On Structural Parameterizations of the Bounded-Degree Vertex Deletion Problem

Robert Ganian[1]
Algorithms and Complexity Group, TU Wien, Vienna, Austria

Fabian Klute
Algorithms and Complexity Group, TU Wien, Vienna, Austria

Sebastian Ordyniak
Algorithms and Complexity Group, TU Wien, Vienna, Austria

Abstract

We study the parameterized complexity of the Bounded-Degree Vertex Deletion problem (BDD), where the aim is to find a maximum induced subgraph whose maximum degree is below a given degree bound. Our focus lies on parameters that measure the structural properties of the input instance. We first show that the problem is W[1]-hard parameterized by a wide range of fairly restrictive structural parameters such as the feedback vertex set number, pathwidth, treedepth, and even the size of a minimum vertex deletion set into graphs of pathwidth and treedepth at most three. We thereby resolve the main open question stated in Betzler, Bredereck, Niedermeier and Uhlmann (2012) concerning the complexity of BDD parameterized by the feedback vertex set number. On the positive side, we obtain fixed-parameter algorithms for the problem with respect to the decompositional parameter treecut width and a novel problem-specific parameter called the core fracture number.

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

Keywords and phrases Bounded-degree Vertex Deletion, Feedback Vertex Set, Parameterized Algorithms, Treecut Width

Digital Object Identifier 10.4230/LIPIcs.STACS.2018.33

Funding This work was supported by the Austrian Science Fund (FWF), project P26696.

1 Introduction

This paper studies the Bounded-Degree Vertex Deletion problem (BDD): given an undirected graph G, a degree bound d, and a limit ℓ, determine whether it is possible to delete at most ℓ vertices from G in order to obtain a graph of maximum degree at most d. Aside from being a natural generalization of the classical Vertex Cover problem, BDD has found applications in areas such as computational biology [17] and is the dual problem of the so-called *s-Plex Detection* problem in social network analysis [30, 3, 31, 35].

It is not surprising that the complexity of BDD and several of its variants has been studied extensively by the theory community in the past years [5, 4, 7, 6, 11, 26, 34, 35]. Since the problem is NP-complete in general, it is natural to ask under which conditions does the problem become tractable. In this direction, the *parameterized complexity* paradigm [13, 33, 9] allows a more refined analysis of the problem's complexity than classical complexity. In the

[1] Robert Ganian is also affiliated with FI MU, Brno, Czech Republic.

35th Symposium on Theoretical Aspects of Computer Science (STACS 2018).
Editors: Rolf Niedermeier and Brigitte Vallée; Article No. 33; pp. 33:1–33:14

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

parameterized setting, we associate each instance with a numerical parameter k and are most often interested in the existence of a *fixed-parameter algorithm*, i.e., an algorithm solving the problem in time $f(k) \cdot |V(G)|^{\mathcal{O}(1)}$ for some computable function f. Parameterized problems admitting such an algorithm belong to the class FPT; on the other hand, parameterized problems that are hard for the complexity class W[1] or W[2] do not admit fixed-parameter algorithms (under standard complexity assumptions).

In general, there exist two notable approaches for selecting parameters: a parameter may either originate from the formulation of the problem itself (often called *natural parameters*), or rather from the structure of the input graph (so-called *structural parameters*, most prominently represented by the decomposition-based parameter *treewidth* **tw**). The parameterized complexity of BDD has already been studied extensively through the lens of natural parameters (especially d and ℓ). In particular, BDD is known to be FPT when parameterized by $d + \ell$ [34, 17, 31], W[2]-hard when parameterized only by ℓ [17], and NP-complete when parameterized only by d (as witnessed by the case of $d = 0$, i.e., Vertex Cover). The complexity of BDD is also fairly well understood when considering combinations of natural and structural parameters: it is FPT when parameterized by $\mathbf{tw} + d$ due to Courcelle's Theorem [8] and has been shown to be FPT when parameterized by $\mathbf{tw} + \ell$ [5].

Given the above, it is fairly surprising that the problem has remained fairly unexplored when viewed through the lens of structural parameters only, i.e., in the case where we impose no restrictions on the problem formulation itself but only on the structure of the graph. BDD was shown not to be FPT when parameterized by treewidth [5], complementing the previous $\mathcal{O}(n^{\mathbf{tw}+1})$ algorithm of Dessmark et al. [11]. The only structural parameter which is known to make the problem fixed-parameter tractable is the *feedback edge set number*, i.e., the minimum number of edges whose deletion results in a forest [5].

Contribution

The goal of this paper is to provide new insight into the complexity of BDD parameterized by the structure of the input graph. Our first main result shows that BDD is W[1]-hard parameterized by the *feedback vertex set number*, i.e., the minimum number of vertices whose deletion results in a forest. This resolves the main open question in [5]. Interestingly, our result is significantly stronger since we show that hardness even applies in the case that the remaining parts, after deleting the feedback vertex set, are trees of height three. This rules out fixed-parameter algorithms w.r.t. most of the remaining "classical" decomposition-based structural parameters such as *pathwidth* and *treedepth* [32] as well as w.r.t. the *vertex deletion distance* [19, 32] to bounded pathwidth, treedepth, and treewidth. On the way to our hardness result we show hardness for several multidimensional variants of the classical subset sum problem parameterized by the number of dimensions, which we believe are interesting on their own.

In light of the above, it is natural to ask whether there exist natural decomposition-based parameters for which BDD is fixed-parameter tractable. Our main algorithmic result answers this question affirmatively: we obtain a fixed-parameter algorithm utilizing the recently introduced structural parameter called *treecut width*. The importance of treecut width is that it plays a similar role with respect to the fundamental graph operation of immersion as the graph parameter *treewidth* plays with respect to the minor operation [36, 29]. Up to now, only a handful of problems are known to be FPT when parameterized by treecut width but W[1]-hard when parameterized by treewidth [20]. Furthermore, unlike previously known algorithms using treecut width, this is the first of its kind which does not use an Integer Linear Programming formulation but instead relies purely on combinatorial arguments.

Our second algorithmic result focuses on structural parameters which are not based on any particular decomposition of the graph, but instead measure the "vertex-deletion distance" to a certain graph property. Such structural parameters have been successfully used in the past for a plethora of other difficult problems [19, 22, 27, 15, 14, 21]. In this context and taking into account the strong lower bounds obtained in Section 3, we introduce a structural parameter which is specifically tailored to BDD and which we call the *core fracture number*. Roughly speaking, the core fracture number k is the vertex deletion distance to a graph where each connected component only contains at most k vertices which exceed the degree bound d. We show that computing the core fracture number is FPT which in turn gives rise to a fixed-parameter algorithm for BDD; the latter is achieved by identifying and formalizing a *type-aggregation condition*, allowing for an encoding of the problem into an Integer Linear Program with a controlled number of integer variables. This also resolves the question from [5] if BDD is FPT parametrized by vertex cover.

Finally, we exclude the existence of a *polynomial kernel* [13, 9] for BDD parameterized by the treecut width and core fracture number, and compare the two parameters in Section 5.

