Parameterized Orientable Deletion

Tesshu Hanaka
Department of Information and System Engineering, Chuo University, Tokyo, Japan
hanaka.91t@g.chuo-u.ac.jp

Ioannis Katsikarelis
Université Paris-Dauphine, PSL Research University, CNRS, UMR 7243,
LAMSADE, 75016 Paris, France
ioannis.katsikarelis@dauphine.fr

Michael Lampis
Université Paris-Dauphine, PSL Research University, CNRS, UMR 7243,
LAMSADE, 75016 Paris, France
michail.lampis@dauphine.fr

Yota Otachi
Kumamoto University, Kumamoto, 860-8555, Japan
otachi@cs.kumamoto-u.ac.jp

Florian Sikora
Université Paris-Dauphine, PSL Research University, CNRS, UMR 7243,
LAMSADE, 75016 Paris, France
florian.sikora@dauphine.fr

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.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms, Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Graph orientations, FPT algorithms, Treewidth, SETH

Digital Object Identifier 10.4230/LIPIcs.SWAT.2018.24

Funding This work was financially supported by the "PHC Sakura" program (project GRAPA, number: 38593YJ), implemented by the French Ministry of Foreign Affairs, the French Ministry of Higher Education and Research and the Japan Society for Promotion of Science.

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, 20]). 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 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 [19] 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, 18].

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 omitted.

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]. We also use a corollary (a slightly weaker statement) of the Strong Exponential Time Hypothesis (SETH), stating 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 $tw(G) \leqslant pw(G)$ and $cw(G) \leqslant pw(G) + 2$.*

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: $pw(G') \leqslant pw(G) + 2d' + 1$, $cw(G') \leqslant cw(G) + 4$, and if G is chordal then G' is chordal.*

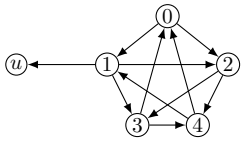

Figure 1 A 2-orientation of a clique K_5. Observe that any edge connecting a vertex u to the clique must be oriented towards u (setting its *capacity*) to maintain a 2-orientation.

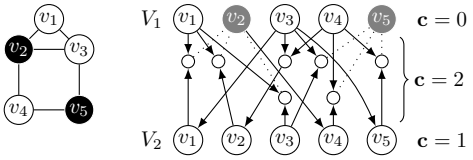

Figure 2 Left: A graph with its dominating set in black. **Right:** The corresponding instance and 2-orientation, where deleted vertices are in gray. Original edges with a deleted vertex as an endpoint are dotted.

Proof (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. ◀

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 $f(k) \cdot n^{o(k)}$, 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 $f(k) \cdot n^{o(k)}$ 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 H 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 H 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 H 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 H 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 vertex 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 $v \in N_G[u]$, i.e. all vertices v that are neighbors of u in G. 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 $\frac{n-k-1}{2}$.

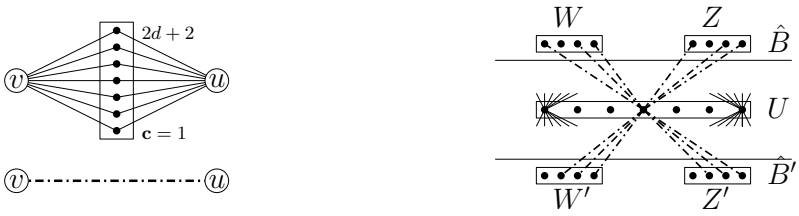

Figure 3 Left: An example OR gadget. In the following, OR gadgets are shown as dotted edges. **Right:** Example connections between a set U and the p sets W, Z in the gadgets $\hat{B}$ of its group.

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 endpoints 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.

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. 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 $X \coloneqq \{x_1, \ldots, x_d\}$, $Y \coloneqq \{y_1, \ldots, y_d\}$, $Q \coloneqq \{q_1, \ldots, q_{2d+1}\}$.
3. Make two sets $W \coloneqq \{w_0, \ldots, w_{d+1}\}$ and $Z \coloneqq \{z_0, \ldots, z_{d+1}\}$.
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).

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

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 $(d+2-\epsilon)^{\mathrm{tw}}$ 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

[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.

