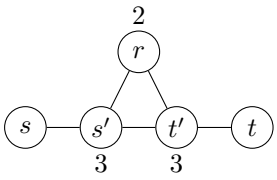

Figure 2 The *Weak-edge* gadget W_{st}, connecting vertices s, t.

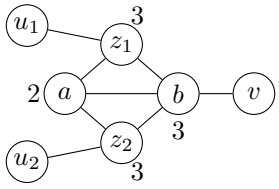

Figure 3 The *Weak-star* gadget S_i for variable x_i.

reduction, we show two types of properties for each gadget: an "$\exists$ list $\mathcal{L}$" property, which we use in order to define a list assignment $\mathcal{L}$ for the forward direction, and a "$\forall$ lists $\mathcal{L}$" property, which shall be useful in proving the converse direction.

1. Bit-chooser gadget B_l (Figure 1). Informally, a list assignment of the bit chooser gadget can enforce that a certain color appears on at least one of the two outputs p_l and q_l, but this is all it can do: in every list assignment, there is a "good" color on p_l and q_l that is compatible with every color on the other output.

▶ **$\exists\mathcal{L}$-Property 1.** There exists a list assignment $\mathcal{L}$ and a color c with $c \in \mathcal{L}(p_l) \cap \mathcal{L}(q_l)$ such that for any proper $\mathcal{L}$-coloring of B_l, at least one of p_l, q_l should receive color c.

▶ **$\forall\mathcal{L}$-Property 1.** For every list assignment $\mathcal{L}$, there exists a color $c \in \mathcal{L}(p_l)$ (resp., $\mathcal{L}(q_l)$) such that for every color $c' \in \mathcal{L}(q_l)$ (resp., $c' \in \mathcal{L}(p_l)$) the pair (c, c') is compatible on (p_l, q_l) (resp., on (q_l, p_l)).

2. Weak-edge gadget W_{st} (Figure 2). Informally, the Weak-edge gadget can prevent a certain combination (c, c') of colors appearing on the two outputs, but it cannot prevent more than one such combinations.

▶ **$\exists\mathcal{L}$-Property 2.** There exists a list assignment $\mathcal{L}$ and colors $c \in \mathcal{L}(s)$, $c' \in \mathcal{L}(t)$ such that the pair (c, c') is incompatible on (s, t).

▶ **$\forall\mathcal{L}$-Property 2.** For every list assignment $\mathcal{L}$, there exists at most one $(c, c') \in \mathcal{L}(s) \times \mathcal{L}(t)$ such that (c, c') incompatible on (s, t). In fact, if (c, c') is incompatible on (s, t), then the following hold: $c, c' \notin \mathcal{L}(r)$; $\mathcal{L}(s') = \{c\} \cup \mathcal{L}(r)$; and $\mathcal{L}(t') = \{c'\} \cup \mathcal{L}(r)$.

Because of their properties, the Weak-edges, if given an appropriate list assignment $\mathcal{L}$, can define an incompatible pair of colors being mutually assigned to their endpoints. Another way to view a Weak-edge is to consider it a directed edge between a precolored vertex s and an uncolored vertex t potentially forbidding a particular color from being assigned to t: if, for W_{st}, (c, c') is incompatible on (s, t) and s has already been assigned color c, then we know that t cannot receive color c'. Observe that, because of the $\forall\mathcal{L}$-Property, each Weak-edge defines at most one such incompatible pair between the endpoints.

3. Weak-star gadget S_i (Figure 3). Informally, the Weak-star gadget has the property that there is at most one color c on v that can forbid one specific color c_1 on u_1 and one specific color c_2 on u_2. When using this gadget on a vertex v whose list size is 2, it will act like an OR-gadget: if this specific color appears on u_1 or u_2, then it forbids color c on v, and hence forces it to take the other color in the list of v.

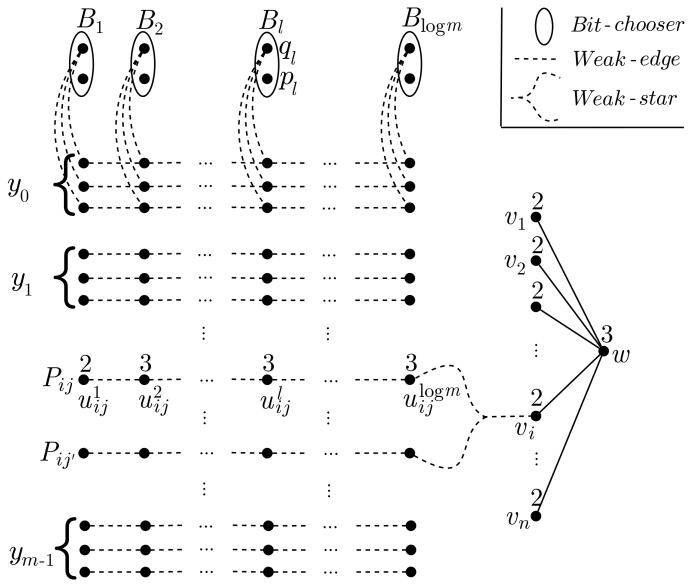

Figure 4 An overview of the construction.