2 Preliminaries

2.1 Basic Notation

We use standard terminology for graph theory, see for instance [12]. All graphs except for those used to compute the torso-size in Subsection 2.3 are simple; the multigraphs used in Subsection 2.3 have loops, and each loop increases the degree of the vertex by 2.

Let G be a graph. We denote by $V(G)$ and $E(G)$ its vertex and edge set, respectively. For a vertex $v \in V(G)$, let $N_G(v) = \{y \in V(G) : vy \in E(G)\}$, $N_G[v] = N_G(v) \cup \{v\}$, and $\deg_G(v)$ denote its open neighborhood, closed neighborhood, and degree, respectively. For a subset $X \subseteq V(G)$, the (open) neighborhood $N_G(X)$ of X is defined as $\bigcup_{x \in X} N(x) \setminus X$. The set $N_G[X]$ refers to the closed neighborhood of X defined as $N_G(X) \cup X$. We refer to the set $N_G(V(G) \setminus X)$ as $\partial_G(X)$; this is the set of vertices in X which have a neighbor in $V(G) \setminus X$. We omit the lower index G, if G is clear from the context. For a vertex set A, we use $G - A$ to denote the graph obtained from G by deleting all vertices in A. We use $[i]$ to denote the set $\{0, 1, \ldots, i\}$. For completeness, we provide a formal definition of our problem of interest below.

Bounded-Degree Vertex Deletion (BDD)	
Input:	An undirected graph $G = (V, E)$ and integers $d \geq 0$ and $\ell \geq 0$.
Question:	Is there a subset $V' \subseteq V$ with $\lvert V' \rvert \leq \ell$ whose removal from G yields a graph in which each vertex has degree at most d?

2.2 Parameterized Complexity

A *parameterized problem* $\mathcal{P}$ is a subset of $\Sigma^* \times \mathbb{N}$ for some finite alphabet Σ. Let $L \subseteq \Sigma^*$ be a classical decision problem for a finite alphabet, and let p be a non-negative integer-valued function defined on Σ^*. Then L *parameterized by* p denotes the parameterized problem $\{ (x, p(x)) \mid x \in L \}$ where $x \in \Sigma^*$. For a problem instance $(x, k) \in \Sigma^* \times \mathbb{N}$ we call x the main part and k the parameter. A parameterized problem $\mathcal{P}$ is *fixed-parameter tractable* (FPT in short) if a given instance (x, k) can be solved in time $\mathcal{O}(f(k) \cdot p(|x|))$ where f is an arbitrary computable function of k and p is a polynomial function; we call algorithms running in this time *fixed-parameter algorithms*. We refer the reader to [13] for more details on parameterized complexity.

Parameterized complexity classes are defined with respect to *fpt-reducibility*. A parameterized problem P is *fpt-reducible* to Q if in time $f(k) \cdot |x|^{O(1)}$, one can transform an instance (x, k) of $\mathcal{P}$ into an instance (x', k') of $\mathcal{Q}$ such that $(x, k) \in \mathcal{P}$ if and only if $(x', k') \in \mathcal{Q}$, and $k' \leq g(k)$, where f and g are computable functions depending only on k. Central to parameterized complexity is the following hierarchy of complexity classes, defined by the closure of canonical problems under fpt-reductions: $\mathsf{FPT} \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \cdots \subseteq \mathsf{XP}$. All inclusions are believed to be strict. In particular, $\mathsf{FPT} \neq \mathsf{W}[1]$ under the Exponential Time Hypothesis [23].

The class $\mathsf{W}[1]$ is the analog of NP in parameterized complexity. A major goal in parameterized complexity is to distinguish between parameterized problems which are in FPT and those which are $\mathsf{W}[1]$-hard, i.e., those to which every problem in $\mathsf{W}[1]$ is fpt-reducible. There are many problems shown to be complete for $\mathsf{W}[1]$, or equivalently $\mathsf{W}[1]$-complete, including the Multi-Colored Clique (MCC) problem [13].

2.3 Treecut Width

The notion of treecut decompositions was first proposed by Wollan [36], see also [29]. A family of subsets $X_1, \ldots, X_k$ of X is a *near-partition* of X if they are pairwise disjoint and $\bigcup_{i=1}^{k} X_i = X$, allowing the possibility of $X_i = \emptyset$.

▶ **Definition 1.** A *treecut decomposition* of G is a pair $(T, \mathcal{X})$ which consists of a rooted tree T and a near-partition $\mathcal{X} = \{X_t \subseteq V(G) : t \in V(T)\}$ of $V(G)$. A set in the family $\mathcal{X}$ is called a *bag* of the treecut decomposition.

For any node t of T other than the root r, let $e(t) = ut$ be the unique edge incident to t on the path to r. Let T_u and T_t be the two connected components in $T - e(t)$ which contain u and t, respectively. Note that $(\bigcup_{q \in T_u} X_q, \bigcup_{q \in T_t} X_q)$ is a near-partition of $V(G)$, and we use $\mathbf{cut}(t)$ to denote the set of edges with one endpoint in each part. We define the *adhesion* of t ($\mathbf{adh}_T(t)$ or $\mathbf{adh}(t)$ in brief) as $|\mathbf{cut}(t)|$; if t is the root, we set $\mathbf{adh}_T(t) = 0$ and $\mathbf{cut}(t) = \emptyset$.

The *torso* of a treecut decomposition $(T, \mathcal{X})$ at a node t, written as H_t, is the graph obtained from G as follows. If T consists of a single node t, then the torso of $(T, \mathcal{X})$ at t is G. Otherwise let $T_1, \ldots, T_\ell$ be the connected components of $T - t$. For each $i = 1, \ldots, \ell$, the vertex set $Z_i \subseteq V(G)$ is defined as the set $\bigcup_{b \in V(T_i)} X_b$. The torso H_t at t is obtained from G by *consolidating* each vertex set Z_i into a single vertex z_i (this is also called *shrinking* in the literature). Here, the operation of consolidating a vertex set Z into z is to substitute Z by z in G, and for each edge e between Z and $v \in V(G) \setminus Z$, adding an edge zv in the new graph. We note that this may create parallel edges.

The operation of *suppressing* (also called *dissolving* in the literature) a vertex v of degree at most 2 consists of deleting v, and when the degree is two, adding an edge between the neighbors of v. Given a connected graph G and $X \subseteq V(G)$, let the *3-center* of (G, X) be the unique graph obtained from G by exhaustively suppressing vertices in $V(G) \setminus X$ of degree at most two. Finally, for a node t of T, we denote by $\tilde{H}_t$ the 3-center of (H_t, X_t), where H_t is the torso of $(T, \mathcal{X})$ at t. Let the *torso-size* $\mathbf{tor}(t)$ denote $|\tilde{H}_t|$.

