Parameterized Orientable Deletion[*]

Tesshu Hanaka[1], Ioannis Katsikarelis[2], Michael Lampis[2], Yota Otachi[3], and Florian Sikora[2]

1 **Department of Economic Engineering, Kyushu University, Fukuoka, Japan**
   `3EC15004S@s.kyushu-u.ac.jp`
2 **Université Paris-Dauphine, PSL Research University, CNRS, UMR LAMSADE, 75016 Paris, France,**
   `(ioannis.katsikarelies|michail.lampis|florian.sikora)@dauphine.fr`
3 **Kumamoto University**
   `otachi@cs.kumamoto-u.ac.jp`

------------- Abstract -------------

A graph is d-orientable if its edges can be oriented so that the maximum in-degree of the resulting digraph is at most d. d-orientability is a well-studied concept with close connections to fundamental graph-theoretic notions and applications as a load balancing problem. In this paper we consider the d-ORIENTABLE DELETION problem: given a graph $G = (V, E)$, delete the minimum number of vertices to make G d-orientable. We contribute a number of results that improve the state of the art on this problem. Specifically:

- We show that the problem is W[2]-hard and $\log n$-inapproximable with respect to k, the number of deleted vertices. This closes the gap in the problem's approximability.
- We completely characterize the parameterized complexity of the problem on chordal graphs: it is FPT parameterized by $d + k$, but W-hard for each of the parameters d, k separately.
- We show that, under the SETH, for all d, ϵ, the problem does not admit a $(d + 2 - \epsilon)^{\mathrm{tw}}$, algorithm where tw is the graph's treewidth, resolving as a special case an open problem on the complexity of PSEUDOFOREST DELETION.
- We show that the problem is W-hard parameterized by the input graph's clique-width. Complementing this, we provide an algorithm running in time $d^{O(d \cdot \mathrm{cw})}$, showing that the problem is FPT by $d + \mathrm{cw}$, and improving the previously best know algorithm for this case.

1 Introduction

In this paper we study the following natural optimization problem: we are given a graph $G = (V, E)$ and an integer d, and are asked to give directions to the edges of E so that in the resulting digraph as many vertices as possible have in-degree at most d. Equivalently, we are looking for an orientation of E such that the set of vertices K whose in-degree is strictly more than d is minimized. Such an orientation is called a d-orientation of $G[V \setminus K]$, and we say that K is a set whose deletion makes the graph d-orientable. The problem of

orienting the edges of an undirected graph so that the in-degree of all, or most, vertices stays below a given threshold has been extensively studied in the literature, in part because of its numerous applications. In particular, one way to view this problem is as a form of scheduling, or load balancing, where edges represent jobs and vertices represent machines. In this case the in-degree represents the load of a machine in a given assignment, and minimizing it is a natural objective (see e.g. [6, 10, 14, 22]). Finding an orientation where all in- or out-degrees are small is also of interest for the design of efficient data structures [11]. For more applications of degree-constrained orientations of graphs we refer the reader to [2, 3, 4, 5, 9] and the references therein.

State of the art. d-orientability has been well-studied in the literature, both because of its practical motivations explained above, but also because it is a basic graph property that generalizes and is closely related to fundamental concepts such as d-degeneracy (as a graph is d-degenerate if and only if it admits an *acyclic d-orientation*), and bounded degree. This places d-ORIENTABLE DELETION in a general context of graph editing problems that measure the distance of a given graph from having one of these properties [7, 17].

Deciding if an unweighted graph is d-orientable is solvable in polynomial time [5], though the problem becomes APX-hard [14] and even W-hard parameterized by treewidth [20] if one allows edge weights. In this paper we focus on unweighted graphs, for which computing the minimum number of vertices that need to be deleted to make a graph d-orientable is easily seen to be NP-hard, as the case $d = 0$ corresponds to VERTEX COVER. This hardness has motivated the study of both polynomial-time approximation and parameterized algorithms, as well as algorithms for specific graph classes. For approximation, if the objective function is to maximize the number of non-deleted vertices, the problem is known to be $n^{1-\epsilon}$-inapproximable; if one seeks to minimize the number of deleted vertices, the problem admits an $O(\log d)$-approximation, but it is not known if this can be improved to a constant [2]. From the parameterized point of view, the problem is W[1]-hard for any fixed d if the parameter is the number of non-deleted vertices [9]. To the best of our knowledge, the complexity of this problem parameterized by the number of deleted vertices is open.

We remark that sometimes in the literature a d-orientation is an orientation where all *out-degrees* are at most d, but this can be seen to be equivalent to our formulation by reversing the direction of all edges. d-ORIENTABLE DELETION has sometimes been called MIN-$(d + 1)$-HEAVY/MAX-d-LIGHT [2], depending on whether one seeks to minimize the number of deleted vertices, or maximize the number of non-deleted vertices (the two are equivalent in the context of exact algorithms). The problem of finding an orientation minimizing the maximum out-degree has also been called MINIMUM MAXIMUM OUT-DEGREE [5].

An important special case that has recently attracted attention from the FPT algorithms point of view is that of $d = 1$. 1-orientable graphs are called pseudo-forests, as they are exactly the graphs where each component contains at most one cycle. 1-ORIENTABLE DELETION, also known as PSEUDOFOREST DELETION, has been shown to admit a 3^k algorithm, where k is the number of vertices to be deleted [8, 19].

Our contribution. We study the complexity of d-ORIENTABLE DELETION mostly from the point of view of exact FPT algorithms. We contribute a number of new results that improve the state of the art and, in some cases, resolve open problems from the literature.

We first consider the parameterized complexity of the problem with respect to the natural parameter k, the number of vertices to be deleted to make the graph d-orientable. We show that for any fixed $d \geqslant 2$, d-ORIENTABLE DELETION is W[2]-hard parameterized by k. This

result is tight in two respects: it shows that, under the ETH, the trivial n^k algorithm that tries all possible solutions is essentially optimal; and it cannot be extended to the case $d = 1$, as in this case the problem is FPT [8]. Because our proof is a reduction from DOMINATING SET that preserves the optimal, we also show that the problem cannot be approximated with a factor better than $\ln n$. This matches the performance of the algorithm given in [2], and closes a gap in the status of this problem, as the previously best known hardness of approximation bound was 1.36 [2].

Second, we consider the complexity of d-ORIENTABLE DELETION when restricted to chordal graphs, motivated by the work of [9], who study the problem on classes of graphs with polynomially many minimal separators. We are able to completely characterize the complexity of the problem for this class of graphs with respect to the two main natural parameters d and k: the problem is W[1]-hard parameterized by d, W[2]-hard parameterized by k, but solvable in time roughly $d^{O(d+k)}$, and hence FPT when parameterized by $d + k$. We recall that the problem is poly-time solvable on chordal graphs when d is a constant [9], and trivially in P in general graphs when k is a constant, so these results are in a sense tight.

Third, we consider the complexity of d-ORIENTABLE DELETION parameterized by the input graph's treewidth, perhaps the most widely studied graph parameter. Our main contribution here is a lower bound which, assuming the Strong ETH, states that the problem cannot be solved in time less than $(d + 2)^{\mathrm{tw}}$, for any constant $d \geqslant 1$. As a consequence, this shows that the 3^{tw} algorithm given for PSEUDOFOREST DELETION in [8] is optimal under the SETH. We recall that Bodlaender et al. [8] had explicitly posed the existence of a better treewidth-based algorithm as an open problem; our results settle this question in the negative, assuming the SETH. Our result also extends the lower bound of [16] which showed that VERTEX COVER (which corresponds to $d = 0$) cannot be solved in $(2 - \epsilon)^{\mathrm{tw}}$.

Finally, we consider the complexity of the problem parameterized by clique-width. We recall that clique-width is probably the second most widely studied graph parameter in FPT algorithms (after treewidth), so after having settled the complexity of d-ORIENTABLE DELETION with respect to treewidth, investigating clique-width is a natural question. On the positive side, we present a dynamic programming algorithm whose complexity is roughly $d^{O(d \cdot \mathrm{cw})}$, and is therefore FPT when parameterized by $d + \mathrm{cw}$. This significantly improves upon the dynamic programming algorithm for this case given in [9], which runs in time roughly $n^{O(d \cdot \mathrm{cw})}$. The main new idea of this algorithm, leading to its improved performance, is the observation that sufficiently large entries of the DP table can be merged using a more careful characterization of feasible solutions that involve large bi-cliques. On the negative side, we present a reduction showing that d-ORIENTABLE DELETION is W[1]-hard if cw is the only parameter. This presents an interesting contrast with the case of treewidth: for both parameters we can obtain algorithms whose running time is a function of d and the width; however, because graphs of treewidth w always admit a w-orientation (since they are w-degenerate), this immediately also shows that the problem is FPT for treewidth, while our results imply that obtaining a similar result for clique-width is impossible (under standard assumptions). Due to space constraints, some proofs (marked with a ★) are in the appendix.

2 Definitions and Preliminaries

Complexity background. We assume that the reader is familiar with the basic definitions of parameterized complexity, such as the classes FPT and W[1] [13]. We will also make use of the *Exponential Time Hypothesis* (ETH), a conjecture by Impagliazzo et al. asserting that there is no $2^{o(n)}$-time algorithm for 3-SAT on instances with n variables [15]. The Strong

Exponential Time Hypothesis (SETH) states that SAT cannot be solved in time $O^*((2-\epsilon)^n)$ for any $\epsilon > 0$ [15].

Graph widths. We also make use of standard graph width measures, such as pathwidth, treewidth, and clique-width, denoted as $\mathrm{pw}, \mathrm{tw}, \mathrm{cw}$ respectively. For the definitions we refer the reader to standard textbooks [13, 12]. We recall the following standard relations:

▶ **Lemma 1.** *For all graphs $G = (V, E)$ we have $\mathrm{tw}(G) \leqslant \mathrm{pw}(G)$ and $\mathrm{cw}(G) \leqslant \mathrm{pw}(G) + 2$.*

Graphs and Orientability. We use standard graph-theoretic notation. If $G = (V, E)$ is a graph and $S \subseteq V$, $G[S]$ denotes the subgraph of G induced by S. For $v \in V$, the set of neighbors of v in G is denoted by $N_G(v)$, or simply $N(v)$, and $N_G(S) := (\bigcup_{v \in S} N(v)) \setminus S$ will often be written just $N(S)$. We define $N[v] := N(v) \cup \{v\}$ and $N[S] := N(S) \cup S$. Depending on the context, we use (u, v), where $u, v \in V$ to denote either an undirected edge connecting two vertices u, v, or an arc (that is, a directed edge) with tail u and head v. An *orientation* of an undirected graph $G = (V, E)$ is a directed graph on the same set of vertices obtained by replacing each undirected edge $(u, v) \in E$ with either the arc (u, v) or the arc (v, u). In a directed graph we define the in-degree $\delta^-(v)$ of a vertex u as the number of arcs whose head is u. A d-orientation of a graph $G = (V, E)$ is an orientation of G such that all vertices have in-degree at most d. If such an orientation exists, we say that G is d-orientable. Deciding if a given graph is d-orientable is solvable in polynomial time, even if d is part of the input [5]. Let us first make some easy observations on the d-orientability of some basic graphs.