▶ **$\exists\mathcal{L}$-Property 3.** There exist a list assignment $\mathcal{L}$ and a color $c \in \mathcal{L}(u_1) \cap \mathcal{L}(u_2) \cap \mathcal{L}(v)$ such that (c, c) is incompatible on both (u_1, v) and (u_2, v).

▶ **$\forall\mathcal{L}$-Property 3.** For every list assignment $\mathcal{L}$, there exist colors $c_1 \in \mathcal{L}(u_1), c_2 \in \mathcal{L}(u_2), c \in \mathcal{L}(v)$ such that if (d_1, d_2, d) is incompatible on (u_1, u_2, v) then $d = c$ and $(d_1 = c_1) \vee (d_2 = c_2)$.

3.2 Construction

We will now proceed with the description of the construction of G. Consider an arbitrary ordering of the m constraints of Y, assigning a unique index $\{0, \ldots, m-1\}$ to each of them. Now construct $\log m$ *Bit-chooser* gadgets $B_1, \ldots, B_{\log m}$ as shown in Figure 1. For gadget B_l, vertex p_l shall correspond to bit 1 and vertex q_l to 0.

Furthermore, for each appearance of a variable x_i in constraint y_j we construct a *Chain* P_{ij}, which is essentially a path on $\log m$ vertices $u_{ij}^1, \ldots, u_{ij}^{\log m}$, where we have substituted every edge (u_{ij}^l, u_{ij}^{l+1}) by a *Weak-edge* gadget $W_{u_{ij}^l u_{ij}^{l+1}}$ as the one shown in Figure 2, by identifying u_{ij}^l with s and u_{ij}^{l+1} with t. We also set $f(u_{ij}^1) = 2$ and $f(u_{ij}^l) = 3$ for $l > 1$.

Then we need to connect the Chains to the Bit-choosers. To do so, for each Chain P_{ij}, we write j in binary representation and for $l = 1, \ldots, \log m$, if the l^{th} bit is 1, we connect u_{ij}^l to p_l, else we connect it to q_l. The connection is by a Weak-edge $W_{p_l u_{ij}^l}$ or $W_{q_l u_{ij}^l}$ respectively.

In addition, we construct n variable-vertices $v_1, \ldots, v_n$, one for each variable $x_1, \ldots, x_n$ and we set $f(v_i) = 2$. Eventually, we need to connect the two Chains which correspond to the two appearances of x_i to vertex v_i. In order to do so, we use a *Weak-star* gadget S_i (see Figure 3). For Chains P_{ij} and $P_{ij'}$ with $j < j'$, corresponding to variable x_i in constraints y_j and $y_{j'}$, respectively, we identify $u_{ij}^{\log m}$ with u_1, $u_{ij'}^{\log m}$ with u_2, and v with variable-vertex v_i.

Last, we construct a *checker* vertex w with $f(w) = 3$ and connect it to all $v_1, \ldots, v_n$ (using an ordinary edge). This completes the construction. Our claim is that H is satisfiable if and only if G is not f-choosable.

It is easy to verify that the constructed graph G has pathwidth (and hence treewidth) bounded by $O(\log m)$.

▶ **Lemma 13.** *$pw(G) = O(\log m)$.*

3.3 Satisfying assignment ⇒ uncolorable list assignment

In this section, we prove the forward direction of the correctness of the reduction.

▶ **Lemma 14.** *If H is satisfiable, then G is not f-choosable.*

Proof. Assume that H is satisfiable. Then there should be an assignment $h : X \to \{1, 2, 3\}$ satisfying all the constraints in Y. We will produce a list assignment $\mathcal{L}$ for which the graph will not be $\mathcal{L}$-colorable.

First, we construct the lists of the v_i's according to the satisfying assignment of H, $\mathcal{L}(v_i) = \{h(x_i), c\}$. For the checker vertex w, we have $\mathcal{L}(w) = \{1, 2, 3\}$. For the main vertices of the Chains P_{ij}, we set $\mathcal{L}(u^1_{ij}) = \{c, c'\}$ and $\mathcal{L}(u^l_{ij}) = \{c, c', c''\}$, for $l \in \{2, \ldots, \log m\}$. Last, for the gadget vertices, we shall construct their lists according to those from the proofs of the $\exists\mathcal{L}$-Properties, as described below.

- For vertices in the Bit-choosers, lists match exactly those in the proof of $\exists\mathcal{L}$-Property 1.
- For vertices in the Weak-stars, lists match exactly those in the proof of $\exists\mathcal{L}$-Property 3.
- The Weak-edges should specify appropriate incompatible pairs of colors on their endpoints, and the list assignment should follow that of proof of $\exists\mathcal{L}$-Property 2: Weak-edges connecting Bit-choosers to Chains should forbid (c, c') on (p_l, u^l_{ij}) (resp., (q_l, u^l_{ij})); Weak-edges interconnecting Chain-vertices should forbid (c, c'') on $(u^{(l-1)}_{ij}, u^l_{ij})$.