▶ **Definition 2.** The width of a treecut decomposition $(T, \mathcal{X})$ of G is $\max_{t \in V(T)} \{\mathbf{adh}(t), \mathbf{tor}(t)\}$. The treecut width of G, or $\mathbf{tcw}(G)$ in short, is the minimum width of $(T, \mathcal{X})$ over all treecut decompositions $(T, \mathcal{X})$ of G.

We conclude this subsection with some notation related to treecut decompositions. Given a tree node t, let T_t be the subtree of T rooted at t. Let $Y_t = \bigcup_{b \in V(T_t)} X_b$, and let G_t denote

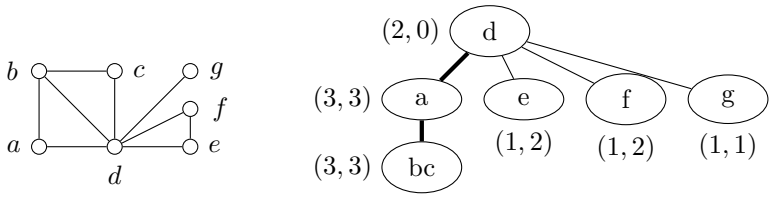

Figure 1 A graph G and a width-3 treecut decomposition of G, including the torso-size (left value) and adhesion (right value) of each node.

the induced subgraph $G[Y_t]$. The *depth* of a node t in T is the distance of t from the root r. The vertices of $\partial_t = \partial_G(Y_t)$ are called the *border* at node t. A node $t \neq r$ in a rooted treecut decomposition is *thin* if $\mathbf{adh}(t) \leq 2$ and *bold* otherwise. For a node t, we let B_t and A_t denote the set of children of t which are thin and bold, respectively.

While it is not known how to compute optimal treecut decompositions efficiently, there exists a fixed-parameter 2-approximation algorithm which fully suffices for our purposes.

▶ **Theorem 3** ([24]). *There exists an algorithm that takes as input an n-vertex graph G and integer k, runs in time $2^{\mathcal{O}(k^2 \log k)} n^2$, and either outputs a treecut decomposition of G of width at most $2k$ or correctly reports that $\mathbf{tcw}(G) > k$.*

A treecut decomposition $(T, \mathcal{X})$ is *nice* if it satisfies the following condition for every thin node $t \in V(T)$: $N(Y_t) \cap \bigcup_{b \text{ is a sibling of } t} Y_b = \emptyset$. The intuition behind nice treecut decompositions is that we restrict the neighborhood of thin nodes in a way which facilitates dynamic programming.

▶ **Lemma 4** ([20]). *There exists a cubic-time algorithm which transforms any rooted treecut decomposition $(T, \mathcal{X})$ of G into a nice treecut decomposition of the same graph, without increasing its width or number of nodes.*

The following property of nice treecut decompositions will be crucial for our algorithm.

▶ **Lemma 5** ([20]). *Let t be a node in a nice treecut decomposition of width k. Then $|A_t| \leq 2k + 1$.*

We refer to previous work [20] for a comparison of treecut width to other parameters.

3 Hardness Results

In this section we show that BDD is W[1]-hard parameterized by a vertex deletion set to trees of height at most three, i.e., a subset D of the vertices of the graph such that every component in the graph, after removing D, is a tree of height at most three. On the way towards this result, we provide hardness results for several interesting versions of the multidimensional subset sum problem (parameterized by the number of dimensions) which we believe are interesting in their own right. In particular, we note that the hardness results also hold for the well-known and more general multidimensional knapsack problem [18].

Our first auxiliary result shows hardness for the following problem.

Multidimensional Subset Sum (MSS)	
Input:	An integer k, a set $S = \{s_1, \ldots, s_n\}$ of item-vectors with $s_i \in \mathbb{N}^k$ for every i with $1 \leq i \leq n$ and a target vector $t \in \mathbb{N}^k$.
Parameter:	k
Question:	Is there a subset $S' \subseteq S$ such that $\sum_{s \in S'} s = t$?

▶ **Lemma 6.** MSS *is* W[1]*-hard even if all integers in the input are given in unary.*

Proof sketch. The proof is by a parameterized reduction from the well-known W[1]-hard Multicolored Clique (MCC) problem [13]: given a k-partite graph G with partition $V_1, \ldots, V_k$, decide whether G contains a clique of size k. For an instance $\mathcal{I} = (G, k)$ of MCC we construct an equivalent instance $\mathcal{I}' = (2\binom{k}{2} + k, S, t)$ of MSS in polynomial time, as follows. For every $v \in V(G)$ we construct one item-vector s_v in S and for every $e \in E(G)$ one item-vector s_e. Furthermore, we impose the following requirements on every solution $S' \subseteq S$ of $\mathcal{I}'$: (1) exactly one vector s_v with $v \in V_i$ is contained in S' for every i with $1 \leq i \leq k$, (2) exactly one vector s_e, with e being an edge between V_i and V_j, is contained in S' for every i and j with $1 \leq i < j \leq k$, and (3) for every edge e with $s_e \in S'$ and endpoints $v_i \in V_i$, $v_j \in V_j$ we find $s_{v_i}, s_{v_j} \in S'$. To ensure (1), the target vector has k entries with value one and every vector s_v with $v \in V_i$ has value one at the i-th of those entries. Property (2) is ensured in a similar way by using $\binom{k}{2}$ entries with value one in the target vector. To ensure Property (3), we assign to every vertex v of G a unique number $\mathcal{S}(v)$ from a Sidon sequence $\mathcal{S}$ of length $|V(G)|$ [16]. A Sidon sequence is a sequence of natural numbers such that the sum of each pair of numbers is unique; it can be shown that it is possible to construct such sequences whose maximum value is bounded by a polynomial in its length [1, 16]. The target vector then contains one additional entry $I(i, j)$ for every i and j with $1 \leq i < j \leq k$ with value $\max_2(\mathcal{S}) + 1$, where $\max_2(\mathcal{S})$ is the maximum sum of any two numbers in $\mathcal{S}$. Moreover, every vector s_v for $v \in V(G)$ has value $\mathcal{S}(v)$ at every entry $I(l, r)$ with $l = i$ or $r = i$ and similarly every vector s_e for an edge e between V_i and V_j has value $(\max_2(\mathcal{S}) + 1) - (\mathcal{S}(u) + \mathcal{S}(v))$ at entry $I(i, j)$. Then, because $\mathcal{S}$ is a Sidon sequence, it holds that the $I(i, j)$-th entry of $\sum_{s \in S'} s$ for a solution S' is equal to the $I(i, j)$-th entry of t if and only if the endpoints of the unique edge chosen between V_i and V_j are equal to the unique vertices v_i and v_j chosen in V_i and V_j, respectively. ◀

The proof of the above lemma also implies hardness for the following slightly adapted version of MSS, which we call the Restricted Multidimensional Subset Sum (RMSS) problem. For RMSS an additional integer k' is given (which will be part of the parameter) and we ask for a solution of the MSS problem of size exactly k'. Before presenting our hardness result for BDD, we need to show hardness for the following more relaxed version of RMSS, which we call the Multidimensional Relaxed Subset Sum (MRSS) problem. For MRSS both the input as well as the parameters are the same as in the case of RMSS however one now asks whether there is a subset $S' \subseteq S$ with $|S'| \leq k'$ such that $\sum_{s \in S'} s \geq t$.

