On the boolean-width of a graph: structure and applications[⋆]

I. Adler[1], B.-M. Bui-Xuan[1], Y. Rabinovich[2], G. Renault[1], J. A. Telle[1], and M. Vatshelle[1]

[1] Department of Informatics, University of Bergen, Norway
[2] Department of Computer Science, Haifa University, Israel

Abstract Boolean-width is a recently introduced graph invariant. Similar to tree-width, it measures the structural complexity of graphs. Given any graph G and a decomposition of G of boolean-width k, we give algorithms solving a large class of vertex subset and vertex partitioning problems in time $O^*(2^{O(k^2)})$. We relate the boolean-width of a graph to its branch-width and to the boolean-width of its incidence graph. For this we use a constructive proof method that also allows much simpler proofs of similar results on rank-width by Oum (JGT 2008). For a random graph on n vertices we show that almost surely its boolean-width is $\Theta(\log^2 n)$ – setting boolean-width apart from other graph invariants – and it is easy to find a decomposition witnessing this. Combining our results gives algorithms that on input a random graph on n vertices will solve a large class of vertex subset and vertex partitioning problems in quasi-polynomial time $O^*(2^{O(\log^4 n)})$.

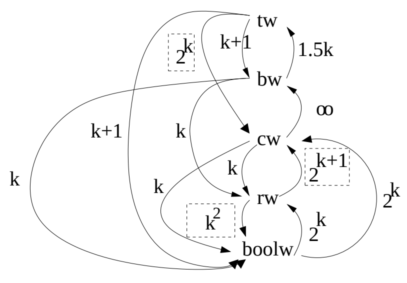
1 Introduction

Width parameters of graphs, like tree-width, branch-width, clique-width and rank-width, have many applications in the field of graph algorithms and especially in FPT algorithmics, see *e.g.* Downey and Fellows [7], Flum and Grohe [8], and Hliněný et al. [11]. When comparing such parameters, we should consider the values of the parameters on various graph classes, the runtime of algorithms for finding a decomposition, the classes of problems that can be solved by dynamic programming along such a decomposition, and the runtime of these algorithms. Recently, Bui-Xuan et al. [3] introduced a new width parameter of graphs called boolean-width. While rank-width is based on the number of GF(2)-sums $(1+1=0)$ of rows of adjacency matrices, boolean-width is based on the number of Boolean sums of rows $(1+1=1)$. Although is it open whether computing boolean-width is FPT, the number of Boolean sums of rows for a matrix is easy to compute in FPT time by an incremental approach. Surprisingly, this number is the same for the matrix and its transpose.

In this paper we study the structure of graphs of bounded boolean-width and we give new algorithmic applications. Given a decomposition tree of boolean-width k of any graph, we give algorithms solving a large class of vertex subset and vertex partitioning problems in time $O^*(2^{O(k^2)})$. Then, we show that the boolean-width of G is at most its branch-width. For a random graph on n vertices we show that almost surely its boolean-width is $\Theta(\log^2 n)$, and it is easy to find the corresponding decomposition tree. Combining our results gives algorithms that on input a random graph on n vertices will solve a large class of vertex subset and vertex partitioning problems in quasi-polynomial time $O^*(2^{O(\log^4 n)})$.

It is well-known that for any class of graphs tree-width is bounded if and only if branch-width is bounded: we say the two parameters are equivalent. Likewise, clique-width, rank-width, and boolean-width are equivalent. For any graph class we have only three possibilities: either all five parameters are bounded (*e.g.* for trees) or none of them are bounded (*e.g.* for grids) or only clique-width, rank-width and boolean-width are bounded (*e.g.* for cliques). Note that the information in Figure 1 allows for a finer comparison. Let us say that parameter P is polylog on a class of graphs C if the value of P for any graph G in C is polylogarithmic in the size of G. Then if P is polylog on C any algorithm with FPT runtime single exponential in parameter P runs in quasi-polynomial time on input a graph in C. From Figure 1 we see that if any

[⋆] Supported by the Norwegian Research Council, projects PARALGO and Graph Searching.



	tree-width	branch-width	clique-width	rank-width	boolean-width
MDS	$O^*(2^{1.58tw})$[21]	$O^*(2^{2bw})$ [6]	$O^*(2^{4cw})$ [16]	$O^*(2^{0.75rw^2+O(rw)})$ [2,9]	$O^*(2^{3boolw})$ [3]

Figure 1. Upper bounds tying parameters tw=tree-width, bw=branch-width, cw=clique-width, rw=rank-width and $boolw$=boolean-width, and runtimes achievable for Minimum Dominating Set using various parameters. In the upper part of the figure, an arrow from P to Q labelled $f(k)$ means that any class of graphs having parameter P at most k will have parameter Q at most $f(k)$, and ∞ means that no such upper bound can be shown. Except for the labels in a box the bounds are known to be tight, meaning that there is a class of graphs for which the bound is $\Omega(f(k))$. For the two boxes containing labels 2^k and 2^{k+1}, a $\Omega(2^{k/2})$ bound is known [4]. For the one containing label k^2 a $\Omega(k)$ bound is known [3]. The arrows $bw \rightarrow boolw$ and $tw \rightarrow boolw$ are shown in Theorem 3 of this paper.

of tree-width, branch-width, clique-width or rank-width is polylog on a class of graphs then so is boolean-width, while the random graphs give an example of a class where boolean-width is polylog but none of the other parameters are. An even finer comparison can be made by looking at the bounds between the parameters in combination with the runtimes achievable for a particular problem, as done for Minimum Dominating Set (MDS) in Figure 1. In this way, note that for MDS, and in fact all problems adressed in Section 3, boolean-width compares well to the other parameters. Finally, note that while defining a parameter with an artificially low value, say by taking logarithms, would favor the comparison with other parameters in the upper part of Figure 1, one would pay for it in the runtime of the algorithms, say by going from single exponential $O^*(2^{poly(k)})$ to double exponential $O^*(2^{2^{poly(k)}})$.

The paper is organized as follows. In Section 2 we define in a common framework branch-width, rank-width, and boolean-width. In Section 3 we give algorithms for a large class of NP-hard vertex subset and vertex partitioning problems, namely (σ, ρ)-problems and D_q-problems [22]. These are related to domination, independence and homomorphism, including Max or Min Perfect Code, Max or Min Independent Dominating Set, Min k-Dominating Set, Max Induced k-Regular Subgraph, Max Induced k-Bounded Degree Subgraph, H-Coloring, H-Homomorphism, H-Covering, H-Partial Covering. Algorithms parameterized by either the tree-width of the input graph, or by its clique-width, have already been given for this class of problems [22,10], recently improved by van Rooij et al. [21]. The runtime achieved by these algorithms are $O^*(2^{O(tw)})$ and $O^*(2^{2^{poly(cw)}})$. Having small boolean-width is witnessed by a decomposition of the graph into cuts with few different unions of neighborhoods across the cut. This makes the decomposition natural for fast dynamic programming to solve problems, like Max Independent Set, where vertex sets having the same neighborhoods can be treated as equivalent [3]. Surprisingly, in this paper we extend such an observation to the much larger class of vertex subset and vertex partitioning problems. Several new techniques are introduced in order to achieve this and the runtime of these algorithms is $O^*(2^{O(boolw^2)})$, which then can also be interpreted as $O^*(2^{O(cw^2)})$ and $O^*(2^{O(rw^4)})$ by using the relationships in Figure 1, improving the $O^*(2^{2^{poly(cw)}})$ runtime in [10].

In Section 4 we relate boolean-width to branch-width. We prove for every graph G with $\mathbf{bw}(G) \neq 0$ that $\mathbf{boolw}(G) \leq \mathbf{bw}(G)$. For the proof we develop a general method of constructive manipulations of the decompositions, which also allows a simplified proof of a theorem by Oum [19] showing that $\mathbf{rw}(G) \leq \mathbf{bw}(G)$ (unless $E(G) \neq \emptyset$ and no two edges of G are adjacent).

While Oum's proof uses deep results from matroid theory, our argument avoids matroid theory and is based on constructive manipulations of the decompositions, giving a good understanding of the connections between the graph parameters. Kanté [13] gave a constructive proof showing that the rank-width of a graph is at most 4 times its tree-width plus 2. We improve this result.

In Section 5 we show that a random graph on n vertices almost surely has boolean-width $\Theta(\log^2 n)$, and it is easy to find the corresponding decomposition tree. This contrasts sharply with a series of negative results establishing that almost surely a random graph on n vertices has tree-width and branch-width [15], clique-width [12] and rank-width [17] all in $\Theta(n)$. The importance of this result is possibly not in the random graphs themselves, but in the indication that boolean-width is sometimes – actually, quite often – much smaller than all the other parameters, and therefore potentially very useful. Our result also implies the following: any problem solvable by dynamic programming in time $O^*(2^{poly(k)})$ *given* a decomposition of boolean-width k, can be solved in quasi-polynomial time on input a random graph (where we do not need a decomposition as part of input). Such problems include Minimum Dominating Set and Maximum Independent Set which can be solved in time $O(n(n+2^{3k}k))$ [3]. Moreover, combining our results from Sections 3 and 5 we get an algorithm that given a random graph on n vertices, solves (σ,ρ)-problems and D_q-problems in quasi-polynomial time $O^*(2^{O(\log^4 n)})$.

Throughout the paper, missing proofs are given in the appendix.