▶ **Lemma 2 (★).** *K_{2d+1}, the clique on $2d+1$ vertices, is d-orientable. Furthermore, in any d-orientation of K_{2d+1} all vertices have in-degree d.*

▶ **Lemma 3 (★).** *The complete bipartite graph $K_{2d+1,2d}$ is not d-orientable.*

▶ **Definition 4.** In d-ORIENTABLE DELETION we are given as input a graph $G = (V, E)$ and an integer d. We are asked to determine the smallest set of vertices $K \subseteq V$ (the *deletion set*) such that $G[V \setminus K]$ admits a d-orientation.

▶ **Definition 5.** In CAPACITATED-d-ORIENTABLE DELETION we are given as input a graph $G = (V, E)$, an integer $d \geqslant 1$, and a *capacity* function $\mathbf{c} : V \to \{0, \ldots, d\}$. We are asked to determine the smallest set of vertices $K \subseteq V$ such that $G[V \setminus K]$ admits an orientation with the property that for all $u \in V \setminus K$, the in-degree of u is at most $\mathbf{c}(u)$.

It is clear that CAPACITATED-d-ORIENTABLE DELETION generalizes d-ORIENTABLE DELETION, which corresponds to the case where we have $\mathbf{c}(u) = d$ for all vertices. It is, however, not hard to see that the two problems are in fact equivalent, as shown in the following lemma. Furthermore, the following lemma shows that increasing d can only make the problem harder.

▶ **Lemma 6.** *There exists a polynomial-time algorithm which, given an instance $[G = (V, E), d, \mathbf{c}]$ of CAPACITATED-d-ORIENTABLE DELETION, and an integer $d' \geqslant d$, produces an equivalent instance $[G' = (V', E'), d']$ of d-ORIENTABLE DELETION, with the same optimal value and the following properties: $\mathrm{pw}(G') \leqslant \mathrm{pw}(G) + 2d' + 1$, $\mathrm{cw}(G') \leqslant \mathrm{cw}(G) + 4$, and if G is chordal then G' is chordal.*

(Sketch). The idea of the reduction is to construct for every $u \in V$ for which $\mathbf{c}(u) < d'$ a clique of $K_{2d'+1}$ vertices and connect $d' - \mathbf{c}(u)$ vertices of the clique to u (see Figure 1). Lemma 2 ensures that the edges connecting the clique to u will be oriented towards u, simulating its decreased capacity. We can then argue that in the new instance there always exists an optimal solution that does not delete any of the new vertices, therefore optimal solutions are preserved. ◀
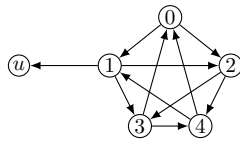
Figure 1 A 2-orientation of a clique K_5. Observe that any edge connecting a vertex u to the clique must be oriented towards u (setting its *capacity*) to maintain a 2-orientation.
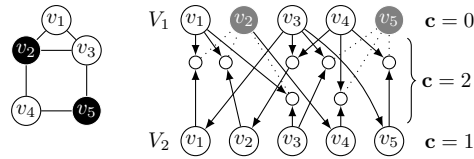
Figure 2 Left: A graph with its dominating set in black. **Right:** The corresponding instance and 2-orientation, where deleted vertices are in gray. Original edges with a deleted vertex as an endpoint are dotted.

3   Hardness of Approximation and W-hardness

In this section we present a reduction from DOMINATING SET to d-ORIENTABLE DELETION for $d \geqslant 2$ that exactly preserves the size of the solution. As a result, this establishes that, for any fixed $d \geqslant 2$, d-ORIENTABLE DELETION is W[2]-hard, and the minimum solution cannot be approximated with a better than logarithmic factor. We observe that it is natural that our reduction only works for $d \geqslant 2$, as the problem is known to be FPT for $d = 1$, which is known as PSEUDOFOREST DELETION, and $d = 0$, which is equivalent to VERTEX COVER.

▶ **Theorem 7.** *For any $d \geqslant 2$, d-ORIENTABLE DELETION is W[2]-hard parameterized by the solution size k. Furthermore, for any $d \geqslant 2$, d-ORIENTABLE DELETION cannot be solved in time $n^{o(k)}$, unless the ETH is false.*

Proof. We will describe a reduction from Dominating Set, which is well-known to be W[2]-hard and not solvable in $n^{o(k)}$ under the ETH, to CAPACITATED-d-ORIENTABLE DELETION for $d = 2$. We will then invoke Lemma 6 to obtain the claimed result for d-ORIENTABLE DELETION. Let $[G(V, E), k]$ be an instance of Dominating Set. We begin by constructing a bipartite graph by taking two copies of V, call them V_1, V_2. For each $v \in V_2$ we construct a binary tree with $|N_G[v]|$ leaves. We identify the root of this binary tree with $v \in V_2$ and its leaves with the corresponding vertices in V_1. We now define the capacities of our vertices: each vertex of V_1 has capacity 0; each internal vertex of the binary trees has capacity 2; and each vertex of V_2 has capacity 1.

We will now claim that G has a dominating set of size k if and only if the new graph can be oriented in a way that respects the capacities by deleting at most k vertices.

For the forward direction, suppose that there is a dominating set in G of size k. In the new graph we delete the corresponding vertices of V_1. We argue that the remaining graph is orientable in a way that respects the capacities. We compute an orientation as follows:

1. We orient the remaining incident edges away from every vertex of V_1 that is not deleted.
2. For each non-leaf vertex u of the binary tree rooted at $v \in V_2$ we define the orientation of the edge connecting u to its parent as follows: u is an ancestor of a set $S_u \subseteq N_G[v]$ of vertices of V_1. If S_u contains a deleted vertex, then we orient the edge connecting u to its parent towards u, otherwise we orient it towards u's parent.

The above description completely defines the orientation of the remaining graph (see also Figure 2). Let us argue why the orientation respects all capacities. This should be clear for vertices of V_1. For any non-leaf vertex u of a binary tree, if we orient the edge connecting it to its parent away from u, then the in-degree of u is at most 2, which is its capacity. On the other hand, if we orient this edge towards u, there is a deleted vertex in S_u. However, this

implies either that one of u's children has been deleted, or that one of the edges connecting u to one of its children is oriented away from u. In both cases, the in-degree of u is at most 2, equal to its capacity. Finally, for each $u \in V_2$, if we started with a dominating set, then one of the children of u in the binary tree is either deleted or its edge to u is oriented towards it.

For the converse direction, suppose that there is a set of k vertices in the new graph whose deletion makes the graph orientable in a way that respects the capacities. Suppose now that we have a solution that deletes some vertex $v \in V_2$ or some internal vertex of a binary tree. We re-introduce v in the graph, orient all its incident edges towards v, and then delete one of the children of v. This preserves the size and validity of the solution. Repeating this argument ends with a solution that only deletes vertices of V_1. We now claim that these k vertices are a dominating set. To see this, observe that any undeleted vertex of V_1 has all its edges connecting it to binary trees oriented away from it. Hence, if there is a binary tree with root $v \in V_2$ such that none of its leaves are undeleted, all its internal edges must be oriented towards v, which would make the in-degree of v greater than its capacity.   ◀

▶ **Corollary 8 (★).** *For any $d \geqslant 2$, if there exists a polynomial-time $o(\log n)$-approximation for d-ORIENTABLE DELETION, then P=NP.*

4   Chordal Graphs

In this section we consider the complexity of d-ORIENTABLE DELETION on chordal graphs parameterized by either d or k (the number of deleted vertices). Our main results state that the problem is W-hard for each of these parameters individually (Theorems 9 and 10); however, the problem is FPT parameterized by $d + k$ (Theorem 11).

▶ **Theorem 9 (★).** *d-ORIENTABLE DELETION is W[1]-hard on chordal graphs parameterized by d. Furthermore, it cannot be solved in $n^{o(d)}$ under the ETH.*

▶ **Theorem 10.** *d-ORIENTABLE DELETION is W[2]-hard on chordal graphs parameterized by the solution size k. Furthermore, under the ETH it cannot be solved in time $n^{o(k)}$.*

Proof. We start from an instance of DOMINATING SET: we are given a graph $G = (V, E)$ and an integer k and are asked if there exists a dominating set of size k. We will retain the same value of k and construct a chordal instance of CAPACITATED-d-ORIENTABLE DELETION, for which we later invoke Lemma 6. Let $|V| = n$ and we assume without loss of generality that $n - k$ is odd (otherwise we can add an isolated edge to G). We construct G' as follows. Take two copies of V, call them V_1, V_2 and add all possible edges between vertices of V_2. For each $u \in V$, we connect $u \in V_1$ with all vertices $v \in V_2$ such that $v \in N_G[u]$. Let us also define the capacities: each $u \in V_1$ has capacity $d_G(u)$; each $u \in V_2$ has capacity $\frac{n-k-1}{2}$. This completes the construction. G' is chordal because it is a split graph.

Suppose that G has a dominating set of size k. We delete the same vertices of V_2 and claim that G' becomes orientable. We observe that all vertices of V_1 have at least a deleted neighbor, since we deleted a dominating set of G, hence for each such vertex the number of remaining incident edges is at most its capacity. We therefore orient all edges incident on V_1 towards V_1. Finally, for the remaining vertices of V_2 which induce a clique of size $n - k$ we orient their edges using Lemma 2 so that they all have in-degree exactly $\frac{n-k-1}{2}$.

For the converse direction, suppose we can delete at most k vertices of the new graph to make it orientable respecting the capacities. Again, as in Theorem 9 we assume we have a solution of size exactly k, otherwise we add some vertices. Furthermore, any used vertex of V_1 can be exchanged with one of its neighbors in V_2, since all vertices of V_1 have degree

one more than their capacities, hence we assume that the solution deletes k vertices of V_2. We show that these vertices are a dominating set of G. Suppose for contradiction that they are not, so $u \in V_1$ does not have any deleted neighbors in V_2. Since there are $d(u) + 1$ edges connecting $u \in V_1$ to V_2, at least one of them is oriented towards V_2. But now the $n - k$ non-deleted vertices of V_2, because of Lemma 2 all have in-degree exactly equal to the capacities inside the clique they induce. Hence, the additional edge from V_1 will force a vertex to violate its capacity. ◀

▶ **Theorem 11** (★). d-ORIENTABLE DELETION *can be solved in time* $d^{O(d+k)}n^{O(1)}$ *on chordal graphs, where k is the size of the solution.*

5 SETH Lower Bound for Treewidth

Overview. We follow the approach for proving SETH lower bounds for treewidth algorithms introduced in [16] (see also Chapter 14 in [13]), that is, we present a reduction from SAT to d-ORIENTABLE DELETION showing that if there exists a better than $(d + 2)^{\text{tw}}$ algorithm for d-ORIENTABLE DELETION, we obtain a better than 2^n algorithm for SAT.