▶ **Lemma 7.** MRSS *is* W[1]*-hard even if all integers in the input are given in unary.*

We are now ready to show our main hardness result for BDD using a reduction from MRSS.

▶ **Theorem 8.** BDD *is* W[1]*-hard parameterized by the size of a vertex deletion set into trees of height at most* 3*.*

Proof Sketch. We prove the theorem by a parameterized reduction from MRSS. Namely, given an instance $\mathcal{I} = (k, S, t, k')$ of MRSS we construct an equivalent instance $\mathcal{I}' = (G, d, \ell)$ of BDD such that G has a FVS D of size $k \cdot k'$. The core idea of the reduction relies on transforming the decision of whether to select a vector into a solution S' for $\mathcal{I}$ into the decision of whether to resolve a tree gadget in G in one of two possible ways.

The set D consists of $(k' + 1)$ vertices $d_i^1, \ldots, d_i^{k'+1}$ for every i with $1 \leq i \leq k$. Moreover, for every $s \in S$ we introduce the gadget $G(s)$ defined as follows. $G(s)$ consists of $\max(s)$ stars with centers $c_1^s, \ldots, c_{\max(s)}^s$ and $d + 1$ leaves. For every star with center c_i^s, we denote by l_i^s one of its leaves (chosen arbitrarily). Additionally, $G(s)$ has a root vertex, denoted

by r^s, that has an edge to every center vertex c_i^s. Finally, we add edges between the leaves $l_1^s, \ldots, l_{\max(s)}^s$ and the vertices in D such that for every i and j with $1 \leq i \leq k$ and $1 \leq j \leq k'+1$, it holds that d_i^j has $s[i]$ neighbors among the leaves $l_1^s, \ldots, l_{\max(s)}^s$ of $G(s)$. Clearly this is always possible and can be done in an arbitrary manner.

We set d to be the maximum degree of the part of G constructed so far (note that this maximum is reached by one of the vertices in D). Moreover, we now ensure that for every i and j with $1 \leq i \leq k$ and $1 \leq j \leq k'+1$, the vertex d_i^j has degree $d + t[i]$ in G by attaching a appropriate number of leaves to d_i. Finally, we set ℓ to be $\left(\sum_{s \in S} \max(s)\right) + k'$. This completes the construction of $\mathcal{I}'$. Clearly, $\mathcal{I}'$ can be constructed in polynomial time. Moreover, $|D| \leq k \cdot k'$ and each component of $G - D$ is a tree with height at most 3. To complete the proof, it suffices to establish the equivalence between $\mathcal{I}$ and $\mathcal{I}'$. ◀

Clearly trees of height at most three are trivially acyclic. Moreover, it is easy to verify that such trees have pathwidth [25] and treedepth [32] at most three, which implies:

▶ **Corollary 9.** BDD *is* W[1]*-hard parameterized by any of the following parameters:*

- *the size of a feedback vertex set,*
- *the pathwidth and treedepth of the input graph,*
- *the size of a minimum set of vertices whose deletion results in components of pathwidth/treedepth at most three.*

4 Solving BDD using Treecut-width

The goal of this section is to provide a fixed-parameter algorithm for BDD parameterized by treecut-width. The core of the algorithm is a dynamic programming procedure which runs on a nice treecut decomposition of the input graph. First we define the data table the algorithm is going to dynamically compute for individual nodes of the treecut decomposition. For each node $t \in T$, the table is going to contain two components, which we will call the *universal cost* u_t and the *specific cost* s_t. Informally, the universal cost captures the minimum number of vertices which need to be deleted from Y_t to satisfy the degree bound in G_t. The specific cost captures how many more vertices (than the universal cost) we need to delete in order to satisfy the degree bound in G_t when we also place restrictions on how G_t will interact with the rest of the graph. We formalize these notions below.

Let us fix an instance (G, d, ℓ) of BDD and a treecut decomposition $(T, \mathcal{X})$ of G of width at most k and rooted at r. A *configuration* δ of a graph H with a designated vertex-subset Z is a mapping $Z \mapsto [k] \cup \text{del}$. Intuitively, configurations are going to be used to place additional restrictions on the deletion sets we are interested in. We let $\mathbf{bdd}(H, Z, \delta)$ denote the minimum size of a vertex set $W \subseteq V(H)$ such that:

- $v \in W \cap Z$ if and only if $\delta(v) = \text{del}$, and
- for each $v \in Z \setminus W$, the degree of v in $H - W$ is at most $d - \delta(v)$,
- for each $v \in V(H) \setminus (Z \cup W)$, the degree of v in $H - W$ is at most d.

Figure 2 depicts an illustration of $\mathbf{bdd}(H, Z, \delta)$. Informally, **bdd** captures the size of a minimum deletion set which intersects the designated subset precisely in the vertices specified by δ, and for the remainder of the designated subset it overshoots the degree bound by a buffer specified by δ. If $\mathbf{bdd}(H, Y, \delta)$ is not defined (which may happen, e.g., if $d < |Y|$), we formally set $\mathbf{bdd}(H, Y, \delta) = \infty$. For each node $t \in V(T)$, we can now define:

- $u_t = \mathbf{bdd}(G_t, \emptyset, \emptyset)$, and
- for each $\delta : \partial_t \to [k] \cup \text{del}$ such that each $v \in \partial_t$ is mapped to del or to an integer $i \leq |N(v) \setminus Y_t|$, we let $s'_t(\delta) = \mathbf{bdd}(G_t, \partial_t, \delta) - u_t$.

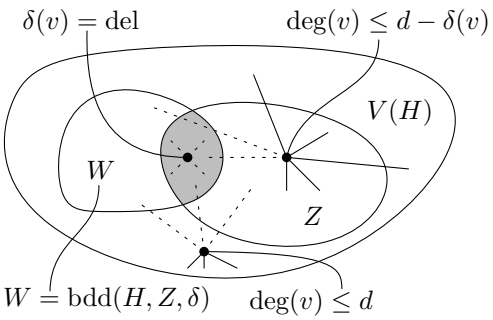

Figure 2 Illustration of the set $\mathbf{bdd}(H, Z, \delta)$. The dotted edges are not considered for the degree of a node v.

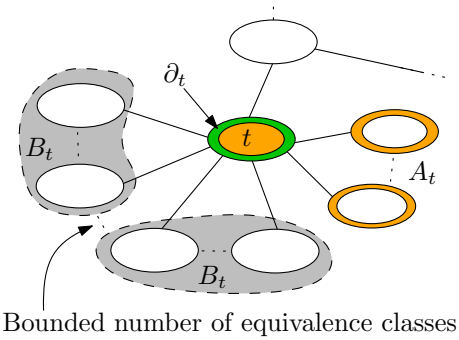

Figure 3 The three branching sets for a node $t \in T$, first branch on ∂_t (green), then on the boundaries of the bold nodes A_t together with the "interior" of t (orange) and finally on the equivalence classes of B_t (gray).