2 Framework

In this paper, graphs are loopless simple undirected graphs. Let G be a graph with vertex set $V(G)$ and edge set $E(G)$. For a vertex $v \in V(G)$ let $N(v)$ be the set of all neighbours of v in G. We extend this to subsets $X \subseteq V(G)$ by letting $N(X) := \bigcup_{v\in X} N(v)$. For a tree T we denote the set of leaves by $L(T)$. A tree is *subcubic* if every vertex has degree either 1 or 3.

Let A be a finite set. For a subset $X \subseteq A$ let $\overline{X} := A \setminus X$. Let $f\colon 2^A \to \mathbb{R}$ be a *symmetric* set function, i.e. f satisfies $f(X) = f(\overline{X})$ for all $X \subseteq A$. A *decomposition tree* of f (on A) is a pair (T,δ), where T is a subcubic tree and $\delta\colon L(T) \to A$ is a bijection. Each edge $e \in E(T)$ yields a partition P_e of A, induced by the leaf labels of the two trees we get by removing e from T: if T_1 and T_2 denote the two components of $T-e$, then $P_e := \big(\delta(L(T_1)\cap L(T)), \delta(L(T_2)\cap L(T))\big)$. We extend the domain of f to edges e of T by letting $f(e) := f(X)$ for $P_e = (X,\overline{X})$. This is well-defined because f is symmetric. The f-*width* of a decomposition tree (T,δ) is $f\text{-w}(T,\delta) := \max\{f(e) \mid e \in E(T)\}$. The *width* of f is $\mathbf{width}(f) := \min\{f\text{-w}(T,\delta) \mid (T,\delta) \text{ decomposition tree of } f\}$. If $|A| \le 1$, then f has no decomposition tree and we let $\mathbf{width}(f) := f(A)$.

We now define branch-width of a graph. For a graph G and a subset $X \subseteq E(G)$ let

$$\partial(X) := \{v \in V(G) \mid v \text{ is incident to both an edge from } X \text{ and from } E(G)\setminus X\}$$

denote the *border* of X. We define $\mathbf{cut\text{-}bw}_G := \mathbf{cut\text{-}bw}\colon 2^{E(G)} \to \mathbb{N}$ as $\mathbf{cut\text{-}bw}(X) := |\partial(X)|$. Clearly, $\mathbf{cut\text{-}bw}$ is symmetric. The *branch-width* of G is defined as $\mathbf{bw}(G) := \mathbf{width}(\mathbf{cut\text{-}bw})$.

For subsets $X, Y \subseteq V(G)$ let $M_{(X,Y)}$ denote the $X\times Y$-submatrix of the adjacency matrix of G. Let Δ denote the symmetric difference of sets: $A\Delta B = (A\setminus B)\cup(B\setminus A)$. We define $\mathbf{cut\text{-}rk}_G := \mathbf{cut\text{-}rk}\colon 2^{V(G)} \to \mathbb{N}$ as

$$\mathbf{cut\text{-}rk}(X) := \log_2 \left| \{B \subseteq \overline{X} \mid \exists A \subseteq X \text{ with } B = \mathop{\Delta}_{v\in A} N(v)\cap \overline{X}\} \right| = \mathrm{rk}\big(M_{(X,\overline{X})}\big),$$

where $\mathrm{rk}\big(M_{(X,\overline{X})}\big)$ denotes the GF(2)-rank of $M_{(X,\overline{X})}$. Then the *rank-width* of G is $\mathbf{rw}(G) := \mathbf{width}(\mathbf{cut\text{-}rk})$.

For boolean-width we define $\textbf{cut-bool}_G := \textbf{cut-bool}\colon 2^{V(G)} \to \mathbb{R}$ as

$$\textbf{cut-bool}(X) := \log_2 \left|\{B \subseteq \overline{X} \mid \exists A \subseteq X \text{ with } B = N(A) \cap \overline{X}\}\right| .$$

Surprisingly, the function **cut-bool** is symmetric [14, Theorem 1.2.3]. The *boolean-width* of a graph G is $\textbf{boolw}(G) := \textbf{width}(\textbf{cut-bool})$. Let us give an alternative view on boolean-width. Let $R(M_{(X,Y)})$ denote the set of all vectors spanned by the rows of $M_{(X,Y)}$ by taking Boolean sums, i.e. $1+1=1$. It is easy to see that $\textbf{cut-bool}(X) = \log_2 \left|R(M_{(X,\overline{X})})\right|$.

3 Vertex subset and vertex partitioning problems

Given a graph G together with a decomposition tree of **cut-bool** of width $boolw$, in this section we give algorithms with runtime $O^*(2^{O(boolw^2)})$ solving a large class of problems, the so-called (σ,ρ) vertex subset and D_q vertex partitioning problems as defined in [22]. Before coming to the formal definition, let us give a general discussion about these problems and the graph parameters we are addressing.

Firstly, these are problems expressible in MSO logic and it follows from Courcelle's Theorem [5] that they belong to FPT when parameterized by either the tree-width, branch-width, clique-width, rank-width or boolean-width of the graph, when an appropriate decomposition is also given in the input. However, applying Courcelle's Theorem can only be seen as a theoretical result for deciding the complexity class, since the runtime contains a highly exponential factor (tower of powers). This results in several papers aiming at the design of algorithms having the lowest dependency on the parameters [22,10,21], originally done for tree-width and clique-width, but having an implication to the other parameters as well, *e.g.*, by using the relationships in Figure 1 in a straightforward manner. Also, these are well-behaved problems when parameterized by tree-width tw, namely there are $O^*(2^{O(tw)})$ algorithms solving the problem if a tree decomposition of width tw is given [22], with the fastest known for (σ,ρ)-problems being $O^*((d(\sigma)+d(\rho)+2)^{tw})$ [21], where $d(\sigma)$ and $d(\rho)$ are problem specific constants (see below). This is not the same situation for clique-width, where until now the best runtime contains a $O^*(2^{2^{poly(cw)}})$ double exponential factor [10]. Finally, for rank-width and boolean-width, no specific algorithm is known so far other than a direct translation of the algorithm in [10] using Figure 1, which then results in a runtime containing a triple exponential factor in the parameter. The algorithms we give in this section are the first having a runtime with single exponential dependency in the boolean-width, and hence also single exponential in rank-width and clique-width. We now define formally this class of problems.

Definition 1. Let σ and ρ be finite or co-finite subsets of natural numbers. A subset X of vertices of a graph G is a *sigma-rho set*, or simply *(σ,ρ)-set*, of G if

$$\forall v \in V(G) : |N(v) \cap X| \in \begin{cases} \sigma \text{ if } v \in X, \\ \rho \text{ if } v \in V(G) \setminus X. \end{cases}$$

The *vertex subset problems* consist of finding the size of a minimum or maximum (σ,ρ)-set in G. Several NP-hard problems are expressible in this framework, *e.g.*, Max Independent Set($\{0\}$, $\mathbb{N}$), Min Dominating Set($\mathbb{N}$, $\mathbb{N}\setminus\{0\}$), Max Strong Stable Set($\{0\}$, $\{0,1\}$), Max or Min Perfect Code($\{0\}$, $\{1\}$), Also if we let $M_k = \{0,1,2,\ldots k\}$ then Min k-Dominating Set($\mathbb{N}$, $\mathbb{N}\setminus M_k$), Max Induced k-Regular Subgraph($\{k\}$, $\mathbb{N}$) (see [22] for further details and a more complete list). This framework is extendible to problems asking for a partition of $V(G)$ into q classes, with each class satisfying a certain (σ,ρ)-property:

Definition 2. A *degree constraint* matrix D_q is a q by q matrix with entries being finite or co-finite subsets of natural numbers. A *D_q-partition* in a graph G is a partition $\{V_1,V_2,...,V_q\}$ of $V(G)$ such that for $1 \le i,j \le q$ we have $\forall v \in V_i : |N(v) \cap V_j| \in D_q[i,j]$.

The *vertex partitioning problems* for which we give algorithms in this paper consist of deciding if G has a D_q partition, the so-called $\exists D_q$ problem. NP-hard problems fitting into this framework include *e.g.* for any fixed graph H the problems known as H-Coloring or H-Homomorphism (with q-Coloring being K_q-Coloring), H-Covering, H-Partial Covering, and in general the question of deciding if a graph has a partition into q (σ,ρ)-sets [22].

We focus on algorithms for the vertex subset problems. Let a graph G and a decomposition tree (T,δ) of **cut-bool** be given as input. Our algorithm will follow a bottom-up dynamic programming approach: subdivide an arbitrary edge of T to obtain a root r, and denote by T_r the resulting rooted tree. With each node w of T_r we associate a table data structure Tab_w, that will store optimal solutions to subproblems related to V_w, the set of vertices of G mapped to the leaves of the subtree of T_r rooted at w. Each index of the table will be associated with a certain class of equivalent subproblems that we need to define depending on the problem on which we are focusing.

Let $d(\mathbb{N})=0$. For every finite or co-finite set $\mu\subseteq\mathbb{N}$, let $d(\mu)=1+min\{max_{x\in\mathbb{N}}x : x\in\mu, max_{x\in\mathbb{N}}x : x\notin\mu\}$. We denote by $d(\sigma,\rho)$, or simply by d when it appears clearly in the context that σ and ρ are involved, the value $d=d(\sigma,\rho)=max\{d(\sigma),d(\rho)\}$. Note that when checking if a subset A of vertices is a (σ,ρ)-set, as in Definition 1, it suffices to count the number of neighbors up to d that a vertex has in A. This is the key to getting fast algorithms and motivates the following equivalence relation.