Let us now explain why G is not $\mathcal{L}$-colorable, no matter which colors we pick from the lists. We are going to show that any possible coloring of the Bit-choosers accordant with $\mathcal{L}$ corresponds to selecting a constraint y, by selecting a 0-1 value for $\log m$ bits, which together select a constraint index in $\{0, \ldots, m-1\}$.

From $\exists\mathcal{L}$-Property 1 and for the particular list assignment which emerges from its proof, we are required to *select* (at least) one of the two endpoints p_l, q_l by giving it a special color c. Selecting p_l is interpreted as setting the l^{th} bit to 1, whereas selecting q_l is interpreted as setting it to 0. A selection of $\log m$ bit-vertices (one from each Bit-chooser) corresponds to picking a binary number, which shall represent the index of the constraint we select.

Suppose that a constraint $y_j = (x_{j_1}, x_{j_2}, x_{j_3})$ is selected this way. This means that for the 3 Chains $P_{j_1 j}, P_{j_2 j}, P_{j_3 j}$ we have color c' being removed from all $\mathcal{L}(u^l_{j_i j})$. In particular, this means that $u^1_{j_i j}$ has the unique color c available for it, which in turn forces color c to $u^2_{j_i j}$ by forbidding c'', and so on. This way, color c will propagate throughout the Chains, forcing color c on $u^{\log m}_{ij}$.

From $\exists\mathcal{L}$-Property 3 and for the list described in its proof, forcing color c on $u^{\log m}_{ij}$ forbids color c on v_i, forcing us to choose color $h(x_i)$, which in turn should forbid $h(x_i)$ from w. Since we have three variable-vertices $v_{j_1}, v_{j_2}, v_{j_3}$ with forced choices and $x_{j_1}, x_{j_2}, x_{j_3}$ belong to the same constraint y_j which h satisfies, $h(x_{j_1}), h(x_{j_2}), h(x_{j_3})$ should all be distinct values from $\{1, 2, 3\}$. Thus, all three colors should be forbidden in w, which is fatal since $\mathcal{L}(w) = \{1, 2, 3\}$. ◀

3.4 Uncolorable list assignment ⇒ satisfying assignment

Let us now proceed with the converse direction. First we determine some properties that every uncolorable list assignment $\mathcal{L}'$ of G needs to have. Then we extract an assignment

from $\mathcal{L}'$ for H and use the properties of the lists to show that it is a satisfying assignment.

The following definition will be convenient in the proofs. Let B' be the vertices of all the $\log m$ Bit-choosers. Given some list assignment $\mathcal{L}$, we say that a partial $\mathcal{L}$-coloring $g : B' \to \mathbb{N}$ *activates* vertex u^l_{ij} if $g(p_l) = c$ for some color $c \in \mathcal{L}(p_l)$ and there is a Weak-edge $W_{p_l u^l_{ij}}$ forbidding (c, c') on (p_l, u^l_{ij}) (or similarly for q_l). Thus, if we were to extend g to a proper $\mathcal{L}$-coloring of the whole graph, we would have one less choice for u^l_{ij}. In this case, vertex u^l_{ij} is called *active*. A Chain P_{ij} is called *active* if all its vertices are active. The crucial observation is that if a Chain P_{ij} is not active, no matter what color $c \in \mathcal{L}'(u^{\log m}_{ij})$ we assign to $u^{\log m}_{ij}$, this can be extended to a proper $\mathcal{L}'$-partial coloring of the chain P_{ij}. Suppose for example that vertex u^l_{ij} is not activated, that is, the coloring on the Bit-chooser B_l does not forbid any color on u^l_{ij}. Then we can start coloring the vertices u^1_{ij}, u^2_{ij}, …, u^{l-1}_{ij} in this order, then the vertices $u^{\log m}_{ij}$, $u^{\log m-1}_{ij}$, …, u^{l+1}_{ij} in this order, and there is still at least one available color left for u^l_{ij}. Furthermore, even if P_{ij} is active, it is possible to extend the partial coloring of the Bit-choosers to it, but this may force a certain color on $u^{\log m}_{ij}$.

▶ **Lemma 15.** *If G is not $\mathcal{L}'$-colorable, then any partial $\mathcal{L}'$-coloring of the Bit-choosers activates at least 3 of the Chains.*

Proof. Consider a partial $\mathcal{L}'$-coloring of the Bit-choosers. Suppose that only two Chains $P_{i_1 j_1}$, $P_{i_2 j_2}$ are activated, potentially forcing a coloring on $u^{\log m}_{i_1 j_1}$ and $u^{\log m}_{i_2 j_2}$. By $\forall\mathcal{L}$-Property 3, the $\mathcal{L}'$-coloring can be extended through the Weak-stars S_{i_1} and S_{i_2} even if $i_1 = i_2$, potentially reducing $|\mathcal{L}'(v_{i_1})|$ and $|\mathcal{L}'(v_{i_2})|$ to a unique choice. For every other vertex $u^{\log m}_{ij}$ vertex, any color can be extended to its Chain. Hence the Weak-star S_i for $i \notin \{i_1, i_2\}$ does not forbid any color on v_i.