We proceed with a few observations. Naturally, the value of u_t can be much larger than k (as an example, consider a collection of disjoint stars), and this is not an issue for our algorithm. Furthermore, for every δ it holds that $0 \leq s'_t(\delta)$, since $u_t \leq \mathbf{bdd}(G_t, \partial_t, \delta)$; notice that u_t attains the value of the smallest deletion set for G_t, while $\mathbf{bdd}(G_t, \partial_t, \delta)$ attains the value of a smallest deletion set for G_t which satisfies certain additional restrictions.

Crucially, the value of $s'_t(\delta)$ can be much larger than k, and this represents a significant obstacle for our algorithm. The role of the specific cost in the dynamic programming procedure is to capture how a node may interact with the solution and how such interactions affect the size of a deletion set. The algorithm relies heavily on having only a bounded number of possible interactions in order to achieve its run-time bounds. Luckily, we will prove that any value of $s'_t(\delta)$ exceeding k must lead to a dead end and can be disregarded.

▶ **Lemma 10.** *Let S be a minimum-size bounded degree deletion set in G. Let δ^t_S be defined over ∂_t as follows: $\delta^t_S(v) = \text{del}$ if $v \in S$, and otherwise $\delta^t_S(v) = |(N(v) \setminus Y_t) \setminus S|$. Then $s'_t(\delta^t_S) \leq |N(Y_t)| \leq k$.*

Proof. For brevity, let $q = N(Y_t)$. The fact that $q \leq k$ follows immediately from the bound on the adhesion of t, hence we only need to prove that $s'_t(\delta^t_S) \leq q$. So, assume for a contradiction that $s'_t(\delta^t_S) > q$. Let P be a witness for the value of u_t, i.e., let P be a minimum-cardinality vertex subset of G_t such that the maximum degree in $G_t - P$ is at most d. Observe that $|P \cup N(Y_t)| = u_t + q$. Now consider the set $S' = (S \setminus Y_t) \cup P \cup N(Y_t)$. First of all, note that $|S'| < |S|$, since we obtained S' from S by removing more than $u_t + q$ vertices (recall that, by our assumption, $s'_t(\delta^t_S) > q$) and then adding back at most $u_t + q$ vertices. Second, we claim that S' is also a bounded degree deletion set in G. Indeed, consider for a contradiction that $G - S'$ contains a vertex v of degree higher than d. Such a v cannot lie in Y_t since P was a solution in G_t and $N(Y_t)$ separates G_t from the rest of G. On the other hand, v cannot lie outside of Y_t due to the fact that S itself was a solution in $G[V(G) - Y_t]$. So the claim holds, and S' contradicts the optimality of S. ◀

Thanks to Lemma 10, we can safely focus our attention on those configurations δ where $s'_t(\delta) \leq |N(Y_t)|$. Hence we define $s_t(\delta) = s'_t(\delta)$ if $s'_t(\delta) \leq |N(Y_t)|$ and $s'_t(\delta) = \infty$ otherwise. Observe that, unlike s'_t, the number of distinct possibilities of what a special cost s_t may look like is bounded by a function of k. The high-level strategy for the algorithm is now the following:

1. Compute (u_t, s_t) when t is a leaf,
2. Compute (u_t, s_t) when t is not a leaf, but the universal and specific costs are known for all of its children, and
3. Use the values (u_r, s_r) at the root node $r \in T$.

As we will see below, points **1.** and **3.** are straightforward.

▶ **Observation 11.** *(u_t, s_t) can be computed in time at most $2^{\mathcal{O}(k)}$ if t is a leaf.*

Proof. To compute u_t it suffices to exhaustively loop through all vertex subsets $L \subseteq X_t$ and check whether $G_t - L$ has degree at most d. Then u_t is equal to the minimum size of such a subset. To compute s_t, we proceed similarly: for each configuration δ such that each $v \in \partial_t$ is mapped to del or to an integer $i \leq |N(v) \setminus Y_t|$, we exhaustively loop through all $L \subseteq X_t \setminus \partial_t$ in order to determine the value of $\mathbf{bdd}(G_t, \partial_t, \delta)$, and we then use that value and u_t to determine $s_t(\delta)$. ◀

▶ **Observation 12.** *(G, d, ℓ) is a YES-instance of* BDD *if and only if $u_r \leq \ell$.*

Given the above, the last remaining obstacle is handling point (2), i.e., the dynamic propagation of information from leaves to the root. This is also the by far most challenging part of the algorithm. Let us fix a node $t \in T$ and for all its children p we assume (u_p, s_p) to be already computed.

Our strategy is to apply a 2-step approach. Figure 3 shows an illustration of the upcoming branching sets for a node t. Recall that A_t and B_t denote the set of all children of t which are bold and thin, respectively. First, we exhaustively loop over all options of how a deletion set candidate intersects with X_t and the borders of nodes in A_t, resulting in a set of "templates" which provide us with additional information about a potential solution. Here the bound on $|A_t|$ provided in Lemma 5 will be crucial. Second, we use branching and network flows to find an optimal way of extending such a template to a solution which deals with B_t. In this step, we overcome the fact that there may be an unbounded number of children p in B_t by "aggregating" them into types based on their s_p component. Lemma 10 along with our definition of specific costs then guarantees that the number of aggregated types will depend only on k. Informally, if two nodes p_1, p_2 in B_t have the same specific cost, then their behavior ("contribution") to any solution is fully interchangeable. In particular, even if p_1, p_2 have different universal costs, both of these costs will need to be "paid" by every solution regardless of how the solution handles the borders of these nodes. When formalizing the above sketched algorithm we obtain.

▶ **Lemma 13.** *Point* **2.** *can be solved in time $2^{\mathcal{O}(k^2)} \cdot |B_t|^2$, where $|B_t|$ is upper-bounded by the number of children of t.*

▶ **Theorem 14.** BDD *can be solved in time $n^3 + 2^{\mathcal{O}(k^2 \cdot \log k)} \cdot n^2$, where k and n are the treecut-width and number of vertices of the input graph, respectively.*

Proof. We begin by applying Theorem 3 followed by Lemma 4 to obtain a nice treecut decomposition $(T, \mathcal{X})$ of width at most $2k$. We then use a dynamic programming algorithm to compute the values u_t and s_t at every node $t \in T$. For leaves, this is carried out by Observation 11, while for non-leaves we invoke Lemma 13. Finally, once we compute u_r for the root r, we can determine the answer to a BDD instance using Observation 12. ◀

Finally, using standard techniques it is not difficult to show that BDD parameterized by treecut width does not admit a polynomial kernel.

▶ **Theorem 15.** *BDD parameterized by treecut width has no polynomial kernel unless* coNP $\subseteq$ NP/poly.