Definition 3 (d-neighbor equivalence). Let G be a graph and $A\subseteq V(G)$. Two vertex subsets $X\subseteq A$ and $X'\subseteq A$ are *d-neighbor equivalent* w.r.t. A, denoted by $X\equiv^d_A X'$, if $\forall v\in\overline{A} : \big(|N(v)\cap X| = |N(v)\cap X'|\big)\vee\big(|N(v)\cap X|\geq d \wedge |N(v)\cap X'|\geq d\big)$.

We now depict the entries of the table data structure Tab_w. Roughly, we aim at solving the vertex subset problems using one d-neighbor equivalence class per entry in Tab_w. For this, we first define a canonical representative for every d-neighbor equivalence class.

Lemma 1. *Let G be a graph and $A\subseteq V(G)$. Then, for every $X\subseteq A$, there is $R\subseteq A$ such that $R\equiv^d_A X$ and $|R|\leq d\cdot\textbf{cut-bool}(A)$. Moreover, the number of equivalence classes of $\equiv^d_A$ is at most $2^{d\cdot\textbf{cut-bool}(A)^2}$.*

Proof. We start with the first part namely bounding the size of the minimal members. Let $R=X$, go through all vertices $x\in X$ and delete x from R if $R\setminus\{x\}\equiv^d_A X$. Notice that $R\subseteq X$ and $R\equiv^d_A X$ and hence fulfill two of the requirements.

We then know that $\forall x\in R\,\exists y\in N(x)\backslash A : |N(y)\cap X|\leq d$, this since otherwise $R\backslash\{x\}\equiv^d_A X$. We build a set S as follows:
Let $S=\emptyset$, $R'=R$. While you can find a pair $x\in R', y\in\overline{A}$ such that $|N(y)\cap R'|\leq d$ and $x\in N(y)$. Remove $N(y)$ from R', add x to S.

Note that all the x's and y's chosen are different since you remove the neighborhood of y from R' and that you can always find such a set unless $R'=\emptyset$. Therefore we know $|S|\geq|R|/d$ since we remove at most d nodes in each step. Any of the $2^{|S|}$ combinations of elements from S will form a unique neighborhood. Therefore, we get from definition $\textbf{cut-bool}(A)=\textbf{cut-bool}(\overline{A})\geq|S|$. Since $|S|\geq|R|/d$, R fulfills the last requirement.

To bound the number of equivalence classes we know from the previous arguments that we only need to find the equivalence classes among the subsets of A of size at most $d\cdot\textbf{cut-bool}(A)$. Let H be obtained from the bipartite subgraph of G with color classes $A,\overline{A}$ after doing twin contraction of all twins and adding an isolated vertex to each color class unless there already is one. We know that every node of $V(H)\cap A$ has a unique neighborhood, hence $|V(H)\cap A|\leq 2^{\textbf{cut-bool}(A)}$. Any subset of A is a multiset of $|V(H)\cap A|$, and a trivial bound on number of multisets of $|V(H)\cap A|$ with size $d\cdot\textbf{cut-bool}(A)$ gives us: number of equivalence classes less than or equal to $2^{d\cdot\textbf{cut-bool}(A)^2}$. □

We now define the canonical representative $can^d_{V_w}(X)$ of every subset $X \subseteq V_w$, and the canonical representative $can^d_{\overline{V_w}}(Y)$ of every subset $Y \subseteq \overline{V_w}$. For simplicity we define this for V_w only, but the definition can be used for $\overline{V_w}$ as well, since everything we say about $X \subseteq V_w$, $can^d_{V_w}(X)$ and $\equiv^d_{V_w}$ will hold also for $can^d_{\overline{V_w}}(Y)$, $Y \subseteq \overline{V_w}$ and $\equiv^d_{\overline{V_w}}$. Canonical representatives are to be used for indexing the table Tab_w at node w of the tree T_r. Three properties will be required. Firstly, if $X \equiv^d_{V_w} X'$, then we must have $can^d_{V_w}(X) = can^d_{V_w}(X')$. Secondly, given (X, Y), we should have a fast routine that outputs a pointer to the entry $Tab_w[can^d_{V_w}(X)][can^d_{\overline{V_w}}(Y)]$. Thirdly, we should have a list whose elements can be used as entries of the table, *i.e.* a list containing all canonical representatives w.r.t. $\equiv^d_{V_w}$. The following definition trivially fulfills the first requirement.

Definition 4. We assume that a total ordering of the vertices of $V(G)$ is given. For every $X \subseteq V_w$, the canonical representative $can^d_{V_w}(X)$ is defined as the lexicographically smallest set $R \subseteq V_w$ such that: $|R|$ is minimized and $R \equiv^d_{V_w} X$.

Definition 5. Let G be a graph, $A \subseteq V(G)$, and $\mu \subseteq \mathbb{N}$. For $X \subseteq V(G)$, we say that X *μ-dominates* A if $\forall v \in A : |N(v) \cap X| \in \mu$. For $X \subseteq A$, $Y \subseteq \overline{A}$, we say that (X, Y) *σ, ρ-dominates* A if $(X \cup Y)$ σ-dominates X and $(X \cup Y)$ ρ-dominates $A \setminus X$.

Definition 6. Let *opt* stand for either function *max* or function *min*, depending on whether we are looking for a maximum or minimum (σ, ρ)-set, respectively. For every node w of T_r, for $X \subseteq V_w$ and $Y \subseteq \overline{V_w}$, let $R_X = can^d_{V_w}(X)$ and $R_Y = can^d_{\overline{V_w}}(Y)$. We define the contents of $Tab_w[R_X][R_Y]$ as:

$$Tab_w[R_X][R_Y] \overset{\text{def}}{=} \begin{cases} opt_{S \subseteq V_w}\{|S| : S \equiv^d_{V_w} X \text{ and } (S, Y)\ \sigma, \rho\text{-dominates } V_w\}, \\ -\infty \text{ if no such set } S \text{ exists and } opt = max, \\ +\infty \text{ if no such set } S \text{ exists and } opt = min. \end{cases}$$

Lemma 2. *For every node w of T_r, a list containing all canonical representatives w.r.t. $\equiv^d_{V_w}$ can be output in time $O(m + d \cdot \mathbf{cut\text{-}bool}(V_w) \cdot 2^{2d \cdot \mathbf{cut\text{-}bool}(V_w)^2 + \mathbf{cut\text{-}bool}(V_w)})$. For every subset $X \subseteq V_w$, a pointer to $can^d_{V_w}(X)$ in that list is found in time $O(|X| \cdot 2^{\mathbf{cut\text{-}bool}(V_w)})$.*

Proof. Let H be obtained from the bipartite subgraph of G with color classes $A, \overline{A}$ after doing twin contraction of all twins and and adding an isolated vertex to each color class unless there already is one. This can be computed in $O(|E(G)|)$ time [2] from the so-called external module partition defined therein. We know from the proof of Lemma 1 that $|V(H) \cap V_w| \leq 2^{\mathbf{cut\text{-}bool}(V_w)}$. Combining Lemma 1 with the fact that a canonical representative is chosen among the sets of minimal size belonging to that equivalence class, we know that the size of any representative is no more than $d \cdot \mathbf{cut\text{-}bool}(V_w)$. Knowing this we see that it is possible to loop through all sets of size less than or equal to the largest representative.

A list C will be used in order to store the canonical representatives. We will later want to access $Tab_w[can^d_{V_w}(X)]$, so in the list C we will store pairs, one being the set of vertices that actually appear in $can^d_{V_w}(X)$, and the other being a certain index. And we make one big table M indexed by all subsets of size at most $d \cdot \mathbf{cut\text{-}bool}(V_w)$. For each entry in M we will store a pointer its representative in C. So given a set $X : |X| \leq d \cdot \mathbf{cut\text{-}bool}(V_w)$, $M[X]$ will return both the canonical representative and its index. To check if two sets X and Y are equivalent we find their neighborhoods and see if they are equal up to d neighbors. Hence it can be done in $O((|X| + |Y|)2^{\mathbf{cut\text{-}bool}(V_w)})$ time.

We can compute list C as follows. Set $index = 0$. We first loop over all sets X of size at most $d \cdot \mathbf{cut\text{-}bool}(V_w)$. To make sure we find the representatives as described in the definition of canonical representatives, we loop over smaller sets first, and in lexicographically order among those with same size. Then we loop over all sets $R \in C$ and check if $X \equiv^d_A R$, if so $M[X]$ points to R. If no $R \in C$ is equivalent to X, add $X, index$ to C, increase index and $M[X]$ points to X. In total this becomes $O(m + 2^{d \cdot \mathbf{cut\text{-}bool}(V_w)^2} 2^{d \cdot \mathbf{cut\text{-}bool}(V_w)^2} 2d \cdot \mathbf{cut\text{-}bool}(V_w)) 2^{\mathbf{cut\text{-}bool}(V_w)}) = O(m + 2^{2d \cdot \mathbf{cut\text{-}bool}(V_w)^2 + \mathbf{cut\text{-}bool}(V_w)} d \cdot \mathbf{cut\text{-}bool}(V_w))$ time.