Similarly to these proofs, our reduction is based on the construction of "long paths" of *Block gadgets*, that are serially connected in a path-like manner. Each such "path" corresponds to a group of variables of the given formula, while each *column* of this construction is associated with one of its clauses. Intuitively, our aim is to embed the 2^n possible variable assignments into the $(d + 2)^{\text{tw}}$ states of some optimal dynamic program that would solve the problem on our constructed instance. The hard part of the reduction is to take the natural $d + 2$ options available for each vertex, corresponding to its in-degree $(d + 1)$ or the choice to delete it $(+1)$, and use them to compress n boolean variables into roughly $\frac{n}{\log(d+2)}$ units of treewidth.

Below, we present a sequence of gadgets used in our reduction. The aforementioned block gadgets, which allow a solution to choose among $d + 2$ reasonable choices, are the main ingredient. We connect these gadgets in a path-like manner that ensures that choices remain consistent throughout the construction, and connect clause gadgets in different "columns" of the constructed grid in a way that allows us to verify if the choice made represents a satisfying assignment, without increasing the graph's treewidth.

OR gadget. We use an OR gadget with two endpoint v, u whose purpose is to ensure that in any optimal solution, either v or u will have to be deleted. This gadget is simply a set of $2d + 2$ vertices of capacity 1, connected to both v and u, as shown in Figure 3.
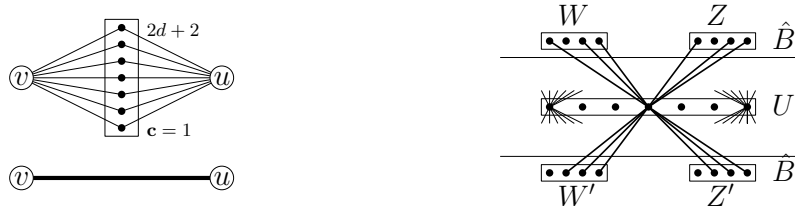
Figure 3 Left: An example OR gadget. In the following, OR gadgets are shown as bold edges. **Right:** Example connections between a set U and the p sets W, Z in the gadgets $\hat{B}$ of its group.

Clause gadget $\hat{C}(N)$. This gadget is identical to the one used for INDEPENDENT SET in [16] (where all vertices are of capacity 0), as finding a maximum independent set can be seen as equivalent to finding a minimum-sized deletion set for 0-orientability. Due to space

restrictions, its construction and proof of correctness are omitted here (see Appendix C). The gadget has N *input* vertices and its purpose is to offer an 1-in-N choice, while its pathwidth remains constant. The non-deleted input will correspond to a true literal within the clause.

Block gadget $\hat{B}$.   This gadget is the basic building block of our construction:
1. Make three vertices a, a', b. Note that in the final construction, our block gadgets will be connected serially, with vertex a' being identified with the following gadget's vertex a.
2. Make three independent sets $X := \{x_1, \ldots, x_d\}$, $Y := \{y_1, \ldots, y_d\}$, $Q := \{q_1, \ldots, q_{2d+1}\}$.
3. Make two sets $W := \{w_0, \ldots, w_{d+1}\}$ and $Z := \{z_0, \ldots, z_{d+1}\}$.
4. Connect all vertices of X with vertex a and with all vertices of Q.
5. Connect all vertices of Y with vertex a' and with all vertices of Q.
6. Connect all vertices from W except w_{d+1} to b and all vertices of X.
7. Connect all vertices from Z except z_{d+1} to b and all vertices of Y.
8. Attach OR gadgets between the pairs: a and b, b and a', a and w_{d+1}, a' and z_{d+1}.
9. Attach OR gadgets between any pair of vertices in $W \cup Z$, except for the pairs (w_i, z_i) for $i \in \{0, \ldots, d+1\}$. In other words, $W \cup Z$ is an OR-clique, minus a perfect matching.

We set the capacities as follows (see also Figure 4).

- $\mathbf{c}(a) = \mathbf{c}(a') = d$, $\mathbf{c}(b) = 0$.
- $\mathbf{c}(q_i) = d, \forall i \in [1, 2d+1]$, $\mathbf{c}(x_i) = \mathbf{c}(y_j) = 0, \forall i, j \in [1, d]$.
- $\mathbf{c}(w_i) = i, \mathbf{c}(z_i) = d - i, \forall i \in [0, d]$, $\mathbf{c}(w_{d+1}) = \mathbf{c}(z_{d+1}) = 0$.

Intuitively, there are $d + 2$ options in each gadget, linked to the circumstances of vertices a, a' and b:[1] there will have to be d vertices deleted in total from $X \cup Y$ and the numbers will be complementary: if i vertices remain in X then, due to Q being of size $2d + 1$ (it is never useful to delete any of them), there must be $d - i$ vertices remaining in Y. Thus, the $d + 1$ options can be seen as represented by the number of vertices remaining in X, while for each one, vertex b must also be deleted due to the OR gadgets connecting it to a, a'. The extra option is to ignore the actual number of deletions within X and remove both a, a' instead.

The sets W, Z are connected in such a way that any reasonable feasible solution will delete all of their vertices, except for a pair w_i, z_i for some $i \in \{0, \ldots, d+1\}$. The non-deleted pair is meant to encode a choice for this block gadget.

Global construction.   Fix some integer d, and suppose that for some $\epsilon > 0$ there exists a $(d+2-\epsilon)^{\text{tw}}$ algorithm for d-ORIENTABLE DELETION. We give a reduction which, starting from any SAT instance with n variables and m clauses, produces an instance of d-ORIENTABLE DELETION, such that applying this supposed algorithm on the new instance would give a better than 2^n algorithm for SAT.

We are faced with the problem that $d + 2$ is not a power of 2, hence we will need to create a correspondence between groups of variables of the SAT instance and groups of block gadgets. We first choose an integer $p = \lceil \frac{1}{(1-\lambda)\log_2(d+2)} \rceil$, for $\lambda = \log_{d+2}(d + 2 - \epsilon) < 1$. We then group the variables of ϕ into $t = \lceil \frac{n}{\gamma} \rceil$ groups $F_1, \ldots, F_t$, where $\gamma = \lfloor \log_2(d+2)^p \rfloor$ is the maximum size of each group. Our construction then proceeds as follows (see Figure 5):

---

[1] Each such option can be seen to correspond with one of the states that some optimal dynamic programming algorithm for the problem would assign to vertex a: it is either deleted, or has a number $i \in [0, d]$ of incoming edges within the gadget.
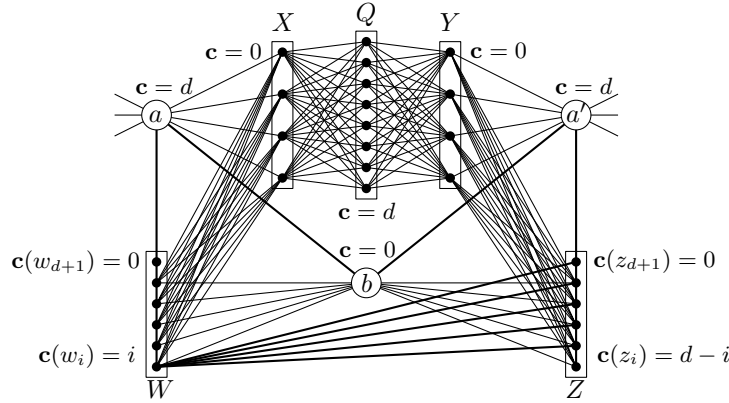
Figure 4 Our block gadget $\hat{B}$. Capacities are shown next to vertices/sets, while the connections between W and Z are only shown for w_0.

1. Make a group of p block gadgets $\hat{B}_\tau^{1,\pi}$ for $\pi \in [1, p]$, for each group F_τ of variables of ϕ with $\tau \in [1, t]$.

2. Make a clique $U_\tau^1 := \{u_\tau^{1,1}, \ldots, u_\tau^{1,(d+2)^p}\}$ on $(d+2)^p$ vertices, whose capacities are all set to 0, for each group F_τ of variables of ϕ with $\tau \in [1, t]$.

3. Associate each of these $(d+2)^p$ vertices from U_τ^1 with one of the $d+2$ options *for each* gadget $\hat{B}_\tau^{1,\pi}$, i.e. over all $\pi \in [1, p]$.

4. Connect each $u_\tau^{1,i}$ for $i \in [1, (d+2)^p]$ to each vertex from each W and Z within each of the p gadgets $\hat{B}$ that *do not match* the option associated with $u_\tau^{1,i}$ via OR gadgets (see Figure 3 for an example).

5. Make $m(tpd + 2)$ copies of this first "column" of gadgets.

6. Identify each vertex a' in $\hat{B}_\tau^{l,\pi}$ with the vertex a of its following gadget $\hat{B}_\tau^{l+1,\pi}$, i.e. for fixed $\tau \in [1, t]$ and $\pi \in [1, p]$, all block gadgets are connected in a path-like manner.

7. For every clause C_μ, with $\mu \in [1, m]$, make a clause gadget $\hat{C}_\mu^o$ with $N = q_\mu$ inputs, where q_μ is the number of literals[2] in clause C_μ and $o \in [0, tpd + 1]$.

8. For every $\tau \in [1, t]$, associate one of the $(d+2)^p$ vertices of U_τ^l (that is in turn associated with one of $d+2$ options for each of the p gadgets of group F_τ), with an assignment to the variables in group F_τ. Note that as there are at most $2^\gamma = 2^{\lfloor \log_2(d+2)^p \rfloor}$ assignments to the variables in F_τ and $(d+2)^p \geqslant 2^\gamma$ such vertices, the association can be unique for each τ (and the same for all $l \in [1, m(tpd + 2)]$).

9. Each of the clause gadget's q_μ inputs will correspond to a literal appearing in clause C_μ.

10. Connect via OR gadgets each input from each $\hat{C}_\mu^o$, corresponding to a literal whose variable appears in group F_τ, to the all vertices from the set $U_\tau^{mo+\mu}$ (in its appropriate column) whose associated assignments *do not satisfy* the input's literal.