5 Core Fracture Number

In this section we introduce the new structural parameter core fracture number and provide a fixed-parameter algorithm for BDD parameterized by this parameter. An important prerequisite for the introduction of this parameter is the following simple preprocessing procedure that can be applied to any BDD instance. Given an instance $\mathcal{I} = (G, d, \ell)$ of BDD, the *core* of $\mathcal{I}$, denoted by $\mathbf{c}(\mathcal{I}) = (\mathbf{c}(G), d, \ell)$, is the BDD instance obtained from $\mathcal{I}$ after removing all edges whose both endpoints have degree at most d from G.

▶ **Observation 16.** *Let $\mathcal{I} = (G, d, \ell)$ be a BDD instance. Then $\mathcal{I}$ and $\mathbf{c}(\mathcal{I})$ are equivalent instances of BDD in the sense that any solution for $\mathcal{I}$ is also a solution for $\mathbf{c}(\mathcal{I})$ and vice versa. Moreover, $\mathbf{c}(\mathcal{I})$ can be computed in linear time w.r.t. the number of edges of G.*

In the following we will assume that we have already applied the above preprocessing procedure to any BDD instance and hence the graph of the instance does not contain any edges between vertices whose degree is already below the given degree bound. The *core fracture number* of a BDD instance $\mathcal{I} = (G, d, \ell)$, denoted by $\mathbf{cfn}(\mathcal{I})$, is the minimum integer k such that there is a deletion set $D \subseteq V(G)$ with $|D| \leq k$ and the number of vertices in any component C of $G \setminus D$ of degree larger than d in G is at most k. In other words, each connected component of $G - D$ may contain only at most k vertices of degree greater than d. We start by showing that this parameter is orthogonal to treecut width.

▶ **Theorem 17.** *There is a class of BDD instances with bounded treecut width and unbounded core fracture number. Similarly, there is a class of BDD instances with bounded core fracture number and unbounded treecut width. Moreover, both classes only contain BDD instances $\mathcal{I}$ with $\mathbf{c}(\mathcal{I}) = \mathcal{I}$.*

We are now ready to present our fixed-parameter algorithm for BDD parameterized by the core fracture number. The algorithm consists of two steps: (1) it computes a deletion set D of size at most k, witnessing that $\mathbf{cfn}(\mathcal{I}) \leq k$ and (2) it solves $\mathcal{I}$ with the help of the deletion set D. Namely, our algorithm will consists of fixed-parameter algorithms for the following two parameterized problems. Given an instance $\mathcal{I} = (G, d, \ell)$ of BDD and an integer k (which also serves as the parameter of the problem), the Core Fracture Number Detection (CFND) problem, asks whether $\mathbf{cfn}(\mathcal{I}) \leq k$ and if so outputs a set $D \subseteq V(G)$ witnessing this. On the other hand the Core Fracture Number Evaluation (CFNE) problem asks whether $\mathcal{I}$ has a solution for a given BDD instance $\mathcal{I} = (G, d, \ell)$ and a set $D \subseteq V(G)$ witnessing that $\mathbf{cfn}(\mathcal{I}) \leq |D|$ and is parameterized by $|D|$.

▶ **Theorem 18.** *CFND can be solved in time $\mathcal{O}((2k+1)^k|E(G)|)$ and is hence fixed-parameter tractable. Moreover, CFND can be approximated in polynomial time within a factor of $2k+1$.*

Let $\mathcal{I} = (G, d, \ell, D)$ be an instance of CFNE and assume w.l.o.g. that $\mathbf{c}(G) = G$. We start by showing that we do not need to consider solutions $V' \subseteq V(G)$ for $\mathcal{I}$ that contain more than $2k - 1$ vertices from any component C of $G - D$.

▶ **Lemma 19.** *If $\mathcal{I}$ has a solution, then it has a solution V' such that $|V' \cap V(C)| < 2k$ for every component C of $G - D$.*

Proof. Let V' be a solution for $\mathcal{I}$ and C be a component of $G - D$ with $|V' \cap V(C)| \geq 2k$; if no such component exists, then we are done. Let M be the set of all vertices in C, whose degree is larger than d in G. Then $(V' \setminus V(C)) \cup M \cup D$ is also a solution for $\mathcal{I}$ and moreover $|(V' \setminus V(C)) \cup M \cup D| \leq |V'| - 2k + k + k \leq |V'|$. By iterating the same process for every component C with $|V' \cap V(C)| \geq 2k$, one obtains the desired solution for $\mathcal{I}$. ◀

Let C be a component of $G - D$ and let $M \subseteq V(C)$ be the set of all vertices with degree larger than d in G. Then the *signature* of C, denoted by $\mathcal{S}(C)$, contains all pairs (D', Γ) such that $D' \subseteq D$, and Γ is the set of all pairs (o, γ) such that:

- o is an integer with $0 \leq o < 2k$, and
- $\gamma : D \setminus D' \to \{0, \ldots, 2k-1\}$ is a mapping such that there is a set $V' \subseteq V(C)$ with $|V'| = o$ satisfying the following conditions:
  - **(S1)** every vertex in $M \setminus V'$ has degree at most d in $G - (V' \cup D')$ and
  - **(S2)** for every vertex v in $D \setminus D'$, V' contains exactly $\gamma(v)$ neighbors of v.

Informally, for every subset D' of vertices that we decide to delete from D, the signature tells us how many vertices in C we need to delete and how their deletion affects the degrees of the remaining vertices in $D - D'$. Because we only need to consider solutions containing less than $2k$ vertices from C (Lemma 19), the number of ways in which different solutions effect the degrees of vertices in D is bounded, which allows us to compute the signatures.

▶ **Lemma 20.** *The signature $\mathcal{S}(C)$ can be computed in time $O(|V(C)| + |E(C)| + 2^k(2k)^{2^{2k}})$ for any component C of $G - D$.*

Let $D' \subseteq D$ and C and C' be two distinct components of $G - D$. We say that C and C' are *equivalent w.r.t.* D' if $(D', \Gamma) \in \mathcal{S}(C) \cap \mathcal{S}(C')$ for some Γ. Let $\mathcal{P}(D')$ be the partition of all components of $G - D$ into equivalence classes and for an equivalence class $\mathcal{C} \in \mathcal{P}(D')$ let $\Gamma(\mathcal{C})$ denote the set Γ such that $(D', \Gamma) \in \mathcal{S}(C)$ for every $C \in \mathcal{C}$. Note that $|\mathcal{P}(D')| \leq 2^{k(2k)^k}$.