Finally, we compute $can^d_{V_w}(X)$ as follows. Let $X' = X$ first go through all vertices $x \in X$ and delete x from X' if $X' \setminus \{x\} \equiv^d_{V_w} X$. As we showed in proof of Lemma 1 we can now assume $|X'| \leq d \cdot \mathbf{cut\text{-}bool}(V_w)$. Now we look up in M and find a pointer to both the canonical representative and the index. This means $can^d_{V_w}(X)$ can be computed in $O(|X| 2^{\mathbf{cut\text{-}bool}(V_w)})$ time. □

Note that at the root r of T_r the value of $Tab_r[X][\emptyset]$ (for all $X \subseteq V(G)$) would be exactly equal to the size of a maximum, resp. minimum, (σ, ρ)-set of G (*cf.* $\equiv^d_{V_r}$ has only one equivalence class). For initialization, the value of every entry of Tab_w will be set to $+\infty$ or $-\infty$ depending on whether we are solving a minimization or maximization problem, respectively. For a leaf l of T_r, we perform a brute-force update: let $A = \{l\}$ and $B = \overline{A}$, for every canonical representative R w.r.t. $\equiv^d_B$, we set:

- If $|N(l) \cap R| \in \sigma$ then $Tab_l[A][R] = 1$.
- If $|N(l) \cap R| \in \rho$ then $Tab_l[\emptyset][R] = 0$.

For a node w of T_r with children a and b, the algorithm performs the following steps. For every canonical representative $R_{\overline{w}}$ w.r.t. $\equiv^d_{\overline{V_w}}$, for every canonical representative R_a w.r.t. $\equiv^d_{V_a}$, and for every canonical representative R_b w.r.t. $\equiv^d_{V_b}$, do:

- Compute $R_w = can^d_{V_w}(R_a \cup R_b)$, $R_{\overline{a}} = can^d_{\overline{V_a}}(R_b \cup R_{\overline{w}})$ and $R_{\overline{b}} = can^d_{\overline{V_b}}(R_a \cup R_{\overline{w}})$
- Update $Tab_w[R_w][R_{\overline{w}}] = opt(Tab_w[R_w][R_{\overline{w}}], Tab_a[R_a][R_{\overline{a}}] + Tab_b[R_b][R_{\overline{b}}])$.

Lemma 3. *The table at node w is updated correctly, namely for any canonical representatives R_w and $R_{\overline{w}}$ w.r.t. $\equiv^d_{V_w}$ and $\equiv^d_{\overline{V_w}}$, if $Tab_w[R_w][R_{\overline{w}}]$ is not $\pm\infty$ then*

$$Tab_w[R_w][R_{\overline{w}}] = opt_{S \subseteq V_w}\{|S| : S \equiv^d_{V_w} R_w \wedge (S, R_{\overline{w}})\ \sigma,\rho\text{-dominates } V_w\}.$$

If the value of the table is $\pm\infty$ then there is no such above set S.

Theorem 1. *For every n-vertex, m-edge graph G given along with a decomposition tree (T, δ) for* **cut-bool**, *any vertex subset problem on G can be solved in $O(n(m + d \cdot \mathbf{cut\text{-}bool\text{-}w}(T,\delta) 2^{3d \cdot \mathbf{cut\text{-}bool\text{-}w}(T,\delta)^2 + \mathbf{cut\text{-}bool\text{-}w}(T,\delta)}))$ time, where $d = d(\sigma, \rho)$.*

Proof. Correctness follows directly from what has been said in this section. For complexity analysis, for every node w of T_r, we basically call the first computation of Lemma 2 once, then loop through every triplet $R_{\overline{w}}$, R_a, R_b of equivalence classes, call the second computation of Lemma 2 three times, and perform the table update. □

The algorithms for vertex partitioning problems are similar but require some graph-theoretic observations and several technical details. For space reasons this has all been moved to the appendix.

Theorem 2. *For every n-vertex, m-edge graph G given along with a decomposition tree (T, δ) of* **cut-bool**, *any D_q-problem on G, with $d = \max_{i,j} d(D_q[i,j])$, can be solved in $O(n(m + qd \cdot \mathbf{cut\text{-}bool\text{-}w}(T,\delta) 2^{3qd \cdot \mathbf{cut\text{-}bool\text{-}w}(T,\delta)^2 + \mathbf{cut\text{-}bool\text{-}w}(T,\delta)}))$ time.*

4 Boolean-width is less than or equal to Branch-width

In this section we relate boolean-width to branch-width, and show the following

Theorem 3. *Any graph G satisfies* $\mathbf{boolw}(G) \leq \mathbf{bw}(G)$ *(unless $E(G) \neq \emptyset$ and no two edges of G are adjacent).*

In order to clarify how the decomposition trees relate to each other, we divide our result into two steps, addressing the intermediary notion of an incidence graph (see Lemmata 4 and 5). However, we will also show how to easily derive from our method a direct proof without incidence graphs. Our framework not only applies for boolean-width, but also captures other settings including rank-width. The *incidence graph* $I(G)$ of a graph G is the graph with vertex set $V(G)\dot{\cup}E(G)$, where x and y are adjacent in $I(G)$ if one of x, y is a vertex of G, the other is an edge of G and x and y are incident in G.

Lemma 4. *Any graph G satisfies* $\mathbf{boolw}(I(G)) \leq \mathbf{bw}(G)$ *and* $\mathbf{rw}(I(G)) \leq \mathbf{bw}(G)$*, unless $E(G) \neq \emptyset$ and no two edges of G are adjacent. In this case,* $\mathbf{bw}(G) = 0$ *and* $\mathbf{rw}(I(G)) = \mathbf{boolw}(I(G)) = 1$.

The proof is given in the appendix, but let us sketch the idea. Starting with a decomposition tree (T, δ) of $\mathbf{cut\text{-}bw}_G$ of width k, we modify the decomposition tree in two steps. In the first step, we replace every leaf ℓ of T by a subcubic tree with three leaves, and we label one of the three leaves with the edge $\delta(\ell)$ and we label the other two leaves with the two vertices incident with $\delta(\ell)$. In a second step, for each $v \in V(G)$ we choose one leaf with label v, we keep this leaf, and delete all other leaves that are labelled by v. In this way we obtain a decomposition tree of $\mathbf{cut\text{-}bool}_{I(G)}$ (and of $\mathbf{cut\text{-}rk}_{I(G)}$) of boolean-width (rank-width, resp.) at most k.

Lemma 5. *Any graph G satisfies* $\max\{\mathbf{boolw}(G), \mathbf{rw}(G)\} \leq \min\{\mathbf{boolw}(I(G)), \mathbf{rw}(I(G))\}$.

Theorem 3 now follows immediately from Lemmata 4 and 5. It is also easy to give a direct proof using the proof idea of Lemma 4. The only difference is in the first modification step. Instead of taking a subcubic tree with three leaves, we take the subcubic tree with two leaves (since we do not need to assign leaves to graph edges). Note that there is no bound in the converse direction: the class of all complete graphs has unbounded branch-width and the boolean-width is at most 1. Nevertheless, moving to incidence graphs we prove a weak converse.

Lemma 6. *Any graph G satisfies* $\mathbf{bw}(G) \leq 2 \cdot \min\{\mathbf{boolw}(I(G)), \mathbf{rw}(I(G))\}$.

Altogether, the same technique for proving Theorem 3 also shows that

Corollary 1. *Any graph G satisfies* $\mathbf{boolw}(G) \leq \mathbf{bw}(G) \leq 2 \cdot \mathbf{boolw}(I(G))$ *(unless $E(G) \neq \emptyset$ and no two edges of G are adjacent).*

Corollary 2. *Any graph G satisfies*

1. $\mathbf{boolw}(I(G)) \leq \mathbf{bw}(I(G)) \leq 2 \cdot \mathbf{boolw}(I(G))$ *and*
2. $\mathbf{boolw}(I(G)) \leq \mathbf{rw}(I(G)) + 1 \leq 2 \cdot \mathbf{boolw}(I(G)) + 1$.

Proof. Note that $\mathbf{bw}(G) = \mathbf{bw}(I(G))$, unless $E(G) \neq \emptyset$ and no two edges of G are adjacent. In this case, $\mathbf{bw}(G) = 0$ and $\mathbf{bw}(I(G)) = 1$. Using this, the first statement follows from Lemmata 4 and 6. The second statement follows from the first by using a theorem from [19] stating that $\mathbf{rw}(I(G)) \in \{\mathbf{bw}(G), \mathbf{bw}(G) - 1\}$. □

5 Random graphs

Let G_p be a random graph on n vertices where each edge is chosen randomly and independently with probability p (independent of n). There has been a series of negative results [15,12,17] establishing that almost surely G_p has rank-width, tree-width, branch-width and clique-width $\Theta(n)$. In contrast we show in this section the following.

Theorem 4. *Almost surely,* $\mathbf{boolw}(G_p) = \Theta\left(\frac{\ln^2 n}{p}\right)$.

We start with the upper bound and prove first the following lemma.

Lemma 7. *Let G_p be a graph as above, and let $k_p = \lfloor \frac{2\ln n}{p} \rfloor$. Then, almost surely, for* all *subsets of vertices $S \subset V(G)$ with $|S| = k_p$ it holds that $|N(S) \setminus S| \geq |\overline{S}| - k_p$.*

Proof. In what follows, we write simply G and k. Fix a particular S with $|S| = k$. For every $v \in \overline{S}$, let X_v be 1 if $v \notin N(S)$, and 0 otherwise. Clearly, $X_v = 1$ with probability $(1-p)^k$, and $\sum_{v\in\overline{S}} X_v = |\overline{S} \setminus N(S)|$. Observe that $E[\sum_{v\notin S} X_v] = (1-p)^k(n-k) < (1-p)^k n$. Call this expectation μ. By Chernoff Bound (see *e.g.* [18], p.68),

$$\Pr\left[\sum_{v\in\overline{S}} X_v \geq k\right] < \left(\frac{e\mu}{k}\right)^k < \left((1-p)^k n\right)^k = \left((1-p)^{2\ln n/p} n\right)^k < n^{-k},$$

the last inequality due to the fact that for $p \in (0,1)$, $(1-p)^{\frac{1}{p}} \leq e^{-1}$.