Now observe that, even if both v_{i_1}, v_{i_2} have forced colors c_1, c_2, there is always a third color $c \in \mathcal{L}'(w)$, $c_1 \neq c \neq c_2$, which we can assign to w. Further observe that, since all other v_i have two available compatible choices with the rest of G, there will be at least one color in every v_i's list which should be different than c. Use these colors to complete a proper $\mathcal{L}'$-coloring for G. Of course this is a contradiction. Thus there should be exactly 3 active Chains. ◀

▶ **Lemma 16.** *For every list assignment $\mathcal{L}'$, there is some partial $\mathcal{L}'$-coloring of the Bit-choosers that activates exactly 3 Chains. Furthermore, for every constraint there exists a partial $\mathcal{L}'$-coloring that activates its 3 corresponding Chains.*

Proof. From $\forall\mathcal{L}$-Property 1, we know that there exists a color $c_p \in \mathcal{L}'(p_l)$ which is compatible with all colors in $\mathcal{L}'(q_l)$ and a color $c_q \in \mathcal{L}'(q_l)$ which is compatible with all colors in $\mathcal{L}'(p_l)$. Let c'_p be an arbitrary color of $\mathcal{L}'(p_l)$ different from c_p, and let c'_q be an arbitrary color of $\mathcal{L}'(q_l)$ different from c_q. Note that both (c_p, c'_q) and (c'_p, c_q) can be extended to the Bit-chooser B_l and we obtain two different colorings for B_l this way.

Let us consider all possible colorings of the Bit-choosers that arise from selecting one of these two colorings for each B_l; there are $2^{\log m} = m$ combinations. We claim that each Chain P_{ij} is activated by at most one of these colorings. Suppose that there are two colorings that both activate P_{ij}. The two colorings must differ on at least one of the Bit-choosers, say, on B_l. Suppose that P_{ij} is connected to p_l with a Weak-edge (the case when it is connected to q_l is analogous). Color c_p appears on p_l in one of the colorings and color c'_p appears in the other coloring. As $c_p \neq c'_p$, it is not possible that the Weak-edge $W_{p_l u^l_{ij}}$ forces a color on u^l_{ij} in both cases, a contradiction.

Since each of the m colorings activates at least 3 Chains by Lemma 15 and we have shown that each of the $3m$ Chains is activated by at most one of the colorings, a counting argument shows that each coloring activates exactly 3 Chains, and each Chain is activated by exactly one of the m colorings. That is, the Chains can be partitioned into m triples, and each triple can be activated by one of the colorings. To prove the second statement of the lemma, we need to show that each such triple contains Chains corresponding to one of the constraints. Suppose for a contradiction that one of the m colorings activates two Chains P_{ij} and $P_{i'j'}$ with $j \neq j'$. As the numbers j and j' are different, there is a bit l where they differ, which means that, without loss of generality that P_{ij} is connected to p_l, while $P_{i'j'}$ is connected to q_l. Suppose that this coloring assigns (c_p, c'_q) to (p_l, q_l) (the case when (c'_p, c_q) appears is similar). Recall that c_p on p_l is compatible with any color of $\mathcal{L}'(q_l)$ appearing on q_l. Let c''_q be the third color of $\mathcal{L}'(q_l)$, different from c_q and c'_q. We may modify the coloring on B_i such that (c_p, c''_q) appears on Bit-chooser B_l. Then $P_{i'j'}$ will no longer be activated: if color c'_q on q_l activated $u^l_{i'j'}$ via Weak-edge $W_{q_l u^l_{i'j'}}$, then color c''_q surely does not activate it. We observe that this modified coloring cannot activate any Chain that was not active before. Indeed, if some Chain $P_{i^*j^*}$ becomes activated by color c''_q on q_l, then $P_{i^*j^*}$ was not activated by any of the m colorings we considered before, as those colorings assigned only colors c_q and c'_q to q_l. This contradicts our earlier claim that each P_{ij} is activated by exactly one of the m colorings. Thus the modified coloring satisfies strictly less than 3 of the Chains, which contradicts Lemma 15. Thus we can conclude that each of the m colorings activates a (different) triple of Chains corresponding to one the m constraints. ◀

▶ **Lemma 17.** *If G is not f-choosable, then H is satisfiable.*

Proof. We may assume without loss of generality that $\mathcal{L}'(w) = \{1, 2, 3\}$. For every Weak-star S_i, consider the color $c_i \in \mathcal{L}'(v_i)$ given by $\forall\mathcal{L}$-Property 3. We define the assignment $h(x_i) := c^*_i$, where $\{c^*_i\} = \mathcal{L}(v_i) \setminus \{c_i\}$. We show that this gives a satisfying assignment: for every constraint $y_j = (x_{i_1}, x_{i_2}, x_{i_3})$, the colors appearing on the variables x_{i_1}, x_{i_2}, x_{i_3} form the set $\{1, 2, 3\}$; in particular, this will imply $h(i) \in \{1, 2, 3\}$ for every i.