▶ **Theorem 12 (★).** *For any fixed $d \geqslant 1$, if d-ORIENTABLE DELETION can be solved in $O^*((d + 2 - \epsilon)^{tw(G)})$ time for some $\epsilon > 0$, then there exists some $\delta > 0$, such that SAT can be solved in $O^*((2 - \delta)^n)$ time.*

▶ **Corollary 13.** *If PSEUDOFOREST DELETION can be solved in $O^*((3 - \epsilon)^{tw(G)})$ time for some $\epsilon > 0$, then there exists some $\delta > 0$, such that SAT can be solved in $O^*((2 - \delta)^n)$ time.*

---

[2] We assume that q_μ is always even, by duplicating some literals if necessary.
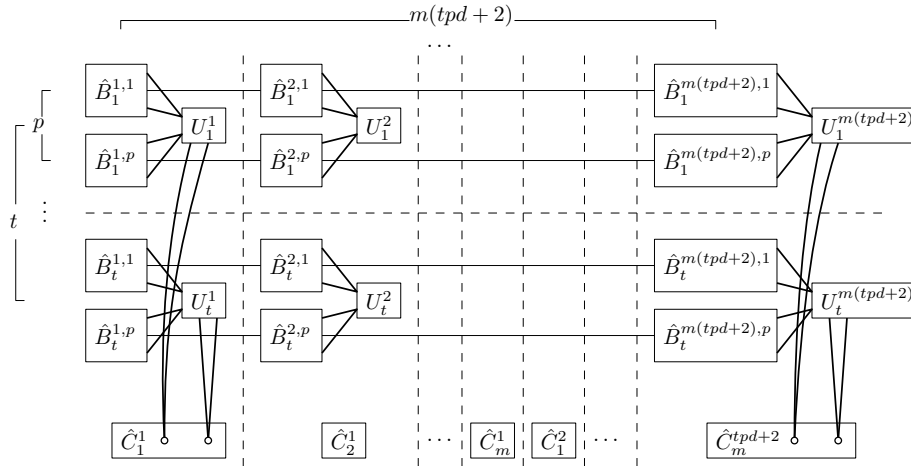
Figure 5 A simplified picture of the complete construction.

6 Algorithm for Clique-Width

In this section we present a dynamic programming algorithm for d-ORIENTABLE DELETION parameterized by the clique-width of the input graph, of running time $d^{O(d \cdot \text{cw})}$. The algorithm is based on the dynamic programming of [9] for MAX W-LIGHT, the problem of assigning a direction to each edge of an undirected graph so that the number of vertices of out-degree at most W is maximized. As noted in [9] (and our Section 1), that problem is supplementary to MIN $(W + 1)$-HEAVY, the problem of minimizing the number of vertices of out-degree at least $W + 1$ in terms of exact computation (though their approximability properties may vary), that in turn can be seen as the optimization version of d-ORIENTABLE DELETION for $d = W$, if we simply consider the in-degree of every vertex instead of the out-degree (by reversing the direction of every edge in any given orientation).

The dynamic programming algorithm of [9] runs in XP-time $n^{O(d \cdot \text{cw})}$, by considering the full number of possible states for each label of a clique-width expression T for the input graph G:[3] for each node t of T, it computes an *in-degree-signature* of G_t, being a table $A_t = (A_t^{i,j}), \forall i \in [1, \text{cw}], j \in [0, d]$, if there is an orientation Λ_t (of every edge of G_t) such that for each label $i \in [1, \text{cw}]$ and *in-degree-class* $j \in [0, d]$, the entry $A_t^{i,j}$ is the number of vertices labelled i with in-degree j in G_t under Λ_t, and also a *deletion set* K_t, where $K_t := \bigcup_{i \in [1, \text{cw}]} K_t^i$ for each $i \in [1, \text{cw}]$, where K_t^i is the set of vertices labelled i that are deleted from G_t. Based on this scheme, the updating process of the tables is straightforward for Leaf, Relabel and Union nodes, while for Join nodes, the computation of the degree signatures is based on a result by [1], stating that an orientation satisfying any given lower and upper in-degree bounds for each vertex can be computed in $O(m^{3/2} \log n)$ time, where m is the number of edges to be oriented. We refer to [9] for details.

The aim of this section is to improve the running time of the above algorithm to $d^{O(d \cdot \text{cw})}$, that is FPT-time parameterized by d and cw, by showing that not all of the natural states utilized therein are in fact required. The main idea behind this improvement is based on the redundancy of *exactly* keeping track of the size of an in-degree-class above a certain threshold (i.e. d^4), since the valid d-orientations of a biclique created after joining such an

---

[3] Slightly paraphrased here for d-ORIENTABLE DELETION, keeping the same notation.

in-degree-class with some other label are greatly constrained, as any optimal solution will always orient all new edges towards the vertices of this "large" class in order to maintain d-orientability (and update in-degree-class sizes accordingly), while respecting the given deletion set and orientation of previously introduced edges.

▶ **Theorem 14 (★).** *Given a graph G along with a cw-expression T of G, the d-ORIENTABLE DELETION problem can be solved in time $O^*(d^{O(d \cdot cw)})$.*

7   W-hardness for Clique-Width

In this section we present a reduction establishing that the algorithm of Section 6 is essentially optimal. More precisely, we show that, under the ETH, no algorithm can solve d-ORIENTABLE DELETION in time $n^{o(cw)}$. As a result, the parameter dependence of $d^{O(cw)}$ of the algorithm in Section 6 cannot be improved to a function that only depends on cw. We prove this result through a reduction from k-MULTICOLORED INDEPENDENT SET.

Construction.   Recall that an instance $[G = (V, E), k]$ of k-MULTICOLORED INDEPENDENT SET consists of a graph G whose vertex set is given to us partitioned into k sets $V_1, \ldots V_k$, with $|V_i| = n$ for all $i \in [1, k]$, and with each V_i inducing a clique. Given such an instance, we will construct an instance $G' = (V', E')$ of d-ORIENTABLE DELETION, where $d = n$. Let $V_i := \{v_i^1, \ldots, v_i^n\}, \forall i \in [1, k]$. To simplify notation, we use E to denote the set of *non-clique* edges, i.e. those connecting vertices in parts V_i, V_j for $i \neq j$. Our construction is given as follows, while Figure 6 provides an illustration:

1. Create two sets $A_i, B_i \subset V', \forall i \in [1, k]$ of n vertices each, of capacities 0.
2. Make a set of *guard* vertices $W_i, \forall i \in [1, k]$, of size $kn + 3|E| + 1$, of capacities n.
3. Connect each vertex of W_i to all vertices of A_i, B_i for all $i \in [1, k]$.
4. For each edge $e = (v_i^l, v_j^o) \in E$ with endpoints $v_i^l \in V_i, v_j^o \in V_j$ (i.e. the l-th vertex of V_i and the o-th vertex of V_j), make four new vertices $a_l^e, b_l^e, a_o^e, b_o^e$.
5. Connect $a_l^e, b_l^e, a_o^e, b_o^e$ to each other via OR gadgets.
6. Connect a_l^e to all vertices of A_i and b_l^e to all vertices of B_i, while a_o^e is connected to all vertices of A_j and b_o^e to all vertices of B_j.
7. Set the capacities $\mathbf{c}(a_l^e) = n - l - 1$, $\mathbf{c}(b_l^e) = l - 1$, $\mathbf{c}(a_o^e) = n - o - 1$ and $\mathbf{c}(b_o^e) = o - 1$.
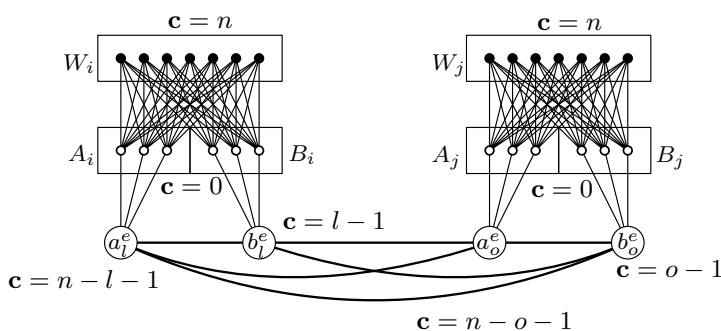
Figure 6 A partial view of the construction, depicting the gadgets encoding the selection for V_i, V_j, as well the representation of an edge $e = (v_i^l, v_j^o)$. Note bold edges signifying OR gadgets.

▶ **Theorem 15** (★). *d*-ORIENTABLE DELETION *is W[1]-hard parameterized by the clique-width of the input graph. Furthermore, if there exists an algorithm solving d*-ORIENTABLE DELETION *in time* $n^{o(cw)}$ *then the ETH is false.*

── **References** ──

1 Y. Asahiro, J. Jansson, E. Miyano, and H. Ono. Upper and lower degree bounded graph orientation with minimum penalty. In *CATS'12*, volume 128, pages 139–145, 2012.

2 Y. Asahiro, J. Jansson, E. Miyano, and H. Ono. Degree-Constrained Graph Orientation: Maximum Satisfaction and Minimum Violation. *Theory Comput. Syst.*, 58(1):60–93, 2016.

3 Y. Asahiro, J. Jansson, E. Miyano, H. Ono, and K. Zenmyo. Approximation algorithms for the graph orientation minimizing the maximum weighted outdegree. *J. Comb. Optim.*, 22(1):78–96, 2011.

4 Y. Asahiro, E. Miyano, and H. Ono. Graph classes and the complexity of the graph orientation minimizing the maximum weighted outdegree. *D.A.M.*, 159(7):498–508, 2011.

5 Y. Asahiro, E. Miyano, H. Ono, and K. Zenmyo. Graph orientation algorithms to minimize the maximum outdegree. *Int. J. Found. Comput. Sci.*, 18(2):197–215, 2007.

6 M. Bateni, M. Charikar, and V. Guruswami. Maxmin allocation via degree lower-bounded arborescences. In *STOC*, pages 543–552. ACM, 2009.

7 N. Betzler, R. Bredereck, R. Niedermeier, and J. Uhlmann. On bounded-degree vertex deletion parameterized by treewidth. *Discrete Applied Mathematics*, 160(1-2):53–60, 2012.

8 H. L. Bodlaender, H. Ono, and Y. Otachi. A faster parameterized algorithm for pseudoforest deletion. In *IPEC*, volume 63, pages 7:1–7:12, 2016.

9 H. L. Bodlaender, H. Ono, and Y. Otachi. Degree-constrained orientation of maximum satisfaction: Graph classes and parameterized complexity. *Algorithmica*, Dec 2017.

10 D. Chakrabarty, J. Chuzhoy, and S. Khanna. On allocating goods to maximize fairness. In *FOCS*, pages 107–116. IEEE Computer Society, 2009.

11 M. Chrobak and D. Eppstein. Planar orientations with low out-degree and compaction of adjacency matrices. *Theor. Comput. Sci.*, 86(2):243–266, 1991.