Applying the union bound, we conclude that the probability that there exists S of size k such that $|N(S) \setminus S| < |\overline{S}| - k$ is at most $\binom{n}{k} \cdot n^{-k} < (k!)^{-1} = o(1)$ and the statement follows. □

Corollary 3. *For $G = G_p$ and $k = k_p$ as before, for* all *cuts $\{A, \overline{A}\}$ in G it holds almost surely that* $\mathbf{cut\text{-}bool}(A) = O\left(\frac{\ln^2 n}{p}\right)$.

Proof. The number of distinct sets $N(S) \cap \overline{A}$ contributed by the sets $S \subseteq A$ with $|S| \leq k$ is at most $\sum_{i=0}^{k} \binom{n}{i}$. By the previous lemma, for all sets $S \subseteq A$ with $|S| \geq k$, it holds almost surely that $|N(S) \cap \overline{A}| \geq |\overline{A}| - k$. Therefore, almost surely, the sets $S \subseteq A$ with $|S| \geq k$, also contribute at most $\sum_{i=0}^{k} \binom{n}{i}$ distinct sets $N(S) \cap \overline{A}$. Thus, almost surely there are at most $2\sum_{i=0}^{k} \binom{n}{i}$ distinct sets $N(S) \cap \overline{A}$ altogether. Taking the logarithm, we arrive at the desired conclusion. □

The upper bound of Theorem 4 now follows easily: for *any* decomposition tree of **cut-bool**, all the cuts it defines will almost surely have boolean-width at most $O\left(\frac{\ln^2 n}{p}\right)$. Next, we move to the lower bound of Theorem 4. For simplicity of exposition, we restrict the discussion to the case $p = 0.5$. The lower bound for that case follows from:

Lemma 8. *Let $\{A, \overline{A}\}$ be a cut where $|A| = |\overline{A}| = m$, and the edges are chosen independently at random with probability 0.5. Then,* $\Pr[\ \mathbf{cut\text{-}bool}(A) = \Omega(\log^2 m)\ ] \geq 1 - 2^{-\Omega(m^{1.3})}$. *More concretely, the probability that among the neighborhoods of the subsets of A of size $k = 0.25 \cdot \log_2 m$, there are less than $2^{c\log^2 m}$ different ones (for a suitable constant c), is at most $2^{-\Omega(n^{1.3})}$.*

To prove this lemma we need some notation and preliminary results first. Let the (random) set $S_i \subseteq \overline{A}$, $i = 1, 2, \ldots, m$ be the neighborhood of the vertex $i \in A$, and let $S_I = \cup_{i\in I} S_i$. We shall only be interested in the I's of size k as above. Call such I *bad* if $m - |S_I| < m^{0.5}$. Call a set I of size k *thick* if there are at least $m^{0.9}$ indices $i \in \{1, 2, \ldots, m\} - I$ such that $S_i \subseteq S_I$.

Claim 1. $\Pr\left[\frac{\textit{the number of bad } I\textit{'s}}{\binom{m}{k}} \geq 0.5\right] < e^{-\Omega(m^{1.74})}$.

Claim 2. *For a fixed set I of size k, the probability that I is thick conditioned on its being good (that is, not bad), is at most $e^{-\Omega(m^{1.3})}$.*

Corollary 4. $\Pr\left[\,\textit{the number of thick } I\textit{'s} \;>\; 0.5\cdot\binom{m}{k}\right] \;<\; e^{-\Omega(m^{1.3})}$.

Proof of Lemma 8: Define a directed random graph $\mathcal{G}$ whose vertices are the sets I of size k, and (I,J) is an edge if $S_J \subseteq S_I$. Observe that the size of maximum independence set in $\mathcal{G}$ is precisely the number of different neighborhoods. Since S_I contains only S_J's where J is a union of i's such that $S_i \subseteq S_I$, we conclude that if a set I is thin, then its degree is no more than

$$d \;=\; \binom{k+m^{0.9}}{k} \;=\; \binom{k+m^{0.9}}{0.25\cdot\log_2 m} \;\approx\; m^{-0.1\cdot 0.25\cdot\log_2 m}\cdot\binom{m}{k} \;=\; 2^{-\Omega(\log^2 m)}\cdot\binom{m}{k}.$$

Consider the subgraph of $\mathcal{G}$ induced by the thin I's. Let $\mathcal{N}$ be its number of vertices.

By a standard greedy argument, the size of the maximum independence set in a graph of a maximal outdegree d is at least $\frac{\mathcal{N}}{2d+1}$. Indeed, mark a vertex of *indegree* $\le d$ in G (there must be such), then remove it and all its neighbours of both types from G, and continue in the same manner on the remaining graph. Since by Corollary 4, $\mathcal{N} \;\ge\; 0.5\cdot\binom{m}{k}$ with probability $1-e^{-\Omega(m^{1.3})}$, the statement follows. □

Proof of Theorem 4: The upper bound has already been proved. For the lower bound we restrict for simplicity of exposition to the case $p = 0.5$. Consider a $(\frac{1}{3},\frac{2}{3})$-balanced cut in G. Due to the monotonicity of the **cut-bool** with respect to taking induced subcuts, the Lemma 8 applies in this case with $m = n/3$. Therefore, the probability that **cut-bool** of this cut is $\Omega(\log^2 n)$ is $1-e^{-\Omega(n^{1.3})}$. Since there at most 2^n cuts in G, we conclude that with probability $1-e^{-\Omega(n^{1.3})}$ *all* balanced cuts have such **cut-bool**. Since any partition tree of G must contain a $(\frac{1}{3},\frac{2}{3})$-balanced cut, the statement follows. □

6 Further Research

In this paper we have seen that for random graphs boolean-width is the right parameter to consider: any decomposition tree will have boolean-width polylogarithmic in n. This also hints at the existence of large classes of graphs where boolean-width is polylogarithmic in the value of the other parameters, and raises the question of identifying these. One such class of graphs is defined by the so-called Hsu-grids [3], where boolean-width is $\Theta(\log n)$ and rank-width, branch-width, tree-width and clique-width are $\Theta(\sqrt{n})$. In contrast, we know that the boolean-width of regular graphs is $\Theta(n)$ [20], thus such an above mentioned class should exclude these graphs.

We believe that boolean-width should be useful for *practical* applications. We have initiated research to find fast and good heuristics computing decompositions of low boolean-width, similar to what is done for treewidth in the TreewidthLIB project [1].

A big open question is to decide if the boolean-width of a graph can be computed in FPT time. The relationship between rank-width and boolean-width is still not completely clear. Could it be that the boolean-width of any graph is linear in its rank-width? Currently the best bound is $boolw(G) \le \frac{1}{4}rw(G)^2 + \frac{5}{4}rw(G) + \log rw(G)$ [3].

The runtime of the algorithms given here for (σ,ρ)-problems and D_q-problems have the square of the boolean-width as a factor in the exponent. For problems where $d = 1$ we can in fact improve this to a factor linear in the exponent [3], but that requires a special focus on these cases. In fact, we believe that also for the other problems (with any constant value of d) we could get runtimes with an exponential factor linear in boolean-width. We must then improve the bound in Lemma 1, by showing that the number of d-neighborhood equivalence classes is no more than the number of 1-neighborhood equivalence classes raised to some function of d. This question can be formulated as a purely algebraic one as follows: First generalize the concept of Boolean sums $(1+1=1)$ to d-Boolean sums $(i+j=\min(i+j,d))$. For a Boolean matrix A let $R_d(A)$ be the set of vectors over $\{0,1,...,d\}$ that arise from all possible d-Boolean sums of rows of A. Is there a function f such that $|R_d(A)| \le |R_1(A)|^{f(d)\log\log|R_1(A)|}$?