Let us verify that that constraint $y_j = (x_{i_1}, x_{i_2}, x_{i_3})$ is satisfied. By Lemma 16, there is a partial assignment to the Bit-choosers that activates exactly the Chains $P_{i_1 j}$, $P_{i_2 j}$, and $P_{i_3 j}$. This means that the vertices $u^{\log m}_{i_1 j}$, $u^{\log m}_{i_2 j}$, and $u^{\log m}_{i_3 j}$ are forced to some color, but the other vertices $u^{\log m}_{ij}$ are unaffected. Thus for $i \notin \{i_1, i_2, i_3\}$, the Weak-star S_i does not prevent any color on v_{i_t}. But for $t = 1, 2, 3$, the Weak-star S_t may forbid the use of color c_{i_t} on v_{i_t}. In order to avoid any conflicts, we assign color $h(x_{i_t}) \neq c_{i_t}$ to v_i. If $\{h(x_{i_1}), h(x_{i_2}), h(x_{i_3})\} \neq \{1, 2, 3\}$, then we can assign to w a color not appearing on v_{i_1}, v_{i_2}, v_{i_3}, and then extend the coloring to v_i for each $i \notin \{i_1, i_2, i_3\}$ by choosing a color different from the color of w. This would contradict the assumption that $\mathcal{L}'$ is not colorable. Thus $\{h(x_{i_1}), h(x_{i_2}), h(x_{i_3})\} = \{1, 2, 3\}$, that is, constraint y_j is satisfied. ◀

3.5 Lower Bounds

Now we are ready to establish the lower bounds stated in Theorem 2(2).

▶ **Theorem 18.** (2, 3)-Choosability *cannot be decided in time $2^{2^{o(pw)}} \cdot n^{O(1)}$, where pw is the pathwidth of the input graph, under ETH.*

Proof. Suppose we could decide (2,3)-Choosability in time $2^{2^{o(\mathrm{pw})}} \cdot n^{O(1)}$. We know from Lemma 13 that $\mathrm{pw} = O(\log m)$.

Thus $2^{2^{o(\mathrm{pw})}} \cdot n^{O(1)} = 2^{2^{o(\log m)}} \cdot n^{O(1)}$. From Lemmas 14 and 17, this would imply solving Edge 3-Coloring in time $2^{2^{o(\log m)}}$. This contradicts Theorem 12. ◀

▶ **Corollary 19** (Theorem 2(2)). *Assuming ETH, k-Choosability for any $k \geq 3$ cannot be decided in time $2^{2^{o(pw)}} \cdot n^{O(1)}$, where pw is the pathwidth of the input graph.*

Proof. First observe that the problem for $k \geq \mathrm{pw} + 1$ is meaningless, since the answer is trivially yes: a graph G of pathwidth pw is pw-degenerate, and thus $(\mathrm{pw}+1)$-choosable.

Gutner and Tarsi [21] give a reduction from $(2,3)$-Choosability to k-Choosability, where the pathwidth of the constructed graph G' is $O(\mathrm{pw}(G))$. ◀

4 Triple-exponential lower bound for k-Choosability Deletion

The goal of this section is to prove Theorem 3(2): the triple-exponential lower bound for k-Choosability Deletion. For this proof, we choose a variant of list coloring as the source problem of the reduction.

Our reduction is from Bipartite List 3-Coloring: given a bipartite graph H with vertex set $X = X_1 \cup X_2, X_1 = \{x_{10}, x_{11}, \ldots x_{1(n-1)}\}, X_2 = \{x_{20}, x_{21}, \ldots, x_{2(n-1)}\}$, edge set Y with $|Y| = m$, and a list assignment $\mathcal{D} : X \to 2^{\{1,2,3\}}$ with $|\mathcal{D}(x)| \geq 2$ for all $x \in X$, the task is to find a proper 3-coloring $\phi : V(G) \to \{1,2,3\}$ of G where $\phi(v) \in \mathcal{D}(v)$ for every vertex v. This problem is known to be NP-hard [31] (to ensure $|\mathcal{D}(x)| \geq 2$, one needs to observe that any vertex with $|\mathcal{D}(x)| = 1$ can be removed from the graph after omitting its unique color from the lists of its neighbors). The NP-hardness proof can be observed to give a tight lower bound, which we state in the following form.