12 B. Courcelle and J. Engelfriet. *Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach*, volume 138 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2012.

13 M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.

14 T. Ebenlendr, M. Krcál, and J. Sgall. Graph balancing: A special case of scheduling unrelated parallel machines. *Algorithmica*, 68(1):62–80, 2014.

15 R. Impagliazzo, R. Paturi, and F. Zane. Which Problems Have Strongly Exponential Complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

16 D. Lokshtanov, D. Marx, and S. Saurabh. Known algorithms on graphs on bounded treewidth are probably optimal. In *SODA*, pages 777–789, 2011.

17 L. Mathieson. The parameterized complexity of editing graphs for bounded degeneracy. *Theor. Comput. Sci.*, 411(34-36):3181–3187, 2010.

18 D. Moshkovitz. The Projection Games Conjecture and the NP-Hardness of ln n-Approximating Set-Cover. *Theory of Computing*, 11:221–235, 2015.

19 G. Philip, A. Rai, and S. Saurabh. Generalized pseudoforest deletion: Algorithms and uniform kernel. In *MFCS (2)*, volume 9235 of *LNCS*, pages 517–528, 2015.

20 S. Szeider. Not so easy problems for tree decomposable graphs. *CoRR,abs/1107.1177*, 2011.

21 A. Takahashi, S. Ueno, and Y. Kajitani. Mixed searching and proper-path-width. *Theoretical Computer Science*, 137(2):253–268, 1995.

22 J. Verschae and A. Wiese. On the configuration-lp for scheduling on unrelated machines. *J. Scheduling*, 17(4):371–383, 2014.

A   Omitted Material from Section 2

Proof of Lemma 2

Proof. We show that there is a way to orient all the edges of any K_{2d+1} so that all vertices have in-degree exactly d. To see this, number the vertices $\{0, 1, \ldots, 2d\}$, and then for each $i \in \{0, \ldots, 2d\}$ orient away from vertex i all edges whose other endpoint is $\{i+1, \ldots, i+d\}$, where addition is done modulo $2d + 1$. Observe that this defines the orientation of all edges, and for each vertex it orients away from it d of its $2d$ incident edges. Hence, all vertices have in-degree d in the end (see also Figure 1).

For the second part, observe that K_{2d+1} has $d(2d + 1)$ edges and $2d + 1$ vertices, hence in any orientation the average in-degree must be exactly d. In an orientation where the maximum in-degree is d we therefore also have that the minimum in-degree is also d.   ◄

Proof of Lemma 3

Proof. We observe that in any orientation the sum of all in-degrees is equal to the total number of edges, which is $4d^2 + 2d$. Hence, the average in-degree is $> d$, meaning there will always be at least one vertex of in-degree $> d$.   ◄

Proof of Lemma 6

Proof. We use a saturation gadget to simulate the fact that some vertices of G are meant to be able to accept strictly fewer than d incoming edges. In particular, for each $u \in V$ for which $\mathbf{c}(u) < d'$ we construct a clique on $K_{2d'+1}$ new vertices and connect $d' - \mathbf{c}(u)$ of these vertices (arbitrarily chosen) to u.

Let us argue that this produces an equivalent instance. First, consider a solution to the original instance. We delete the same set of vertices from G', and use the same orientation for edges of E. For the added edges, there is a way to orient all the edges of any $K_{2d'+1}$ by Lemma 2. We therefore use this orientation for all the copies of $K_{2d'+1}$ we added, and orient all other edges incident on such cliques away from the clique. This does not make the in-degree of any vertex of V higher than d', since such vertices have at most $\mathbf{c}(u)$ incoming edges from E (by assumption), and $d' - \mathbf{c}(u)$ edges coming from the clique.

For the converse direction, consider a solution of the new instance. We first observe that without loss of generality we may assume that the solution does not delete any of the vertices of the cliques we added to the graph. This is because, if the solution deletes a vertex of a $K_{2d'+1}$ attached to $u \in V$, we can instead delete u and use the orientation described above for the vertices of the clique (which is now disconnected from the graph). If the solution does not delete any vertex from a $K_{2d'+1}$, by Lemma 2, all vertices of the clique have in-degree exactly d' in the clique. This implies that all edges with one endpoint in the clique must be oriented away from the clique, hence any vertex $u \in V$ may have in-degree at most $\mathbf{c}(u)$ from edges of E.

For the width bounds, first consider any path decomposition of G. We can construct a path decomposition of G' as follows: for each $u \in V$ to which we attached a $K_{2d'+1}$ we find a bag B of the original decomposition that contains u, and insert after B a copy of the same bag into which we add all the vertices of the clique attached to u. Consider now a clique-width expression for G. We can use it to construct a clique-width expression for G' using four new labels by introducing the clique attached to $u \in V$ immediately after the introduction of u, connecting the desired number of its vertices to u, and then renaming all of its vertices to a label that is not to be connected with anything else. Finally, to see that
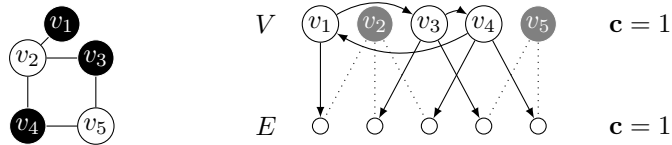
Figure 7 A sample graph with its independent set of size 3 in black on the left. The corresponding constructed instance with its 3-orientation on the right. The $n - k = 2$ deleted vertices are drawn in gray. Original edges with a deleted vertex as an endpoint are drawn dotted. For clarity, edges of the clique in V for deleted vertices are not drawn.

G' is chordal if G is chordal observe that no induced cycle of length 4 or more can contain any of the new vertices since they induce a union of cliques and are connected to the rest of the graph through single cut-vertices. ◀

Proof of Corollary 8

Proof. We observe that the reduction from Dominating Set given in Theorem 7 exactly preserves the optimal value. We can therefore invoke known hardness of approximation results on Dominating Set (see e.g. [18]). ◀

B   Omitted Material from Section 4

Proof of Theorem 9

Proof. We give a reduction from INDEPENDENT SET. Given a graph $G = (V, E)$ we are asked if it contains an independent set of size k. We assume without loss of generality that k is odd. We set $d = k$ and we will construct a chordal instance of CAPACITATED-d-ORIENTABLE DELETION, on which we will invoke Lemma 6 to obtain the claimed result, together with the standard fact that INDEPENDENT SET is W[1]-hard and not solvable in $n^{o(k)}$ under the ETH.

We construct a graph G' from G as follows. First, we subdivide all edges of E, and then we connect all (original) vertices of V into a clique. Let us now define the capacities: for any vertex introduced during a subdivision of an edge $e \in E$ we set its capacity to 1. Furthermore, for each $u \in V$ we set its capacity to $\frac{d-1}{2}$. This completes the construction (see also Figure 7). Observe that we have constructed a split graph, therefore G' is chordal.

Suppose that there is an independent set of size k in G, hence there is a vertex cover of size $n - k$. We delete the same vertices of G' and claim that the remaining graph is orientable in a way that respects the capacities. For every edge $e \in E$, its corresponding vertex has a deleted neighbor (since we started with a vertex cover). Hence, it now has degree at most 1, which is at most equal to its capacity, so we orient its possible remaining incident edge towards it. Finally, for the $k = d$ undeleted vertices of V, we use Lemma 2 to obtain an orientation of the K_d they induce where all vertices have in-degree $\frac{d-1}{2}$, which is equal to their capacities.

For the converse direction, suppose that it is possible to orient the graph respecting the capacities by deleting at most $n - k$ vertices. To simplify things we assume we have a solution that deletes exactly $n - k$ vertices, which can be achieved by adding arbitrary vertices to a smaller solution. If a solution deletes a vertex corresponding to $e \in E$, we can place that vertex back, orient all edges towards it, and if it now has in-degree 2 arbitrarily delete one of its neighbors in V. We therefore suppose that the solution deletes $n - k$ vertices of V and we must show that these vertices are a vertex cover of G. Suppose for contradiction that

there is an edge $e \in E$ such that neither of its endpoints in V was deleted. We now observe that in any orientation one edge connecting the vertex produced in the subdivision of e to V must be oriented towards V, because the vertex corresponding to e has capacity 1. However, the d undeleted vertices of V form a clique, and by Lemma 2, any orientation of the edges of this clique that gives all vertices in-degree at most $\frac{d-1}{2}$ (their capacities), gives all vertices exactly this in-degree. Hence, the additional edge from e will force one non-deleted vertex to violate its capacity. ◀

Proof of Theorem 11

Proof. This proof relies on standard techniques (dynamic programming on tree decompositions), so we will sketch some of the details. Recall that given a chordal graph G, it is known that we can obtain in polynomial time an optimal tree decomposition of G of width $\omega(G)$, where $\omega(G)$ is the maximum clique size of G. We now observe that, because of Lemma 2, we can assume that $\omega(G) < 2d + k + 2$, because if the graph contains a clique on $2d + k + 2$ vertices, even after deleting k vertices we will not be able to produce a d-orientation and we can immediately reject. We therefore have $\text{tw}(G) \leqslant 2d + k + 1$.