▶ **Lemma 21.** *An instance $\mathcal{I} = (G, d, \ell, D)$ has a solution if and only if there is a subset D' of D and a mapping α that assigns to every $\mathcal{C} \in \mathcal{P}(D')$ and every $(o, \gamma) \in \Gamma(\mathcal{C})$ a natural number satisfying the following conditions:*

(C1) $(\sum_{\mathcal{C} \in \mathcal{P}(D') \wedge (o,\gamma) \in \Gamma(\mathcal{C})} o \cdot \alpha(\mathcal{C}, (o, \gamma))) + |D'| \leq \ell$, *i.e., the budget ℓ is not exceeded,*
(C2) $\sum_{(o,\gamma) \in \Gamma(\mathcal{C})} \alpha(\mathcal{C}, (o, \gamma)) = |\mathcal{C}|$ *for every $\mathcal{C} \in \mathcal{P}(D')$, i.e., all components are considered,*
(C3) $\sum_{\mathcal{C} \in \mathcal{P}(D') \wedge (o,\gamma) \in \Gamma(\mathcal{C})} \gamma(v) \cdot \alpha(\mathcal{C}, (o, \gamma) \geq |N_{G-D'}(v)| - d$ *for every $v \in D \setminus D'$, i.e., the degree conditions for the vertices in $D \setminus D'$ are satisfied.*

Proof. Towards showing the forward direction let V' be a solution for $\mathcal{I}$. We start by setting $D' = D \cap V'$. Consider a component C of $G - D$ and let Γ be the set such that $(D', \Gamma) \in \mathcal{S}(C)$. Because of Lemma 19, we can assume that $|V' \cap V(C)| < 2k$. Hence Γ contains a pair $(|V' \cap V(C)|, \gamma)$, which we denote by $A(C)$, such that for every $v \in D \setminus D'$, it holds that v has exactly $\gamma(v)$ neighbors in $V' \cap V(C)$. For every $\mathcal{C} \in \mathcal{P}(D')$ and $(o, \gamma) \in \Gamma(\mathcal{C})$, we now set $\alpha(\mathcal{C}, (o, \gamma))$ to be the number of components C in $\mathcal{C}$ with $A(C) = (o, \gamma)$ and claim that α satisfies the conditions (C1)–(C3). Because $(\sum_{\mathcal{C} \in \mathcal{P}(D') \wedge (o,\gamma) \in \Gamma(\mathcal{C})} o \cdot \alpha(\mathcal{C}, (o, \gamma))) + |D'| = |V'|$ and $|V'| \leq \ell$, we obtain that α satisfies (C1). Condition (C2) follows immediately from the definition of α. Finally, Condition (C3) follows because for every $v \in D \setminus D'$ it holds that $\sum_{\mathcal{C} \in \mathcal{P}(D') \wedge (o,\gamma) \in \Gamma(\mathcal{C})} \gamma(v) \cdot \alpha(\mathcal{C}, (o, \gamma))$ is equal to the number of neighbors of v in $V' \setminus D$ and the fact that v can have at most d neighbors in $G - V'$.

Towards showing the reverse direction let $D' \subseteq D$ and α be a mapping satisfying (C1)–(C3). For a component $C \in \mathcal{C}$ and $(o, \gamma) \in \Gamma$, where $\mathcal{C} \in \mathcal{P}(D')$ and $(D', \Gamma) \in \mathcal{S}(C)$, we denote by $V(C, (o, \gamma))$ a subset of $V(C)$ of size o satisfying the conditions (S1) and (S2) in the definition of a signature. Then a solution V' for $\mathcal{I}$ is obtained as follows. For any $\mathcal{C} \in \mathcal{P}(D')$ we take the union of $V(C, (o, \gamma))$ for exactly $\alpha(\mathcal{C}, (o, \gamma))$ components $C \in \mathcal{C}$. Condition (C2) ensures that there are enough components in $\mathcal{C}$ and moreover that this way we use every component exactly once. Finally, we add D' to V'. Because of Condition (C1), we have that $|V'| \leq \ell$. Moreover, because of Condition (C3), we obtain that every vertex in $D \setminus D'$ has degree at most d in $G - V'$. The same holds for every vertex in any component C of $G - D$, because of Property (S1). Hence V' is a solution for $\mathcal{I}$ of size at most ℓ. ◀

With the help of the above lemma, we can express the existence of a solution in terms of the solution of an integer linear program with a bounded number of variables, which in turn can be solved in fpt-time w.r.t. the number of variables [28].

▶ **Theorem 22.** *CFNE is fixed-parameter tractable.*

Proof sketch. Let $\mathcal{I} = (G, d, \ell, D)$ be the given instance of CFNE. The algorithm first computes the signature $\mathcal{S}(C)$ for every component C of $G - D$ according to Lemma 20. It then uses the characterization given in Lemma 21 to decide whether $\mathcal{I}$ has a solution. Namely, for every $D' \subseteq D$ the algorithm constructs an ILP instance $\mathcal{I}'$ whose optimum is at most $\ell - |D'|$ if and only if the BDD instance $\mathcal{I}$ has a solution V' with $V' \cap D = D'$. In accordance with Lemma 21 the ILP instance $\mathcal{I}'$ has one variable, denote by $x_{\mathcal{C},(o,\gamma)}$, for every $\mathcal{C} \in \mathcal{P}(D')$ and $(o, \gamma) \in \Gamma(\mathcal{C})$ and consists of the following constraints:

$$\begin{aligned} \text{minimize} \quad & \sum_{\mathcal{C}\in\mathcal{P}(D'),(o,\gamma)\in\Gamma(\mathcal{C})} o \cdot x_{\mathcal{C},(o,\gamma)} \\ \text{subject to} \quad & \sum_{(o,\gamma)\in\Gamma(\mathcal{C})} x_{\mathcal{C},(o,\gamma)} = |\mathcal{C}| && \forall \mathcal{C} \in \mathcal{P}(D') \\ & \sum_{\mathcal{C}\in\mathcal{P}(D')\wedge(o,\gamma)\in\Gamma(\mathcal{C})} \gamma(v) \cdot x_{\mathcal{C},(o,\gamma)} \geq |N_{G-D'}(v)| - d && \forall v \in D \setminus D' \end{aligned}$$

Observe that there is a one-to-one correspondence between assignments β for the variables in $\mathcal{I}'$ and the assignment α defined in Lemma 21. Moreover, the constraints of $\mathcal{I}'$ ensure Condition (C2) and (C3) and Condition (C1) can be satisfied if and only if the optimum value of $\mathcal{I}'$ is at most $\ell - |D'|$. This completes the description of the algorithm. ◀

As our final result, we show a kernel lower bound for CFNE.

▶ **Theorem 23.** *CFNE has no polynomial kernel unless* coNP ⊆ NP/poly.

Proof sketch. We give a *polynomial parameter transformation* [2, Proposition 1] from the well-known Set Cover problem parameterized by the size of the universe. It is known that Set Cover does not admit a polynomial kernel unless coNP ⊆ NP/poly [10]. Given an instance $\mathcal{I} = (U, \mathcal{F}, k)$ of Set Cover, we construct an instance $\mathcal{I}' = (G, d, \ell, D)$ of CFNE as follows. G has one vertex v_u for every $u \in U$ as well as one vertex w_F for every $F \in \mathcal{F}$. Moreover, G has an edge between a vertex v_u and a vertex w_F if and only if $u \in F$. We set $D = \{ v_u \mid u \in U \}$. Let Δ be the maximum degree of any vertex in G. Then we attach to every vertex in D new leaf vertices such that the degree of every vertex in D becomes $\Delta + 1$. This completes the construction of G. Finally, we set $d = \Delta$ and $\ell = k$. Because $G - D$ is an independent set, this shows that $\mathbf{cfn}(G) \leq k$. To complete the proof, it remains to show that $\mathcal{I}$ has a solution if and only if so does $\mathcal{I}'$. ◀

6 Concluding Notes

Our results close a wide gap in the understanding of the complexity landscape of BDD parameterized by structural parameters. In particular, they not only resolve the main open question from previous work in the area [5], but push the lower bounds significantly further, specifically to deletion distance to trees of bounded depth. Moreover, we identified structural parameterizations which are better suited for the problem at hand and used these to obtain two novel fixed-parameter algorithms for BDD.