▶ **Lemma 20.** *Assuming ETH, there is no algorithm for* Bipartite List 3-Coloring *with running time $2^{2^{2^{o(\log\log n)}}}$ on bipartite graphs with n vertices on each side. This remains true if we consider only graphs where $\log\log n$ is integer.*

Proof. Given a 3SAT formula with n_0-variables and m_0-clauses, the reduction of Kratochvíl [31] creates an equivalent instance $(G, \mathcal{D})$ of Bipartite List 3-Coloring with $n_1 = O(n_0+m_0)$ vertices and $m_1 = O(n_0+m_0)$ edges. Let n be the smallest integer not smaller than n_1 such that $\log\log n$ is integer. Observe that $n \leq n_1^2$ (as $\log\log n \leq \log\log n_1 + 1$ and hence $\log n \leq 2\log n_1$), hence $\log\log n = \log\log n_1^2 = O(\log\log n_1) = O(\log\log(n_0+m_0))$. Let us add dummy vertices to G until each side has exactly n vertices. Using the assumed algorithm with running time $2^{2^{2^{o(\log\log n)}}}$, we would be able to solve the Bipartite List 3-Coloring instance and hence the equivalent 3SAT instance in time $2^{2^{2^{o(\log\log(n_0+m_0))}}} = 2^{o(n_0+m_0)}$, contradicting ETH. ◀

Given an instance of Bipartite List 3-Coloring, we construct an equivalent instance of $(1,4)$-Choosability Deletion consisting of a graph G and a list-capacity function f. More precisely, there exists a proper $\mathcal{D}$-coloring $h : X \to \{1,2,3\}$ of H if and only if there exists a subset $U \subseteq V(G)$ of vertices with $|U| \leq 4n$ such that $G - U$ is f-choosable. Moreover, graph G has treewidth $O(\log\log n)$. Thus an algorithm for $(1,4)$-Choosability Deletion on graphs of treewidth w with running time $2^{2^{2^{o(w)}}} \cdot n^{O(1)}$ would give an algorithm for Bipartite List 3-Coloring with running time $2^{2^{2^{o(\log\log n)}}} \cdot n^{O(1)}$, contradicting Lemma 20.

Similarly to Section 3, we reformulate Bipartite List 3-Coloring as a constraint satisfaction problem to improve the presentation: the appearance of colors and lists of colors in both the source and target problems would be a source of confusion. We can view Bipartite List 3-Coloring as a constraint satisfaction problem, where the variable set is X and constraints $y = (y_1, y_2) \in Y \subset X_1 \times X_2$ have arity 2. We call an assignment $h : X \to \{1,2,3\}$ a *legal assignment* if we have $h(x) \in \mathcal{D}(x)$ for every $x \in X$. We say

that H is satisfiable if there exists a legal assignment $h : X \to \{1, 2, 3\}$ such that we have $h(y_1) \neq h(y_2)$ for every $y = (y_1, y_2) \in Y$.

Observe that Choosability Deletion has inherently three levels of quantifier alternations in its definition: $\exists$ (set of deleted vertices) $\forall$ (list assignments $\mathcal{L}$) $\exists$ (choice of colors consistent with $\mathcal{L}$). The main idea of the reduction is that we can redefine Bipartite List 3-Coloring using three levels of quantifier alternations. Furthermore, in order to achieve the triple-exponential lower bound, we need to perform an even tighter compression than the one we achieved in Section 3. Keeping those two things in mind we proceed with re-defining Bipartite List 3-Coloring as follows.

From the original definition of Bipartite List 3-Coloring, we have that H is satisfiable if there exists legal assignment $h : X_1 \cup X_2 \to \{1, 2, 3\}$, such that for all $x_{1i} \in X_1$, $x_{2j} \in X_2$ with $h(x_{1i}) \neq h(x_{2j})$, we have $(x_{1i}, x_{2j}) \notin Y$. The latter requirement $(x_{1i}, x_{2j}) \notin Y$ can be re-written as $\forall y = (x_{1i'}, x_{2j'}) \in Y$, either $i' \neq i$ or $j' \neq j$. For some $i < n$, let $\mathcal{B}(i) = [\mathcal{B}(i)_0, \mathcal{B}(i)_1, \ldots, \mathcal{B}(i)_{\log n - 1}]$ be the binary representation of i. Then $i \neq i'$ can be expressed as saying that there exists a $k \in \{0, \ldots, \log n - 1\}$ such that $\mathcal{B}(i)_k \neq \mathcal{B}(i')_k$.

Putting everything together, we have:

▶ **Definition 21** (CSP Bipartite List 3-Coloring defined with 3 levels of quantifier alternations)**.** Given a set $X = X_1 \cup X_2$ of variables with $X_\xi = \{x_{\xi 0}, x_{\xi 1}, \ldots, x_{\xi(n-1)}\}$ for $\xi \in \{1, 2\}$ and a set $Y \subseteq X_1 \times X_2$ of constraints, the task is to decide if

- $\exists$ legal assignment $h : X_\xi \to \{1, 2, 3\}$, such that
- $\forall x_{1i} \in X_1$, $\forall x_{2j} \in X_2$, $\forall y = (x_{1i'}, x_{2j'}) \in Y$ with $h(x_{1i}) = h(x_{2j})$, we have
- $\exists k \in \{0, \ldots, \log n - 1\}$ such that either $\mathcal{B}(i)_k \neq \mathcal{B}(i')_k$ or $\mathcal{B}(j)_k \neq \mathcal{B}(j')_k$.