Our algorithm now performs standard dynamic programming on the given tree decomposition, similarly to the algorithm of [8]: we maintain a table in each bag which, for each vertex of the bag states either that the vertex has been deleted, or its in-degree in the orientation of the current partial solution. Since the in-degree is a number in $\{0, \ldots, d\}$, each vertex has $d + 2$ possible states. This makes the total size of the DP table at most $(d + 2)^{\text{tw}} \leqslant (d + 2)^{2d+k+1}$. It is now not hard to see that such a table can be updated in time polynomial in its size, giving us a solution at the root of the tree decomposition. ◀

C Omitted Material from Section 5

▶ **Lemma 16.** *Let $G = (V, E), \mathbf{c} : V \to \{0, \ldots, d\}$ be an instance of* CAPACITATED-d-ORIENTABLE DELETION*, and u, v be two vertices connected via an OR gadget. Then, any optimal solution must delete at least one of u, v.*

Proof. Suppose there exists an optimal solution that does not delete u or v. Because there are $2d + 2$ internal vertices in the gadget, and at most $2d$ of the edges of the gadget can be oriented towards u or v, there are at least two internal vertices of the gadget that are deleted by the solution. We place these vertices back in the graph and delete u in their place. We now have a strictly smaller solution, for which we can obtain a valid orientation, since the re-introduced vertices have capacity 1 and degree 1. This contradicts the optimality of the initial solution. ◀

▶ **Lemma 17.** *Let $G = (V, E), \mathbf{c} : V \to \{0, \ldots, d\}$ be an instance of* CAPACITATED-d-ORIENTABLE DELETION*, and G' be the graph obtained from G if, for each pair of vertices $u, v \in V$ which are connected through an OR gadget, we delete the internal vertices of the gadget and add the edge (u, v) if it does not exist. Then $\text{pw}(G) \leqslant \text{pw}(G') + 1$.*

Proof. Consider a path decomposition of G' and an OR gadget of G between vertices u, v. Since G' contains the edge (u, v), there is a bag containing both u and v. We insert immediately after this bag $2d + 2$ copies of it, and insert into each copy one of the internal vertices of the OR gadget. Repeating this for all OR gadgets gives a path decomposition of G of width 1 more than the original decomposition. ◀

Clause gadget $\hat{C}(N)$: This gadget is responsible for determining clause satisfaction, based on the choices made in the rest of the graph. It is identical to the one used for INDEPENDENT SET in [16], as finding a maximum independent set can be seen as equivalent to finding a minimum-sized deletion set for 0-orientability. The gadget consists of a sequence of N triangles, where every pair of consecutive triangles is connected by two edges, along with two pendant vertices attached to the first and last triangle. We set the capacities of all vertices to 0 and refer to the N degree-2 vertices within the triangles as the *inputs*. Figure 8 provides an illustration. For a clause C_μ with q_μ literals, the gadget's purpose is to offer an 1-in-q_μ choice, while its pathwidth remains constant: in any valid solution, there must be at least two vertices deleted from every triangle, but not all of the inputs, as this would leave a non-orientable edge between two capacity-0 vertices. In our final construction, the non-deleted leaf will correspond to some true literal within the clause.
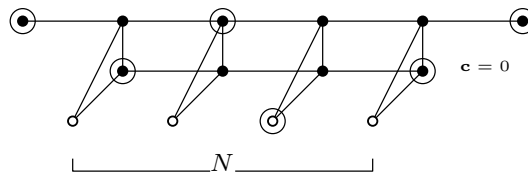
Figure 8 An example $\hat{C}(N)$. Note circled vertices forming an independent set, the rest being a minimum deletion set for 0-orientability.

▶ **Lemma 18.** *The minimum size of a deletion set K in $\hat{C}(N)$ is equal to $2N$, while this cannot include all input vertices and $pw(\hat{C}(N)) \leqslant 3$.*

Proof. We observe that, due to the capacities of all vertices being set to 0, there can be no edge left in the graph after deletion of K, meaning $\hat{C}(N) \setminus K$ must be an independent set. The lemma then follows from lemmas 2 and 3 of [16] (see also Claim 14.39 in [13]). ◀

▶ **Lemma 19.** *If ϕ has a satisfying assignment, then there exists $K \subseteq V(G)$ such that $G \setminus K$ is d-orientable, with $k = |K| = (tpd + 2) \sum_{\mu=1}^{m} (2q_\mu + t(3p(d+1) + (d+2)^p - 1))$.*

Proof. Given a satisfying assignment for ϕ we will show the existence of a deletion set K in G of size k, whose removal from G would leave the graph d-orientable. We first identify, for each $\tau \in [1, t]$, the vertex in U_τ^l (being the "same" for all $l \in [1, m(tpd + 2)]$) associated with the partial assignment to the group F_τ of variables and include in K all other vertices of U_τ^l, for each τ, l. Further, in each block gadget $\hat{B}$, we identify the corresponding option for deletion and include in K all such vertices: vertex a or b and d vertices in total from $X \cup Y$ (that correspond to the vertex deleted from U, for all p block gadgets this U is connected to), as well as the $2(d+1)$ vertices from sets W, Z that do not match this option. Finally, we include in K two vertices per triangle from each clause gadget $\hat{C}$ that do not match the above selections from the sets U: as this is a satisfying assignment, there will be at least one literal that is satisfied in each clause and the set U of the group F_τ that the literal's variable appears in will contain only one vertex that is not already included in K, while the clause gadget will have one input vertex that is connected to the *other* vertices of this U (via OR gadgets), i.e. the vertices whose associated assignments do not satisfy the literal. We thus include all other inputs in K, along with the other vertex per triangle whose removal would leave an independent set within $\hat{C}$ (by Claim 14.39 in [13], we can always find such an independent set).

Concerning the size of K, we have $3(d+1)$ vertices from each block gadget $\hat{B}$, $t(3p(d+1) + (d+2)^p - 1)$ vertices from all t p-groups of block gadgets, $2q_\mu + t(3p(d+1) + (d+2)^p - 1)$

for each column, giving for each m-section of columns $\sum_{\mu=1}^{m}(2q_\mu+t(3p(d+1)+(d+2)^p-1))$ (for one cycle through all clauses), that finally gives for all $tpd+2$ m-sections a total of $k=(tpd+2)\sum_{\mu=1}^{m}(2q_\mu+t(3p(d+1)+(d+2)^p-1))$. What remains is to show that $G\setminus K$ is indeed d-orientable.

In each block gadget $\hat{B}$, if $b\in K$, we orient all remaining edges from X to a and from Y to a' (the following gadget's a). As the capacity of a vertices is d and there are i vertices deleted from each X and $d-i$ from each Y (on the same line), this orientation does not exceed the limit of d for any vertex a. Next, we orient all remaining edges from X and Y towards the vertices of Q, whose capacities of d are again sufficient. Since the vertices remaining in W,Z (not w_{d+1},z_{d+1}, as $b\in K$) match the number of deletions within X,Y, we can safely orient all edges from X,Y towards the two remaining vertices in W,Z, whose capacities will be exactly sufficient (observe that $b\in K$ implies all edges from b to W,Z are also deleted). Further, all OR gadgets are satisfied, as at least one of their endpoints are in K: we only retain one vertex from each W and Z (that are not connected via an OR gadget as they match the *right* deletions from X,Y), while vertices $b,w_{d+1},z_{d+1}\in K$. On the other hand, if $a\in K$, then $a'\in K$ as well and the vertices remaining in W,Z are w_{d+1},z_{d+1}, whose OR gadgets to a,a' are satisfied, while there is no edge between them and b. The division of deletions within X,Y is now irrelevant but their number must still add up to d, thus we can again orient all edges from X,Y towards Q, while there are no remaining edges between X,Y and W,Z.

Concerning the connections between block and clause gadgets, the OR gadgets are again satisfied as K includes at least one of their endpoints and we can safely orient all edges from the other endpoint towards the gadget's inner vertices: there is only one vertex remaining from each W,Z and these complement the only vertex remaining from their connected set U, that in turn complements the remaining input from the column's clause gadget $\hat{C}$. Within each $\hat{C}$, the remaining 0-capacity vertices form an independent set. For the capacity cliques attached to each vertex, we orient all internal edges such that the vertices of the clique all have d incoming edges each and the edges towards the vertex the clique is attached to are oriented towards it, thus fixing its capacity, that is externally satisfied as described above (see also Lemma 6). ◀

▶ **Lemma 20.** *If there exists $K\subseteq V(G)$ such that $G\setminus K$ is d-orientable, with $k=|K|=(tpd+2)\sum_{\mu=1}^{m}(2q_\mu+t(3p(d+1)+(d+2)^p-1))$, then ϕ has a satisfying assignment.*

Proof. Given a deletion set K of size k such that $G\setminus K$ is d-orientable, we will show the existence of a satisfying assignment for ϕ. To identify the necessary structure of K, we first observe that the number of m-sections of columns (corresponding to a complete cycle over all clauses) being $tpd+2$, it suffices to show that for each m-section, any deletion set K will have to include $\sum_{\mu=1}^{m}(2q_\mu+t(3p(d+1)+(d+2)^p-1))$ vertices, meaning that in each column the number of vertices included in K must be $2q_\mu+t(3p(d+1)+(d+2)^p-1)$, by showing that $G\setminus K$ would not be d-orientable otherwise.

By Lemma 18, there will have to be at least $2q_\mu$ vertices included in K from every $\hat{C}$, while this cannot include all its inputs. This leaves $t(3p(d+1)+(d+2)^p-1)$ vertices for all t groups and we again aim to show that each group will require the deletion of at least $3p(d+1)+(d+2)^p-1$ vertices.

As each set U is a clique on $(d+2)^p$ vertices of capacity 1, at least all but one of them will have to be included in K, leaving $3p(d+1)$ vertices for the p block gadgets of each group, with $3(d+1)$ required from each $\hat{B}$: sets W,Z are OR-cliques, meaning that at least all but one vertex can be retained from each, thus accounting for the $2(d+1)$ vertices, while

each vertex in set Q is of capacity d with $2d$ edges from X, Y. As $|Q| = 2d + 1$, set K cannot include all of them and deleting fewer vertices from Q would not suffice to orient the remaining edges. Thus we know there must be at least d vertices from sets X, Y included in K, while for the final vertex from $\hat{B}$ we note that as a and b are connected via an OR gadget, at least one of them will also have to be in K. This gives the correct number of vertices included in K from each part of G.

If in each $\hat{C}$ there is one input that is not included in K, then all vertices in the sets U of this column that it is connected to (via OR gadgets) must be in K. This means the single vertices remaining in sets U in each column must match the inputs remaining in gadgets $\hat{C}$, as every input is connected to all vertices associated with some partial assignment that *does not* satisfy its corresponding literal. Also, each vertex from each set U is connected to all but one vertices of each of the p pairs of sets W, Z in the group of block gadgets $\hat{B}$ of this set U (again via OR gadgets). Thus the vertices remaining within each W, Z must match the vertex retained from their connected set U. Then in each $\hat{B}$, due to the OR gadgets between non-matching vertices of W, Z, we know the vertices retained from W and Z must agree, i.e. they must be of the same index i. For $i \neq d + 1$, this implies $b \in K$ (due to the edges from b to both w_{d+1}, z_{d+1}) and further, that the number of vertices in K from X is i, while the number of vertices in K from Y is $d - i$ (due to the capacities of w_{d+1}, z_{d+1}). For $i = d + 1$, we must have $a \in K$ and also $a' \in K$. Thus the number of deletions within sets X and Y of each block gadget $\hat{B}$, along with deletion of either a or b (i.e. the options within $\hat{B}$) must match the vertices remaining in sets W, Z, that in turn must match (complement) the vertex remaining in set U (over all p block gadgets of that group), that must finally also match (complement) the input remaining in that column's clause gadget $\hat{C}$.