References

1. H. Bodlaender and A. Koster. Treewidth Computations I Upper Bounds. Technical Report UU-CS-2008-032, Department of Information and Computing Sciences, Utrecht University, 2008.
2. B.-M. Bui-Xuan, J. A. Telle, and M. Vatshelle. H-join decomposable graphs and algorithms with runtime single exponential in rankwidth. *Discrete Applied Mathematics. article in press.* `http://dx.doi.org/10.1016/j.dam.2009.09.009`.
3. B.-M. Bui-Xuan, J. A. Telle, and M. Vatshelle. Boolean-width of graphs. In *4th International Workshop on Parameterized and Exact Computation (IWPEC'09)*, volume 5917 of *LNCS*, pages 61–74, 2009.
4. D. Corneil and U. Rotics. On the relationship between clique-width and treewidth. *SIAM Journal on Computing*, 34(4):825–847, 2005.
5. B. Courcelle. Graph rewriting: An algebraic and logic approach. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, pages 193–242. 1990.
6. F. Dorn. Dynamic programming and fast matrix multiplication. In *14th Annual European Symposium (ESA'05)*, volume 4168 of *LNCS*, pages 280–291, 2006.
7. R. Downey and M. Fellows. *Parameterized Complexity.* Springer Verlag, 1999.
8. J. Flum and M. Grohe. *Parameterized Complexity Theory.* Springer Verlag, 2006.
9. R. Ganian and P. Hliněný. On Parse Trees and Myhill-Nerode-type Tools for handling Graphs of Bounded Rank-width. *Discrete Applied Mathematics. article in press.* `http://dx.doi.org/10.1016/j.dam.2009.10.018`.
10. M. Gerber and D. Kobler. Algorithms for vertex-partitioning problems on graphs with fixed clique-width. *Theoretical Computer Science*, 299(1-3):719–734, 2003.
11. P. Hliněný, S. Oum, D. Seese, and G. Gottlob. Width parameters beyond tree-width and their applications. *The Computer Journal*, 51(3):326–362, 2008.
12. Ö. Johansson. Clique-decomposition, NLC-decomposition and modular decomposition – Relatiohships and results for random graphs. *Congressus Numerantium*, 132:39–60, 1998.
13. M. Kanté. Vertex-minor reductions can simulate edge contractions. *Discrete Applied Mathematics*, 155(17):2328–2340, 2007.
14. K. H. Kim. *Boolean matrix theory and its applications.* Marcel Dekker, 1982.
15. T. Kloks and H. Bodlaender. Only few graphs have bounded treewidth. Technical Report UU-CS-92-35, Department of Information and Computing Sciences, Utrecht University, 1992.
16. D. Kobler and U. Rotics. Edge dominating set and colorings on graphs with fixed clique-width. *Discrete Applied Mathematics*, 126(2-3):197–221, 2003. Abstract at *SODA'01.*
17. C. Lee, J. Lee, and S. Oum. Rank-width of Random Graphs, *submitted manuscript.*
18. R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.
19. S. Oum. Rank-width is less than or equal to branch-width. *Journal of Graph Theory*, 57(3):239–244, 2008.
20. Y. Rabinovich and J. A. Telle. On the boolean-width of a graph: structure and applications `arxiv.org/pdf/0908.2765`.
21. J. Rooij, H. Bodlaender, and P. Rossmanith. Dynamic programming on tree decompositions using generalised fast subset convolution. In *17th Annual European Symposium on Algorithms (ESA'09)*, 2009. *to appear.*
22. J. A. Telle and A. Proskurowski. Algorithms for vertex partitioning problems on partial k-trees. *SIAM Journal on Discrete Mathematics*, 10(4):529–550, 1997.

Appendix

Proof of Lemma 3

Let a, b be the children of w in T_r, assume Tab_a and Tab_b are correct. We first show that if $Tab_w[R_w][R_{\overline{w}}] = s$ and hence not $\pm\infty$, then there exists a set S_w satisfying all the conditions. For a value of Tab_w to be set to s, it means that an update happened in the algorithm, hence there exist R_a and R_b such that: $R_{\overline{a}} = can^d_{\overline{V_a}}(R_b \cup R_{\overline{w}})$,$R_{\overline{b}} = can^d_{\overline{V_b}}(R_a \cup R_{\overline{w}})$ and $Tab_a[R_a][R_{\overline{a}}] + Tab_b[R_b][R_{\overline{b}}] = s$. Then we know that there exist S_a and S_b such that $(S_a, R_{\overline{a}})$ σ, ρ-dominates V_a and $(S_b, R_{\overline{b}})$ σ, ρ-dominates V_b and that $|S_a \cup S_b| = s$. Let $S_w = S_a \cup S_b$, then S_w fulfills the two conditions $|S| = s$ and $R_w \equiv^d_{V_w} S_w$, now we need to show that $(S_w, R_{\overline{w}})$ σ, ρ-dominates V_w. Since $(S_b \cup R_{\overline{w}}) \equiv^d_{\overline{V_a}} R_{\overline{a}}$ and $(S_a, R_{\overline{a}})$ σ, ρ-dominates V_a it follows from Definition 3 and 5 that $(S_a, S_b \cup R_{\overline{w}})$ σ, ρ-dominates V_a, this is left as an exercise for the reader. Similarly we conclude that $(S_b, S_a \cup R_{\overline{w}})$ σ, ρ-dominates V_b Let $S = S_w \cup R_{\overline{w}} = S_a \cup S_b \cup R_{\overline{w}}$ then we get S σ-dominates V_a and S σ-dominates V_b, hence it follows from Definition 5 that S σ-dominates V_w. Similarly we get S ρ-dominates V_a and S ρ-dominates V_b, hence S ρ-dominates V_w. Combining the two last facts it follows from definition 5 that $(S_w, R_{\overline{w}})\sigma, \rho$-dominates V_w.

Next we will $\forall R_w, R_{\overline{w}}$ show that if there exist an optimal set $S_w \equiv^d_{V_w} R_w$ such that $(S_w, R_{\overline{w}})$ σ, ρ-dominates V_w, then $Tab_w[can^d_{V_w}(S_w)][R_{\overline{w}}] = |S_w|$. Let $S_a = S_w \cap V_a$ and $S_b = S_w \cap V_b$. Since the algorithm goes through all triples of representatives, it will at some point go through $(R_a, R_b, R_{\overline{w}})$, where $R_a = can^d_{V_a}(S_a)$ and $R_b = can^d_{V_b}(S_b)$. Since $(S_w, R_{\overline{w}})$ σ, ρ-dominates V_w, $(S_a \cup S_b \cup R_{\overline{w}})$ σ-dominates V_w and $(S_a \cup S_b \cup R_{\overline{w}})\rho$-dominates V_w. Then $(S_a, S_b \cup R_{\overline{w}})$ σ, ρ-dominates V_a and $(S_b, S_a \cup R_{\overline{w}})$ σ, ρ-dominates V_b. Since in the algorithm $R_{\overline{a}} = can^d_{\overline{V_a}}(S_b \cup R_{\overline{w}})$, $(S_a, R_{\overline{a}})$ σ, ρ-dominates V_a. Similarly we get that $(S_b, R_{\overline{b}})$ σ, ρ-dominates V_b. Since S_w is the optimal $S_a, S_{\overline{a}}$ and $S_b, S_{\overline{b}}$ must be optimal too, this means that $Tab_a[R_a][R_{\overline{a}}] + Tab_b[R_b][R_{\overline{b}}] = |S_a \cup S_b| = |S_w|$, hence $Tab_w[R_w][R_{\overline{w}}] = |S_w|$.

By induction all tables will be correct. □

Proof of Theorem 2

We use similar techniques as those for vertex subset problems.

Definition 7. Let G be a graph and let $A \subseteq V(G)$ be a vertex subset of G. Two q-tuples $(X_1, X_2, ..., X_q)$ and $(Y_1, Y_2, ..., Y_q)$ of subsets of A are equivalent, denoted by $(X_1, X_2, ..., X_q) \equiv^{q,d}_A (Y_1, Y_2, ..., Y_q)$, if

$$\forall i \forall v \in \overline{A} : (|N(v) \cap X_i| = |N(v) \cap Y_i|) \vee (|N(v) \cap X_i| > d \wedge |N(v) \cap Y_i| > d).$$

Lemma 9. $(X_1, X_2, ..., X_q) \equiv^{q,d}_A (Y_1, Y_2, ..., Y_q)$ *if and only if* $\forall i X_i \equiv^d_A Y_i$. *A consequence is that the number of equivalence classes of* $\equiv^{q,d}_A$ *is at most that of* $\equiv^d_A$ *to the power* q.

The lemma follows directly from Definitions 3 and 7. In the sequel we will define the values of Tab_w directly indexed by the equivalence classes. For this we need to first define canonical representatives. For a node w of T_r, and $\mathcal{X} = (X_1, X_2, ..., X_q) : X_i \subseteq V_w$, we define $can^{q,d}_{V_w}(\mathcal{X}) = (can^d_{V_w}(X_1), can^d_{V_w}(X_2), ..., can^d_{V_w}(X_q))$.

Definition 8. Let G be a graph and $A \subseteq V(G)$. Let $\mathcal{X} = (X_1, X_2, ..., X_q) \in A^q$ and $\mathcal{Y} = (Y_1, Y_2, ..., Y_q) \in \overline{A}^q$. We say that $(\mathcal{X}, \mathcal{Y})$ D_q*-dominates* A if for all i, j we have that $(X_j \cup Y_j)$ $D_q[i, j]$*-dominates* X_i (*cf.* Definition 5).

Definition 9. For every node w of T_r, for every $\mathcal{X} = (X_1, X_2, ..., X_q) \in A^q$ and every $\mathcal{Y} = (Y_1, Y_2, ..., Y_q) \in \overline{A}^q$, let $\mathcal{R}_\mathcal{X} = can^{q,d}_{V_w}(\mathcal{X})$ and $\mathcal{R}_\mathcal{Y} = can^{q,d}_{V_w}(\mathcal{Y})$. We define the contents of $Tab_w[\mathcal{R}_\mathcal{X}][\mathcal{R}_\mathcal{Y}]$ as

$$Tab_w[\mathcal{R}_\mathcal{X}][\mathcal{R}_\mathcal{Y}] \overset{\text{def}}{=} \begin{cases} TRUE & \text{if } \exists \text{ partition } \mathcal{S} = (S_1, S_2, ..., S_q) \text{ of } V_w \text{ such that:} \\ & \mathcal{S} \equiv^{q,d}_{V_w} \mathcal{R}_\mathcal{X} \text{ and } (\mathcal{S}, \mathcal{R}_\mathcal{Y})\ D_q\text{-dominates } V_w \\ FALSE & \text{otherwise.} \end{cases}$$

The solution to the $\exists D_q$-problem is given by checking if some entry in the table at the root has value $TRUE$. The computation of the list of all canonical representatives w.r.t. $\equiv^{q,d}_{V_w}$ is basically q times the one given in the previous section. The same situation holds for the computation of a canonical representative from the input of a q-uplet. Firstly, initialize all values in all tables to $FALSE$.