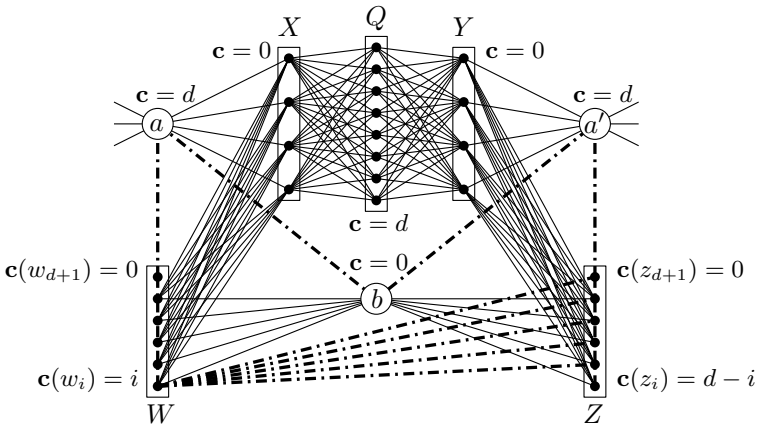

Figure 4 Our block gadget $\hat{B}$. Capacities are shown next to vertices/sets, the OR-connections within W, Z are shown as paths, while the OR-connections between W, Z are only shown for w_0.

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. 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.*

[2] We assume that q_μ is always even, by duplicating some literals if necessary.

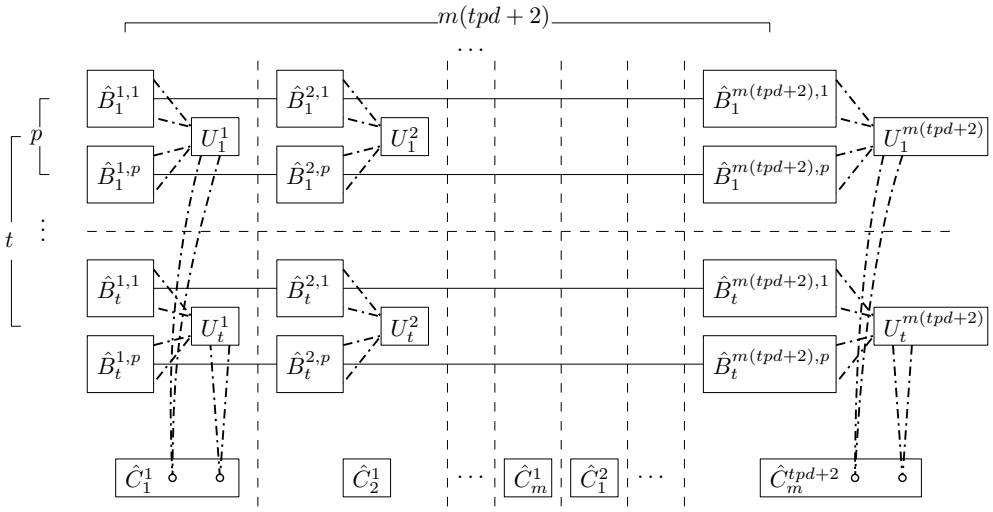

Figure 5 A simplified picture of the complete construction.

▶ **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.*

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 $d^{O(d \cdot \mathrm{cw})}$. 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 $n^{O(d \cdot \mathrm{cw})}$, 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 $A_t = (A_t^{i,j}), \forall i \in [1, \mathrm{cw}], j \in [0, d]$, if there is an orientation Λ_t (of every edge of G_t) such that for each label $i \in [1, \mathrm{cw}]$ 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 $K_t := \bigcup_{i \in [1,\mathrm{cw}]} K_t^i$ for each $i \in [1, \mathrm{cw}]$, 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.

[3] Slightly paraphrased here for d-Orientable Deletion, keeping the same notation.