Next, we require that there exists at least one $o \in [0, tpd + 1]$ for every $\tau \in [1, t]$ for which the selected options within all p block gadgets $\hat{B}_\tau^{mo+\mu,\pi}$ (and thus also the corresponding retained vertices from sets U) do not change for all $\mu \in [1, m]$, i.e. that there exists some m-section (m successive columns) for which the selected options remain unchanged (over all rows). Observe that, if i vertices are included in K from a set X and $d - i$ from Y in some gadget $\hat{B}_\tau^{l,\pi}$ and $a, a' \notin K$, then a' must have i incoming edges from Y, which implies it can have at most $d - i$ incoming edges from the set X of the following gadget $\hat{B}_\tau^{l+1,\pi}$ (inside which it is the vertex a). If, on the other hand, $b \notin K$ in some gadget $\hat{B}_\tau^{l,\pi}$, then both a, a' must be in K due to the OR gadgets between b and a, a' and the budget $(3(d + 1))$ for the following gadget $\hat{B}_\tau^{l+1,\pi}$ implies that $b \notin K$ there as well. This means the options for deletion over a row of gadgets (fixed $\tau \in [1, t], \pi \in [1, p]$) are *monotone*: if the selected option in some gadget $\hat{B}_\tau^{l,\pi}$ corresponds to deleting i vertices from X, then the selected option in the following gadget $\hat{B}_\tau^{l+1,\pi}$ must correspond to deleting i' vertices from X, with $i' \geqslant i$, while if vertex b is retained in some gadget (even if it was deleted in its predecessor) then it must also be retained in its follower. Thus this "shift" in options can happen at most $d + 1$ times for each $\pi \in [1, p]$, giving $p(d + 1)$ times for each $\tau \in [1, t]$, or $tp(d + 1)$ times over all τ. By the pigeonhole principle, there must thus exist an $o \in [0, tp(d + 1)]$ such that no shift happens among the gadgets $\hat{B}_\tau^{mo+\mu,\pi}, \forall \tau \in [1, t], \pi \in [1, p], \mu \in [1, m]$.

Our assignment for ϕ is then given by the selected options within each gadget $\hat{B}_\tau^{mo+1}$ for this o: for every group F_τ we consider the single remaining vertex in U_τ^{mo+1} that is associated with a partial assignment for the variables in F_τ. As there must be exactly one remaining vertex in each set U that matches both the selected options within all p gadgets of its group, as well as the remaining input within that column's clause gadget $\hat{C}$, these selected options being unchanged over the complete m-section, this assignment also satisfies each clause C_μ for all $\mu \in [1, m]$.   ◀

▶ **Lemma 21.** *Graph G has treewidth $tw(G) \leqslant tp + f(d, \epsilon)$, for $f(d, \epsilon) = O(d^p)$.*

Proof. We will show a pathwidth bound of $pw(G) \leqslant tp + O(d^p)$ by providing a mixed search strategy to clean G using at most this many searchers simultaneously. The treewidth bound then follows from Lemma 1 and the well known relationship between pathwidth and *mixed search number* ms (see [21]): *for any graph G, it is $pw(G) \leqslant ms(G) \leqslant pw(G) + 1$.*

We initially place one searcher on every vertex a of gadgets $\hat{B}_\tau^{1,\pi}$ for all $\tau \in [1, t], \pi \in [1, p]$. These account for the tp searchers that we will be moving to the vertices a of the following gadgets on each line after the inner parts of each column have been cleaned, while the remaining searchers will be (re)used to clean each part of each column.

In particular, we assume the input vertices of $\hat{C}_1^1$ are ordered in terms of the group F_τ of variables that their associated literals appear in and place a searcher on each vertex of the set U_τ^1 that the first input vertex of $\hat{C}_1^1$ is connected to, along with three searchers on the vertices of the first triangle. Using some of the remaining searchers we first clean all paths between this input and U_τ^1. Note that OR gadgets involve $2d + 3$ vertices, with only 3 searchers being sufficient (Lemma 17). Concerning the cliques attached to each vertex to set its capacity, see Lemma 6. If the next input is also associated with a variable in group F_τ, we slide the searchers on the next triangle. If the next input is associated with a variable appearing in some following group $F_{\tau'}$, we keep the searchers on the same triangle until this block gadget has been cleaned and we reach the group of variables (and block gadgets) $F_{\tau'}$.

To clean a block gadget $\hat{B}_\tau^{1,\pi}$, we place a searcher on each vertex of sets X, Y, W, Z, Q. These are $< 12d$ and we can completely clean the gadget (and all vertices between it and U_τ^1) using the remaining searchers. We then remove the searcher from vertex a and place it on a' (to clean the following gadget after the first column has been cleaned). Having thus cleaned $\hat{B}_\tau^{1,\pi}$ we remove all searchers from it and place them on the corresponding vertices of $\hat{B}_\tau^{1,\pi'}$ and repeat the process. After all p gadgets for τ have been cleaned in this way, the initial searchers having moved from vertices a to a', we can remove all searchers from U_τ^1 and place them on $U_{\tau'}^1$ and repeat the process for τ'. Performing the above for all $\tau \in [1, t]$ cleans the first column, along with the first clause gadget and we can then remove the three searchers from $\hat{C}_1^1$ and place them on the first triangle of $\hat{C}_2^1$ (see also Lemma 18).

Repeating the above process for all $m(tpd + 2)$ columns completely cleans the graph and the number of searchers we have used simultaneously is $\leqslant tp + O(d^p)$: our tp initial searchers that we move from vertices a to a', three searchers for each clause gadget, $(d + 2)^p$ searchers for each set U, at most $12d$ searchers for each block gadget $\hat{B}$ and at most $4d$ searchers for the smaller gadgets and remaining intermediate paths/edges. ◀

Proof of Theorem 12

Proof. Assuming the existence of some algorithm of running time $O^*((d + 2 - \epsilon)^{tw(G)}) = O^*((d + 2)^{\lambda tw(G)})$ for d-ORIENTABLE DELETION, where $\lambda = \log_{d+2}(d + 2 - \epsilon)$, and given a formula ϕ of SAT, we construct an instance of d-ORIENTABLE DELETION using the above construction and then solve the problem using the $O^*((d + 2 - \epsilon)^{tw(G)})$-time algorithm. Correctness is given by Lemma 19 and Lemma 20, while Lemma 21 gives the upper bound on the running time:

$$O^*((d+2)^{\lambda \text{tw}(G)}) \leqslant O^*\left((d+2)^{\lambda(tp+f(d,\epsilon))}\right) \tag{1}$$

$$\leqslant O^*\left((d+2)^{\lambda p \left\lceil \frac{n}{\lfloor \log_2(d+2)^p \rfloor} \right\rceil}\right) \tag{2}$$

$$\leqslant O^*\left((d+2)^{\lambda p \frac{n}{\lfloor \log_2(d+2)^p \rfloor} + \lambda p}\right) \tag{3}$$

$$\leqslant O^*\left((d+2)^{\lambda \frac{np}{\lfloor p \log_2(d+2) \rfloor}}\right) \tag{4}$$

$$\leqslant O^*\left((d+2)^{\delta' \frac{n}{\log_2(d+2)}}\right) \tag{5}$$

$$\leqslant O^*(2^{\delta'' n}) = O^*((2-\delta)^n) \tag{6}$$

for some $\delta, \delta', \delta'' < 1$. Observe that in line (2) the function $f(d, \epsilon)$ is considered constant, as is λp in line (4), while in line (5) we used the fact that there always exists a $\delta' < 1$ such that $\lambda \frac{p}{\lfloor p \log_2(d+2) \rfloor} = \frac{\delta'}{\log_2(d+2)}$, as we have:

$$p \log_2(d+2) - 1 < \lfloor p \log_2(d+2) \rfloor$$

$$\Leftrightarrow \frac{\lambda p \log_2(d+2)}{p \log_2(d+2) - 1} > \frac{\lambda p \log_2(d+2)}{\lfloor p \log_2(d+2) \rfloor},$$

from which, by substitution, we get $\frac{\lambda p \log_2(d+2)}{p \log_2(d+2) - 1} > \delta',$

now requiring $\frac{\lambda p \log_2(d+2)}{p \log_2(d+2) - 1} \leqslant 1,$

or $p \geqslant \frac{1}{(1-\lambda) \log_2(d+2)},$

that is precisely our definition of p. This concludes the proof. ◀

D Omitted Material from Section 6

We formally define the notion of a (partial) *valid solution*: for a node t of the clique-width expression T, deletion set K_t and orientation Λ_t, a (partial) *solution* $X_t := \{K_t \subseteq V(G_t), \Lambda_t\}$ is *valid*, if $|K_t| \leqslant k$ and $\delta_t^-(v) \leqslant d, \forall v \in V(G_t)$ under orientation Λ_t. For two nodes t_1, t_2 of some clique-width expression T where t_2 is a successor of t_1, we say that a solution $Y_{t_2} := \{K_{t_2}, \Lambda_{t_2}\}$ *extends* a solution $X_{t_1} := \{K_{t_1}, \Lambda_{t_1}\}$, if $K_{t_1} \subseteq K_{t_2}$ and Λ_{t_2} assigns the same directions as Λ_{t_1} to all edges in G_{t_1}. Further, we formally define a *large* in-degree-class: for node t and label $i \in [1, \text{cw}]$, an in-degree-class $j \in [0, d]$ is *large*, if its size is larger than d^4, or $A_t^{i,j} = |v \in V(G_t^i) : \delta_t^-(v) = j| \geqslant d^4$.

▶ **Lemma 22.** *For some node $t \in T$, label $i \in [1, cw]$ and large in-degree-class $j \in [0, d]$ of i, if there exists a valid partial solution $X_t = \{K_t, \Lambda_t\}$, that is extended to a valid global solution $X_r = \{K_r, \Lambda_r\}$ for the root r of T, where Λ_t assigns a direction to some edges (v, u)*

away from vertices $v \in V_t^i$ belonging to the large in-degree-class j, then there exists another valid partial solution $X_t' = \{K_t', \Lambda_t'\}$ that is extended to a valid global solution $X_r' = \{K_r', \Lambda_r'\}$, with $|K_r'| \leqslant |K_r|$, where Λ_t' assigns direction (u, v) to all edges of vertices v belonging to the large in-degree-class j.

Proof. Lemma 3 implies that no valid solution to d-ORIENTABLE DELETION can be obtained for a graph including some biclique larger than $K_{2d,2d}$ (on any side) and in particular, that we can assume that any Join node involves at most one label with a large in-degree-class, while the other cannot have $2d$ vertices in total (all other partial solutions can be discarded).

Consider a Join node t with $\eta_{p,q}$ for $p, q \in [1, \mathrm{cw}]$, where label p contains some large in-degree-class $j \in [0, d]$. As noted, we can assume $|V_t^q| < 2d$. Now, observe that the maximum number of newly created edges between class j and label q that can be oriented towards the vertices of q is $< 2d^2$, as each vertex of q can have at most d incoming edges for the partial solution to remain valid. This means $> d^4 - 2d^2$ vertices of j will have to be moved to another in-degree-class $j' > j$. The number of such shifts is bounded by the number d of in-degree-classes for a partial solution to remain valid. Thus there will always be $> d^4 - 2d^3$ vertices in the corresponding in-degree-class, which for $d \geqslant 2$ is $> d^3$.