For a leaf l of T_r, we will also denote the vertex of G mapped to l by l. Let $A = \{l\}$. Firstly, there are q possible classes l could belong to in a q-partition of A (recall that empty sets are allowed). We call their canonical representatives respectively $\mathcal{R}_{\mathcal{X}_1}$, $\mathcal{R}_{\mathcal{X}_2}$, $\ldots$, $\mathcal{R}_{\mathcal{X}_q}$. Secondly, for vertices in $B = V(G) \setminus \{l\}$ note that they are either neighbors of l or not. Hence we have at most $d+1$ choices (namely $0, 1, ..., d-1, \geq d$) for each of the q partition classes. (A consequence is that Tab_l has at most $q(d+1)^q$ entries.) For every canonical representative $\mathcal{R}_\mathcal{Y} = (Y_1, Y_2, \ldots, Y_q)$ w.r.t. $\equiv^{q,d}_{B}$, we have that $(\mathcal{R}_{\mathcal{X}_i}, \mathcal{R}_\mathcal{Y})$ D_q-dominates $\{l\}$ if and only if $\forall j |N(l) \cap Y_j| \in D_q[i,j]$. Accordingly, we perform the following update for every i and for every $\mathcal{R}_\mathcal{Y}$:

$Tab_l[\mathcal{R}_{\mathcal{X}_i}][\mathcal{R}_\mathcal{Y}]$ is set to be $TRUE$ if and only if $\forall j\ |N(l) \cap Y_j| \in D_q[i,j]$.

In the following, $\bigcup_q$ denotes the componentwise union of two q-tuples. For a node w with children a and b, the algorithm performs the following steps. For every canonical representative $\mathcal{R}_{\overline{w}}$ w.r.t. $\equiv^{q,d}_{\overline{V_w}}$, for every canonical representative $\mathcal{R}_a$ w.r.t. $\equiv^{q,d}_{V_a}$, and for every canonical representative $\mathcal{R}_b$ w.r.t. $\equiv^{q,d}_{V_b}$, do:

- Compute $\mathcal{R}_w = can^{q,d}_{V_w}(\mathcal{R}_a \bigcup_q \mathcal{R}_b)$, $\mathcal{R}_{\overline{a}} = can^{q,d}_{\overline{V_a}}(\mathcal{R}_b \bigcup_q \mathcal{R}_{\overline{w}})$, $\mathcal{R}_{\overline{b}} = can^{q,d}_{\overline{V_b}}(\mathcal{R}_a \bigcup_q \mathcal{R}_{\overline{w}})$
- If $Tab_w[\mathcal{R}_w][\mathcal{R}_{\overline{w}}] = FALSE$ then $Tab_w[\mathcal{R}_w][\mathcal{R}_{\overline{w}}] = Tab_a[\mathcal{R}_a][\mathcal{R}_{\overline{a}}] \wedge Tab_b[\mathcal{R}_b][\mathcal{R}_{\overline{b}}]$

Before proving the algorithm, let us mention that it is straightforward to extend the algorithm to find the maximum or minimum cardinality of a vertex partition class over all D_q-partitions.

We now prove the algorithm. The complexity analysis is very similar to the one given in Theorem 1. The correctness proof follows the same style as the proof of Lemma 3, Some steps are not explained here because they were explained in Lemma 3.

For the correctness, let a, b be the children of w in T_r, assume Tab_a and Tab_b are correct. ($\Rightarrow$) For this direction of the proof we have that $Tab_w[\mathcal{R}_w][\mathcal{R}_{\overline{w}}] = TRUE$. Then there must exist some $\mathcal{R}_a, \mathcal{R}_b$ such that $Tab_a[\mathcal{R}_a][\mathcal{R}_{\overline{a}}] = TRUE$ and $Tab_b[\mathcal{R}_b][\mathcal{R}_{\overline{b}}] = TRUE$, where $\mathcal{R}_{\overline{a}} = can^d_{\overline{V_a}}(\mathcal{R}_b \bigcup_q \mathcal{R}_{\overline{w}})$ and $\mathcal{R}_{\overline{b}} = can^d_{\overline{V_b}}(\mathcal{R}_a \bigcup_q \mathcal{R}_{\overline{w}})$. Hence there exists $\mathcal{S}_a$ partition of V_a and $\mathcal{S}_b$ partition of V_b such that $(\mathcal{S}_a, \mathcal{R}_{\overline{a}})$ D_q-dominates V_a $(\mathcal{S}_b, \mathcal{R}_{\overline{b}})$ D_q-dominates V_b. This means that $\forall i,j : (S_{a_j} \cup R_{\overline{a}_j})$ $D_q[i,j]$-dominates S_{a_i} and $\forall i,j : (S_{b_j} \cup R_{\overline{b}_j})$ $D_q[i,j]$-dominates S_{b_i}. It then follows that: $\forall i,j : (S_{a_j} \cup S_{b_j} \cup R_{\overline{w}_j})$ $D_q[i,j]$-dominates S_{a_i} and $\forall i,j : (S_{a_j} \cup S_{b_j} \cup R_{\overline{w}_j})$ $D_q[i,j]$-dominates S_{b_i}. It then follows that: $\forall i,j : (S_{w_j} \cup R_{\overline{w}_j})$ $D_q[i,j]$-dominates S_{w_i}. Which means $(\mathcal{S}, \mathcal{R}_{\overline{w}})$ D_q-dominates V_w.

($\Leftarrow$) For this direction of the proof we have that there exists a partition $\mathcal{S} = (S_1, ...S_q)$ of V_w such that: $(\mathcal{S}, \mathcal{R}_{\overline{w}})$ D_q-dominates V_w. This means that $\forall i,j : (S_{w_j} \cup R_{\overline{w}_j})$ $D_q[i,j]$-dominates S_{w_i}. Let $\mathcal{S}_a, \mathcal{S}_b$ be the componentwise intersection of $\mathcal{S}_w$ with V_a and V_b respectively. We then have: $\forall i,j : (S_{w_j} \cup R_{\overline{w}_j})$ $D_q[i,j]$-dominates S_{a_i} and $\forall i,j : (S_{w_j} \cup R_{\overline{w}_j})$ $D_q[i,j]$-dominates S_{b_i}.

Hence $\forall i,j : (S_{a_j} \cup S_{b_j} \cup R_{\overline{w}_j})$ $D_q[i,j]$-dominates S_{a_i} and $\forall i,j : (S_{a_j} \cup S_{b_j} \cup R_{\overline{w}_j})$ $D_q[i,j]$-dominates S_{a_i}. Let $\mathcal{R}\overline{a} = can^d_{\overline{V_a}}(\mathcal{S}_b \bigcup_q \mathcal{R}_{\overline{w}})$ and $\mathcal{R}_{\overline{b}} = can^d_{\overline{V_b}}(\mathcal{S}_a \bigcup_q \mathcal{R}_{\overline{w}})$ then $\forall i,j : (S_{a_j} \cup R_{\overline{a}_j})$ $D_q[i,j]$-dominates S_{a_i} and $\forall i,j : (S_{b_j} \cup R_{\overline{b}_j})$ $D_q[i,j]$-dominates S_{b_i}. Let $\mathcal{R}_a = can^d_{V_a}(\mathcal{S}_a)$ and $\mathcal{R}_b = can^d_{V_b}(\mathcal{S}_b)$ then $Tab_a[\mathcal{R}_a][\mathcal{R}_{\overline{a}}] = TRUE$ and $Tab_b[\mathcal{R}_b][\mathcal{R}_{\overline{b}}] = TRUE$. Since the algorithm goes through all triples, it will at some point go through $(\mathcal{R}_a, \mathcal{R}_b, \mathcal{R}_{\overline{w}})$. And it will set $Tab_w[\mathcal{R}_w][\mathcal{R}_{\overline{w}}]$ to true, once it is true it will never change.

By induction all tables will be correct. □

Proof of Lemma 4

If $E(G) = \emptyset$, then all width parameters are 0 and there is nothing to show. If $E(G) \neq \emptyset$ and no two edges of G are adjacent, it is easy to see that $\mathbf{bw}(G) = 0$ and $\mathbf{rw}(I(G)) = \mathbf{boolw}(I(G)) = 1$. Now assume that $E(G)$ contains two adjacent edges. This implies $\mathbf{bw}(G) \geq 1$. Moreover, we may assume that G has no isolated vertices. (We can always add isolated vertices to a decomposition tree without increasing the **cut-bool**-width and the **cut-rk**-width.) Let (T, λ) be a decomposition tree of δ-width k of $G = (V, E)$. Now we modify (T, λ) in two steps.

In the first modification step, for every leaf $t \in L(T)$ we take the unique cubic tree T_t with four leaves $\ell_1^t \dots, \ell_4^t$ such that the distance between ℓ_1^t and ℓ_2^t is 2. We attach T_t to the tree T by identifying ℓ_4^t and t. In this way we obtain a new subcubic tree T' where every leaf t of T is replaced by three new leaves $\ell_1^t \ell_2^t, \ell_3^t$. We now define the function $\lambda' \colon L(T') \to V \dot\cup E$ as follows: Suppose $t \in L(T)$ is a leaf with $\lambda(t) = \{u, v\} \in E(G)$, assuming that if exactly one of the extremities of this edge has degree one, then it is u. Then we let $\lambda'(\ell_1^t) := u$, $\lambda'(\ell_2^t) := \{u, v\}$, and $\lambda'(\ell_3^t) := v$. In this way we obtain a labeled tree (T', λ') where λ' is surjective (because G has no isolated vertices) on vertices of G an. Now for every vertex $v \in V$ the preimage $\lambda'^{-1}(v)$ is nonempty, but is may contain more than one leaf of T', whereas for every $h \in E$ the preimage $\lambda'^{-1}(h)$ is a singleton.