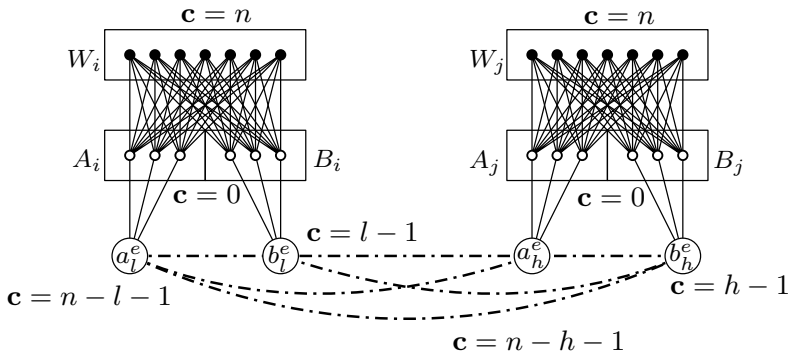

Figure 6 A partial view of the construction, depicting the gadgets encoding the selection for V_i, V_j, as well the representation of an edge $e = (v_i^l, v_j^h)$. Note dotted edges signifying OR gadgets.

The aim of this section is to improve the running time of the above algorithm to $d^{O(d \cdot \mathrm{cw})}$, 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 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 $n^{o(\mathrm{cw})}$. As a result, the parameter dependence of $d^{O(\mathrm{cw})}$ 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. As before, we employ capacities and implicitly utilize Capacitated-d-Orientable Deletion.

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 $V_i \coloneqq \{v_i^1, \ldots, v_i^n\}, \forall i \in [1, k]$. 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 $e = (v_i^l, v_j^h) \in E$ with endpoints $v_i^l \in V_i, v_j^h \in V_j$ (i.e. the l-th vertex of V_i and the h-th vertex of V_j), make four new vertices $a_l^e, b_l^e, a_h^e, b_h^e$.
5. Connect $a_l^e, b_l^e, a_h^e, b_h^e$ 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_h^e is connected to all vertices of A_j and b_h^e to all vertices of B_j.
7. Set the capacities $\mathbf{c}(a_l^e) = n - l - 1$, $\mathbf{c}(b_l^e) = l - 1$, $\mathbf{c}(a_h^e) = n - h - 1$ and $\mathbf{c}(b_h^e) = h - 1$.

▶ **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 Yuichi Asahiro, Jesper Jansson, Eiji Miyano, and Hirotaka Ono. Upper and lower degree bounded graph orientation with minimum penalty. In *CATS'12*, volume 128, pages 139–145, 2012.

2 Yuichi Asahiro, Jesper Jansson, Eiji Miyano, and Hirotaka Ono. Degree-constrained graph orientation: Maximum satisfaction and minimum violation. *Theory Comput. Syst.*, 58(1):60–93, 2016. doi:10.1007/s00224-014-9565-5.

3 Yuichi Asahiro, Jesper Jansson, Eiji Miyano, Hirotaka Ono, and Kouhei Zenmyo. Approximation algorithms for the graph orientation minimizing the maximum weighted outdegree. *J. Comb. Optim.*, 22(1):78–96, 2011.

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

5 Yuichi Asahiro, Eiji Miyano, Hirotaka Ono, and Kouhei Zenmyo. Graph orientation algorithms to minimize the maximum outdegree. *Int. J. Found. Comput. Sci.*, 18(2):197–215, 2007. doi:10.1142/S0129054107004644.

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

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

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

9 Hans L. Bodlaender, Hirotaka Ono, and Yota Otachi. Degree-constrained orientation of maximum satisfaction: Graph classes and parameterized complexity. *Algorithmica*, 80(7):2160–2180, 2018. doi:10.1007/s00453-017-0399-9.

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

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

12 Bruno Courcelle and Joost 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. URL: http://www.cambridge.org/fr/knowledge/isbn/item5758776/?site_locale=fr_FR.

13 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.

14 Tomás Ebenlendr, Marek Krcál, and Jirí Sgall. Graph balancing: A special case of scheduling unrelated parallel machines. *Algorithmica*, 68(1):62–80, 2014. doi:10.1007/s00453-012-9668-9.

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

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

18 Geevarghese Philip, Ashutosh Rai, and Saket Saurabh. Generalized pseudoforest deletion: Algorithms and uniform kernel. In *MFCS (2)*, volume 9235 of *LNCS*, pages 517–528, 2015.

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

20 José Verschae and Andreas Wiese. On the configuration-lp for scheduling on unrelated machines. *J. Scheduling*, 17(4):371–383, 2014. doi:10.1007/s10951-013-0359-4.