The reduction closely follows this equivalent definition of Bipartite List 3-Coloring: we use the three levels of alternation in the definition of $(1, 4)$-Choosability Deletion to express these three levels of alternation. Note also that the last level of alternation, when quantifying over $k \in \{0, \ldots, \log n - 1\}$ can be described as the selection of $\log \log n$ bits. This choice will be expressed by the introduction of $\log \log n$ Bit-choosers, which will be the dominating factor in the treewidth of the constructed instance.

References

1. Mohammad Hossein Bateni, Mohammad Taghi Hajiaghayi, and Dániel Marx. Approximation schemes for Steiner forest on planar graphs and graphs of bounded treewidth. *J. ACM*, 58(5):21, 2011. doi:10.1145/2027216.2027219.
2. M. Biró, Mihály Hujter, and Zsolt Tuza. Precoloring extension. I. interval graphs. *Discrete Mathematics*, 100(1-3):267–279, 1992. doi:10.1016/0012-365X(92)90646-W.
3. Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets Möbius: fast subset convolution. In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 67–74. ACM, 2007. doi:10.1145/1250790.1250801.
4. Ivan Bliznets, Fedor V. Fomin, Marcin Pilipczuk, and Michal Pilipczuk. Subexponential parameterized algorithm for interval completion. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1116–1131. SIAM, 2016. doi:10.1137/1.9781611974331.ch78.
5. Leizhen Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Inf. Process. Lett.*, 58(4):171–176, 1996. doi:10.1016/0020-0190(96)00050-6.
6. Yixin Cao. Linear recognition of almost interval graphs. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms,*

SODA 2016, Arlington, VA, USA, January 10-12, 2016, pages 1096–1115. SIAM, 2016. doi:10.1137/1.9781611974331.ch77.

7 Yixin Cao and Dániel Marx. Chordal editing is fixed-parameter tractable. In Ernst W. Mayr and Natacha Portier, editors, *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014), STACS 2014, March 5-8, 2014, Lyon, France*, volume 25 of *LIPIcs*, pages 214–225. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2014. doi:10.4230/LIPIcs.STACS.2014.214.

8 Yixin Cao and Dániel Marx. Interval deletion is fixed-parameter tractable. *ACM Transactions on Algorithms*, 11(3):21:1–21:35, 2015. doi:10.1145/2629595.

9 Jianer Chen, Fedor V. Fomin, Yang Liu, Songjian Lu, and Yngve Villanger. Improved algorithms for feedback vertex set problems. *J. Comput. Syst. Sci.*, 74(7):1188–1198, 2008. doi:10.1016/j.jcss.2008.05.002.

10 Janka Chlebíková and Klaus Jansen. The d-precoloring problem for k-degenerate graphs. *Discrete Mathematics*, 307(16):2042–2052, 2007. doi:10.1016/j.disc.2005.12.049.

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

12 Marek Cygan, Fedor V. Fomin, 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.

13 Marek Cygan, Dániel Marx, Marcin Pilipczuk, and Michal Pilipczuk. Hitting forbidden subgraphs in graphs of bounded treewidth. In Erzsébet Csuhaj-Varjú, Martin Dietzfelbinger, and Zoltán Ésik, editors, *Mathematical Foundations of Computer Science 2014 – 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part II*, volume 8635 of *Lecture Notes in Computer Science*, pages 189–200. Springer, 2014. doi:10.1007/978-3-662-44465-8_17.

14 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 150–159. IEEE Computer Society, 2011. doi:10.1109/FOCS.2011.23.

15 Frank K. H. A. Dehne, Michael R. Fellows, Michael A. Langston, Frances A. Rosamond, and Kim Stevens. An $o(2^{O(k)}n^3$ FPT algorithm for the undirected feedback vertex set problem. *Theory Comput. Syst.*, 41(3):479–492, 2007. doi:10.1007/s00224-007-1345-z.

16 Paul Erdős, Arthur L Rubin, and Herbert Taylor. Choosability in graphs. *Congr. Numer*, 26:125–157, 1979.

17 Michael R Fellows, Fedor V Fomin, Daniel Lokshtanov, Frances Rosamond, Saket Saurabh, Stefan Szeider, and Carsten Thomassen. On the complexity of some colorful problems parameterized by treewidth. *Information and Computation*, 209(2):143–153, 2011.

18 Fedor V. Fomin and Yngve Villanger. Subexponential parameterized algorithm for minimum fill-in. *SIAM J. Comput.*, 42(6):2197–2216, 2013. doi:10.1137/11085390X.

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

20 Elisabeth Gassner. The Steiner forest problem revisited. *J. Discrete Algorithms*, 8(2):154–163, 2010. doi:10.1016/j.jda.2009.05.002.