References

1 Martin Aigner and Günter M. Ziegler. *Proofs from the Book (3. ed.)*. Springer, 2004.
2 Christer Bäckström, Peter Jonsson, Sebastian Ordyniak, and Stefan Szeider. A complete parameterized complexity analysis of bounded planning. *JCSS*, 81(7):1311–1332, 2015.
3 Balabhaskar Balasundaram, Sergiy Butenko, and Illya V. Hicks. Clique relaxations in social network analysis: The maximum k-plex problem. *Operations Research*, 59(1):133–142, 2011.
4 Balabhaskar Balasundaram, Shyam Sundar Chandramouli, and Svyatoslav Trukhanov. Approximation algorithms for finding and partitioning unit-disk graphs into co-k-plexes. *Optimization Letters*, 4(3):311–320, 2010.
5 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.
6 Hans L. Bodlaender and Babette van Antwerpen-de Fluiter. Reduction algorithms for graphs of small treewidth. *Inf. Comput.*, 167(2):86–119, 2001.
7 Zhi-Zhong Chen, Michael R. Fellows, Bin Fu, Haitao Jiang, Yang Liu, Lusheng Wang, and Binhai Zhu. A linear kernel for co-path/cycle packing. In *Proc. AAIM 2010*, volume 6124 of *LNCS*, pages 90–102. Springer, 2010.
8 Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.
9 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.
10 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.
11 Anders Dessmark, Klaus Jansen, and Andrzej Lingas. The maximum k-dependent and f-dependent set problem. In *Proc. ISAAC 1993*, volume 762 of *LNCS*, pages 88–98. Springer, 1993.
12 Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer Verlag, New York, 2nd edition, 2000.
13 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.
14 Eduard Eiben, Robert Ganian, and Stefan Szeider. Meta-kernelization using well-structured modulators. In Thore Husfeldt and Iyad A. Kanj, editors, *Proc. IPEC 2015*, volume 43 of *LIPIcs*, pages 114–126. Leibniz-Zentrum für Informatik, 2015.
15 Eduard Eiben, Robert Ganian, and Stefan Szeider. Solving problems on graphs of high rank-width. In *Proc. WADS 2015*, volume 9214 of *LNCS*, pages 314–326. Springer, 2015.
16 Paul Erdős and Paul Turán. On a problem of Sidon in additive number theory, and on some related problems. *Journal of the London Mathematical Society*, 1(4):212–215, 1941.
17 Michael R. Fellows, Jiong Guo, Hannes Moser, and Rolf Niedermeier. A generalization of Nemhauser and Trotter's local optimization theorem. *J. Comput. Syst. Sci.*, 77(6):1141–1158, 2011.
18 Arnaud Fréville. The multidimensional 0-1 knapsack problem: An overview. *European Journal of Operational Research*, 155(1):1–21, 2004.
19 Jakub Gajarský, Petr Hlinený, Jan Obdrzálek, Sebastian Ordyniak, Felix Reidl, Peter Rossmanith, Fernando Sánchez Villaamil, and Somnath Sikdar. Kernelization using structural parameters on sparse graph classes. *J. Comput. Syst. Sci.*, 84:219–242, 2017.
20 Robert Ganian, Eun Jung Kim, and Stefan Szeider. Algorithmic applications of tree-cut width. In Giuseppe F. Italiano, Giovanni Pighizzini, and Donald Sannella, editors, *Proc. MFCS 2015*, volume 9235 of *LNCS*, pages 348–360. Springer, 2015.

21 Robert Ganian, Friedrich Slivovsky, and Stefan Szeider. Meta-kernelization with structural parameters. *J. Comput. Syst. Sci.*, 82(2):333–346, 2016.

22 Serge Gaspers, Neeldhara Misra, Sebastian Ordyniak, Stefan Szeider, and Stanislav Zivny. Backdoors into heterogeneous classes of SAT and CSP. *J. Comput. Syst. Sci.*, 85:38–56, 2017.

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

24 Eunjung Kim, Sang-il Oum, Christophe Paul, Ignasi Sau, and Dimitrios M. Thilikos. An FPT 2-approximation for tree-cut decomposition. In Laura Sanità and Martin Skutella, editors, *Proc. WAOA 2015*, volume 9499 of *LNCS*, pages 35–46. Springer, 2015.

25 Ton Kloks. *Treewidth: Computations and Approximations*, volume 842 of *LNCS*. Springer Verlag, Berlin, 1994.

26 Christian Komusiewicz, Falk Hüffner, Hannes Moser, and Rolf Niedermeier. Isolation concepts for efficiently enumerating dense subgraphs. *Theor. Comput. Sci.*, 410(38-40):3640–3654, 2009.

27 Martin Kronegger, Sebastian Ordyniak, and Andreas Pfandler. Variable-deletion backdoors to planning. In *Proc. AAAI 2015*, pages 2300–2307. AAAI Press, 2014.

28 H. W. Lenstra. Integer programming with a fixed number of variables. *MATH. OPER. RES*, 8(4):538–548, 1983.

29 Dániel Marx and Paul Wollan. Immersions in highly edge connected graphs. *SIAM J. Discrete Math.*, 28(1):503–520, 2014.

30 Benjamin McClosky and Illya V. Hicks. Combinatorial algorithms for the maximum k-plex problem. *J. Comb. Optim.*, 23(1):29–49, 2012.

31 Hannes Moser, Rolf Niedermeier, and Manuel Sorge. Exact combinatorial algorithms and experiments for finding maximum k-plexes. *J. Comb. Optim.*, 24(3):347–373, 2012.

32 Jaroslav Nešetřil and Patrice Ossona de Mendez. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012.

33 Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*, volume 31 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press, Oxford, 2006.

34 Naomi Nishimura, Prabhakar Ragde, and Dimitrios M. Thilikos. Fast fixed-parameter tractable algorithms for nontrivial generalizations of vertex cover. *Discrete Applied Mathematics*, 152(1-3):229–245, 2005.

35 Stephen B Seidman and Brian L Foster. A graph-theoretic generalization of the clique concept. *Journal of Mathematical sociology*, 6(1):139–154, 1978.

36 Paul Wollan. The structure of graphs not admitting a fixed immersion. *J. Comb. Theory, Ser. B*, 110:47–66, 2015.