Let $X_t = \{K_t, \Lambda_t\}$ be a valid partial solution that, during the computation of in-degree-signatures for some Join node t with $\eta_{p,q}$, selects an orientation Λ_t for the newly created edges that directs some edges (v, u) away from a vertex v of the large in-degree-class j of label p and towards a vertex u of label q. Also let $Y_r = \{K_r, \Lambda_r\}$ be a valid global solution that extends X_t. Now consider global solution $Y_r' = \{K_r', \Lambda_r'\}$ that differs from Y_r by simply selecting an orientation Λ_r' that differs from Λ_r only in the direction of edges (v, u): if Y_r is valid, it is $|K_r| \leqslant k$ and Λ_r assigns no more than d incoming edges to any vertex. As noted above, there must exist at least one vertex $w \in V_t^p$ belonging to large in-degree-class j, whose edges are all oriented towards it in Λ_r and these must be $\leqslant d$. Since w and vertices v belong to the same label p, any Join operations following node t applied to w are also applied to all v and thus, the orientation Λ_t' that directs (u, v) towards v can also be extended to an orientation Λ_r' for global solution Y_r' that does not require any more deleted vertices than Y_r (keeping $|K_r'| \leqslant |K_r|$) and remains valid as $\delta_r^-(w) = \delta_r^-(v) \leqslant d$ for all such $v \in j$. ◄

Proof of Theorem 14

Proof. As explained above, our algorithm is a modification of the DP given in [9], the necessary modifications being the following. First, due to our focusing on the decision version of d-ORIENTABLE DELETION, we consider that including a vertex in the deletion set and appropriately decreasing the given budget k is decided upon while computing the table of its Leaf node. Next, if for some $i \in [1, \mathrm{cw}], j \in [0, d]$ we have $A_t^{i,j} \geqslant d^4$, we replace the actual numerical value with a distinct marker and refer to this in-degree-class as large.

We compute the tables of Union and Relabel nodes in the same way, noting that any in-degree-class that is combined through such a node's application with a large in-degree-class will have to become large in turn. Further, observe that if both labels involved in a Join operation contain at least one large in-degree-class then the implied partial solution cannot be valid, due to Lemma 3. This also implies that for a valid partial solution, if one label contains at least one large in-degree-class, then the size of the other label must be $< 2d$.

Then, by Lemma 22, to update the tables of a Join node t with $\eta_{p,q}$ for $p, q \in [1, \mathrm{cw}]$, where label p contains some large in-degree-class $j \in [0, d]$, we assume that all newly created edges attached to the vertices of j are oriented towards them in any orientation Λ_t for this node, thus being able to correctly update the sizes of all in-degree-classes and both

labels p, q: as all edges will be oriented towards the vertices of j, then all these vertices will have an increase of in-degree equal to the size of q, meaning that the in-degree-class $j + |V_t^q|$ of p will now have to become large, while all other computations of orientations and in-degree-signatures remain unchanged. As we now require $(d^4)^{d+2}$ states for each label (instead of n^{d+2}), our running time is $O^*(d^{O(d \cdot \mathrm{cw})})$.   ◀

E   Omitted Material from Section 7

▶ **Lemma 23.** *If G has a k-multicolored independent set, then G' has a deletion set K', such that $G' \setminus K'$ is d-orientable, with $|K'| = kn + 3|E|$.*

Proof. Let $K \subseteq V$ be a k-multicolored independent set in G and $v_i^{l_i}$ denote the vertex selected from each V_i, or $K := \{v_1^{l_1}, \ldots, v_i^{l_i}, \ldots, v_k^{l_k}\}$. Our deletion set K' will include l_i vertices from each A_i and $n - l_i$ vertices from each B_i (thus n vertices in total will be deleted from each pair of A_i, B_i). Further, for each edge $e = (v_i^l, v_j^o) \in E$, our deletion set K' will also include 3 out of the 4 vertices $a_l^e, b_l^e, a_o^e, b_o^e$ (it could not include less than 3, due to the OR gadgets anywhere between these four). We identify the vertices to include in K' as follows: since K is a k-multicolored independent set, we know there is no edge $e = (v_i^{l_i}, v_j^{l_j}) \in E$ and thus, at least one endpoint of the edge in question is different than the actual selection within V_i, i.e. either $l \neq l_i$, or $o \neq l_j$, or both (also for the other symmetrical cases). Without loss of generality we assume the selection within V_i differs from the v_i^l endpoint of e, or $l \neq l_i$. Thus, K' includes l_i vertices from A_i and $n - l_i$ vertices from B_i. If $l > l_i$, then $l - 1 \geqslant l_i$ and the capacity of vertex b_l^e will be sufficient for orientation of all (remaining) edges from B_i towards it. We thus include in K' all other three vertices a_l^e, a_o^e, b_o^e. On the other hand, if $l < l_i$, then $n - l - 1 \geqslant n - l_i$ and now the capacity of vertex a_l^e will be sufficient for orientation of all (remaining) edges from A_i towards it. We thus include in K' all other three vertices b_l^e, a_o^e, b_o^e.

In this way we have completed the deletion set K' with $kn + 3|E|$ vertices and what remains is to show that $G' \setminus K'$ is d-orientable. Having deleted n vertices from each pair A_i, B_i, we can orient all edges from the remaining n vertices in each A_i, B_i towards each guard vertex in W_i, whose capacities are equal to n. Having also retained the correct vertex from each quadruple corresponding to some edge e in G, we can safely orient all edges from either A_i or B_i (depending on whether the vertex that is not included in K' is an a_l^e or b_l^e) to the remaining vertex, whose capacity will be sufficient, as explained above. Finally, the edges remaining within the OR gadgets are oriented towards the vertices of the gadget, whose capacity of 1 is sufficient for the single remaining edge, as the other endpoint of the gadget is included in K'.   ◀

▶ **Lemma 24.** *If G' has a deletion set K', such that $G' \setminus K'$ is d-orientable, with $|K'| = kn + 3|E|$, then G has a k-multicolored independent set.*

Proof. First, observe that due to the guard vertices W_i being of capacity n and connected to $2n$ vertices each (whose capacities are equal to 0), at least n vertices must be deleted from each pair of A_i, B_i (included in K'). Deleting any guard vertex does not decrease the number of edges that must be oriented towards the remaining vertices of W_i and the size of K' does not allow for deletion of all guard vertices. Next, observe that due to the OR gadgets connecting all four vertices $a_l^e, b_l^e, a_o^e, b_o^e$ corresponding to some edge $e = (v_i^l, v_j^o) \in E$, at least three of them must be included in K' for $G' \setminus K'$ to be d-orientable. Since $|K'| = kn + 3|E|$, then exactly n vertices are deleted from each pair of A_i, B_i for each $i \in [1, k]$ and exactly 3 vertices are deleted from each quadruple of vertices corresponding to some edge of G.

Let $l_i \in [0, n]$ be the number of vertices deleted from each A_i (meaning $n - l_i$ are deleted from B_i). We let our set K include the l_i-th vertex from each V_i, i.e. $K :=$ $\{v_1^{l_1}, \ldots, v_i^{l_i}, \ldots, v_k^{l_k}\}$ and claim K is a k-multicolored independent set in G. Suppose that this is not the case, meaning there exists some edge $e = (v_i^{l_i}, v_j^{l_j}) \in E$. Consider the quadruple of vertices $a_{l_i}^e, b_{l_i}^e, a_{l_j}^e, b_{l_j}^e$ corresponding to edge e: as there are l_i vertices deleted from A_i, there are $n - l_i$ (remaining) edges between A_i and $a_{l_i}^e$. As the capacities of all vertices in A_i are 0, all these edges must be oriented towards $a_{l_i}^e$, whose capacity is $n - l_i - 1 < n - l_i$. Further, there are $n - l_i$ vertices deleted from B_i and thus l_i edges need to be oriented towards $b_{l_i}^e$, whose capacity is $l_i - 1 < l_i$. This means both $a_{l_i}^e$ and $b_{l_i}^e$ must be included in K' for $G' \setminus K'$ to be d-orientable. The same holds for $a_{l_j}^e, b_{l_j}^e$, however, whose capacities will also be 1 less than the required number of edges to be oriented towards them. This implies that if the vertices from each V_i corresponding to the numbers of deletions within each pair of A_i, B_i are not independent, then either set K' is of size larger than $kn + 3|E|$, or that $G' \setminus K'$ is not d-orientable, which is a contradiction. ◄

▶ **Lemma 25.** *The clique-width of G' is $cw(G') \leqslant 2k + O(1)$.*

Proof. We will describe the sequence of operations to construct graph G' using at most $2k + O(1)$ labels, these being two labels for each $i \in [1, k]$ and a small number of "work" labels, along with a "junk" label to which we relabel the completed parts of the graph in order to reuse their work labels. We first introduce all vertices of sets A_i and B_i using one label per set and then all vertices of W_i using one of the work labels. We then join W_i to both A_i, B_i and relabel W_i to the junk label, thus being able to reuse its former work label for introduction of the following set W_{i+1}.

Following this procedure for all $i \in [1, k]$, we turn to introduction of the quadruples of vertices for each edge $e = (v_i^l, v_j^o) \in E$. We introduce each of the 4 vertices $a_l^e, b_l^e, a_o^e, b_o^e$ using one label per vertex and join them to the appropriate sets A_i, B_i, A_j, B_j. We then introduce the $2d + 2$ vertices of the OR gadget connecting a_l^e, b_l^e using some work label and join them to both a_l^e, b_l^e. Relabelling the vertices of the OR gadget to the junk label allows us to reuse their previous work label for the remaining OR gadgets connecting the quadruple. Having thus completed the construction for edge e, we can relabel all vertices to the junk label and reuse their former work labels for the quadruple of the remaining edges.

Concerning the cliques attached to each vertex to set their capacity, observe that we can introduce the vertex in question using some work label first, then introduce the vertices of the clique that should be adjacent to it using another work label, join them and then continue with the construction of the complete subgraph that forms the gadget using the same work label (see also Lemma 6). Having thus constructed the clique for this vertex, we can relabel it to its "proper" label as described above in order to continue with our construction and also relabel all vertices of the clique to the junk label, thus being able to reuse the same work labels for the remaining cliques. ◄

Proof of Theorem 15

Proof. Given some instance of k-Multicolored Independent Set, we construct an instance of d-Orientable Deletion with $d = n$, as described above. Lemmas 23, 24 show correctness of the reduction, while Lemma 25 gives the bound on the clique-width of the constructed graph. For the running time bound we recall that an algorithm for k-Multicolored Independent Set running in time $n^{o(k)}$ would contradict the ETH. ◄