21 Shai Gutner and Michael Tarsi. Some results on $(a : b)$-choosability. *Discrete Mathematics*, 309(8):2260–2270, 2009.

22 Ian Holyer. The NP-completeness of edge-coloring. *SIAM Journal on computing*, 10(4):718–720, 1981.

23 M. Hujter and Zs. Tuza. Precoloring extension. II. Graphs classes related to bipartite graphs. *Acta Math. Univ. Comenian. (N.S.)*, 62(1):1–11, 1993.

24 Mihály Hujter and Zsolt Tuza. Precoloring extension III: Classes of perfect graphs. *Combinatorics, Probability & Computing*, 5:35–56, 1996. doi:10.1017/S0963548300001826.

25 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62:367–375, 2001.

26 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

27 Yoichi Iwata, Keigo Oka, and Yuichi Yoshida. Linear-time FPT algorithms via network flow. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1749–1761. SIAM, 2014. doi:10.1137/1.9781611973402.127.

28 Bart M. P. Jansen, Daniel Lokshtanov, and Saket Saurabh. A near-optimal planarization algorithm. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1802–1811. SIAM, 2014. doi:10.1137/1.9781611973402.130.

29 Klaus Jansen and Petra Scheffler. Generalized coloring for tree-like graphs. *Discrete Applied Mathematics*, 75(2):135–155, 1997. doi:10.1016/S0166-218X(96)00085-6.

30 Haim Kaplan, Ron Shamir, and Robert Endre Tarjan. Tractability of parameterized completion problems on chordal, strongly chordal, and proper interval graphs. *SIAM J. Comput.*, 28(5):1906–1922, 1999. doi:10.1137/S0097539796303044.

31 J. Kratochvíl. Precoloring extension with fixed color bound. *Acta Math. Univ. Comenian. (N.S.)*, 62(2):139–153, 1993.

32 Jan Kratochvíl and Zsolt Tuza. Algorithmic complexity of list colorings. *Discrete Applied Mathematics*, 50(3):297–302, 1994. doi:10.1016/0166-218X(94)90150-3.

33 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs on bounded treewidth are probably optimal. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 777–789. SIAM, 2011. doi:10.1137/1.9781611973082.61.

34 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the Exponential Time Hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011. URL: http://albcom.lsi.upc.edu/ojs/index.php/beatcs/article/view/96.

35 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Slightly superexponential parameterized problems. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 760–776. SIAM, 2011. doi:10.1137/1.9781611973082.60.

36 Daniel Lokshtanov, N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Faster parameterized algorithms using linear programming. *ACM Transactions on Algorithms*, 11(2):15:1–15:31, 2014. doi:10.1145/2566616.

37 Dániel Marx. NP-completeness of list coloring and precoloring extension on the edges of planar graphs. *Journal of Graph Theory*, 49(4):313–324, 2005. doi:10.1002/jgt.20085.

38 Dániel Marx. Precoloring extension on unit interval graphs. *Discrete Applied Mathematics*, 154(6):995–1002, 2006. doi:10.1016/j.dam.2005.10.008.

39 Dániel Marx. Chordal deletion is fixed-parameter tractable. *Algorithmica*, 57(4):747–768, 2010. doi:10.1007/s00453-008-9233-8.

40 Dániel Marx and Ildikó Schlotter. Obtaining a planar graph by vertex deletion. *Algorithmica*, 62(3-4):807–822, 2012. doi:10.1007/s00453-010-9484-z.

41 Takao Nishizeki, Jens Vygen, and Xiao Zhou. The edge-disjoint paths problem is NP-complete for series-parallel graphs. *Discrete Applied Mathematics*, 115(1-3):177–186, 2001. doi:10.1016/S0166-218X(01)00223-2.

42 Guoqiang Pan and Moshe Y. Vardi. Fixed-parameter hierarchies inside PSPACE. In *LICS*, pages 27–36. IEEE Computer Society, 2006.

43 Michal Pilipczuk. Problems parameterized by treewidth tractable in single exponential time: A logical approach. In Filip Murlak and Piotr Sankowski, editors, *Mathematical Foundations of Computer Science 2011 – 36th International Symposium, MFCS 2011, Warsaw, Poland, August 22-26, 2011. Proceedings*, volume 6907 of *Lecture Notes in Computer Science*, pages 520–531. Springer, 2011. doi:10.1007/978-3-642-22993-0_47.

44 M. S. Ramanujan and Saket Saurabh. Linear time parameterized algorithms via skew-symmetric multicuts. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1739–1748. SIAM, 2014. doi:10.1137/1.9781611973402.126.

45 Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Oper. Res. Lett.*, 32(4):299–301, 2004. doi:10.1016/j.orl.2003.10.009.

46 Zsolt Tuza. Graph colorings with local constraints – a survey. *Discussiones Mathematicae Graph Theory*, 17(2):161–228, 1997. doi:10.7151/dmgt.1049.

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