In the second modification step, for every $v \in V$ we choose one $t_v \in L(T')$ with $\lambda'(t_v) = v$ and we prune all other leaves $\ell \in L(T')$, $\ell \neq t_v$, satisfying $\lambda'(\ell) = v$. Having done this for all $v \in V$, we obtain a subtree $T'' \subseteq T'$. Let λ'' denote the restriction of λ' to the leaves of T'' such that the pair (T'', λ'') is a decomposition tree of **cut-bool** on $I(G)$ and for **cut-rk** on $I(G)$. We claim that both **cut-bool**-w$((T'', \lambda'')) \leq k$ and **cut-rk**-w$((T'', \lambda'')) \leq k$. Towards this, let $e \in E(T'')$ be an edge.

First suppose e was already an edge in T. Then let (Y, Y^c) be the partition of $E(G)$ obtained by removing e from T in the decomposition tree (T, λ) of δ-width k. Let $S := \partial X$. Then $|S| \leq k$. Moreover, let (X, X^c) be the partition of $V(G)$ obtained by removing e from T'' in the decomposition tree (T'', λ''). Recall that $\lambda''(L(T'')) = V \dot\cup E$. Any set $Z \subseteq V \dot\cup E$ can be written as a disjoint union $Z = V_Z \dot\cup E_Z \dot\cup S_Z$, where $V_Z := (Z \cap V) \setminus S$, $E_Z := Z \cap E$ and $S_Z := Z \cap S$.

Consider the matrix $M_{(X,X^c)}$ as an $(S_X \dot\cup V_X \dot\cup E_X) \cdot (S_{X^c} \dot\cup V_{X^c} \dot\cup E_{X^c})$ matrix.

Claim. *Every non-zero entry in $M_{(X,X^c)}$ corresponds to an edge-vertex incidence where the vertex is in S.*

Proof of the Claim. First note that $E_X = Y$ and $E_{X^c} = Y^c$. Moreover, from our construction it follows that

$$v \in V_X \text{ if and only if } v \text{ is incident with an edge in } Y, \text{ and} \tag{1}$$
$$v \in V_{X^c} \text{ if and only if } v \text{ is incident with an edge in } Y^c. \tag{2}$$

Since $I(G)$ is bipartite, obviously, the only submatrices of $M_{(X,X^c)}$ possibly containing non-zero entries are induced by $V_X \cdot E_{X^c}$, $E_X \times V_{X^c}$, $X_S \times E_{X^c}$, and $E_X \cdot S_{X^c}$. Hence it suffices to show that the only submatrices that may contain non-zero entries are $X_S \cdot E_{X^c}$ and $E_X \times S_{X^c}$. Suppose towards a contradicion, that $V_X \cdot E_{X^c}$ has a non-zero entry. Then there is a vertex $v \in V_X$ and an edge $h \in E_{X^c} = Y^c$ such that v is incident with h. But, by (1), v is incident with an edge $h' \in Y$ as well, and hence $v \in S$, a contradiction. Symmetrically, (using (2) instead of (1)) $E_X \cdot S_{X^c}$ induces a zero submatrix. This proves the claim. ■

Hence, non-zero entries only correspond to edge-vertex incidences where the vertices are in S. Since $|S_X| + |S_{X^c}| = |S| \leq k$, it follows that both **cut-bool**-w$(e) \leq |S| \leq k$ and **cut-rk**-w$(e) \leq |S| \leq k$.

Finally, suppose e is an edge of some subcubic tree T_t created and attached to a leaf $t \in L(T)$ in the first modification step. Let $e_t \in E(T)$ be the edge incident with t. We first argue for boolean-width. If **cut-bool**$(e) \leq 1$ there is nothing to show, because $\mathbf{bw}(G) \geq 1$. Otherwise, **cut-bool**-w$(e) = 2$ or **cut-bool**-w$(e) = \log_2(3)$. But then it is easy to see that both ends of $\delta(t)$ have degree at least 2 and **cut-bool**-w$(e) \leq \delta$-w$(e_t) = 2$ and hence δ-w$(T'', \lambda'') \geq 2$. The argument for rank-width is analogous.

Altogether, it follows that both **cut-bool**-w$(T'', \lambda'') \leq k$ and **cut-rk**-w$(T'', \lambda'') \leq k$. Hence $\mathbf{boolw}(I(G)) \leq \mathbf{bw}(G)$ and $\mathbf{rw}(I(G)) \leq \mathbf{bw}(G)$. □

Proof of Lemma 5

Let (T, λ) be any decomposition tree of $I(G)$. We denote by $k = \mathbf{cut\text{-}bool}_{I(G)}$-w$(T, \lambda)$ and $r = \mathbf{cut\text{-}rk}_{I(G)}$-w$(T, \lambda)$ its boolean-width and rank-width, respectively. Let (T', λ') be the decomposition tree of G such that λ' is the restriction of λ to the elements of $V(G)$, and T' is obtained from T by deleting all leaves labeled by elements of $E(G)$ and contracting all internal nodes having degree 2. Let $k' = \mathbf{cut\text{-}bool}_G$-w$(T', \lambda')$ and $r' = \mathbf{cut\text{-}rk}_G$-w$(T', \lambda')$. We will prove that $\max\{k', r'\} \leq \min\{k, r\}$.

Let e' be an edge of T', and (X, X^c) the partition of $V(G)$ induced by the leaves of the trees we get by removing e' from T'. Since T' is obtained from "pruning" some leaves in T, there exists an edge e of T such that the partition (Y, Y^c) of $V(G)\dot{\cup}E(G)$, induced by the leaves of the trees we get by removing e from T, satisfies $X \subseteq Y$ and $X^c \subseteq Y^c$. In order to prove the lemma, we only need to prove that $\max\{\mathbf{cut\text{-}bool}_G(X), \mathbf{cut\text{-}rk}_G(X)\} \leq \min\{\mathbf{cut\text{-}bool}_{I(G)}(Y), \mathbf{cut\text{-}rk}_{I(G)}(Y)\}$.

Let $M_{(Y,Y^c)}$ denote the $Y \cdot Y^c$-submatrix of the adjacency matrix of $I(G)$. Since $V(G)$ and $E(G)$ are independent sets in $I(G)$, we assume w.l.o.g. that

$$M_{(Y,Y^c)} = \begin{pmatrix} 0 & A \\ B & 0 \end{pmatrix}$$

where the rows of A correspond to $Y \cap V(G)$, the columns of A to $Y^c \cap E(G)$, the rows of B to $Y \cap E(G)$, and the columns of B to $Y^c \cap V(G)$. Now remove all columns of A having two 1's, remove all rows of B having two 1's, remove all identical columns in what remains of A, remove all identical rows in what remains of B, remove all rows and columns with only 0-coordinates in what remains of the matrix, and obtain

$$M'_{(Y,Y^c)} \begin{pmatrix} 0 & A' \\ B' & 0 \end{pmatrix}.$$

This process can only decrease the value of either **cut-bool** or **cut-rk** functions. We will now prove that both A' and B' are permutation matrices. Indeed, since A comes from an incidence graph, it is straightforward to check that each column of A' now has exactly one 1. Suppose that a row of A' has more than one 1. Then, there are two columns of A' having coordinate 1 on this row. But both columns have only one 1, and therefore they are identical. Contradiction. Hence, each row of A' has exactly one 1, and A' is a permutation matrix. A similar argument holds for B'. Finally, let $V_{A'}$ be the set of rows of A' and $V_{B'}$ the set of columns of B' Since A' and B' are permutation matrices, both **cut-bool** and **cut-rk** values for $M'_{(Y,Y^c)}$ are equal to $|V_{A'}| + |V_{B'}|$. We have proved that $|V_{A'}| + |V_{B'}| \leq \min\{\textbf{cut-bool}_{I(G)}(Y), \textbf{cut-rk}_{I(G)}(Y)\}$. Note that the elements of $V_{A'}$ and $V_{B'}$ correspond to vertices of G.

Let $M_{(X,X^c)}$ denote the $X \cdot X^c$-submatrix of the adjacency matrix of G. We will split $M_{(X,X^c)}$ into two matrices M_1 and M_2 such that: both $GF(2)$-sum and Boolean-sum of M_1 and M_2 are equal to $M_{(X,X^c)}$; there are at most $|V_{A'}|$ non-zero rows in M_1; and there are at most $|V_{B'}|$ non-zero columns in M_2. This would allow to conclude the lemma because this would imply $\max\{\textbf{cut-bool}_G(X), \textbf{cut-rk}_G(X)\} \leq |V_{A'}| + |V_{B'}|$.

Let M_1 be the matrix where the rows correspond to X and the columns to X^c in such a way that they are ordered as in $M_{(X,X^c)}$ and there is a 1 at the intersection of the row corresponding to a vertex u and the column corresponding to a vertex v if and only if $uv \in (E(G) \cap Y^c)$ (uv is an edge of G and moreover uv and v belong to the same side of the cut of T given by e). Let M_2 be the matrix where the rows correspond to X and the columns to X^c such that they are ordered as in $M_{(X,X^c)}$ and there is a 1 at the intersection of the row corresponding to a vertex u and the column corresponding to a vertex v if and only if $uv \in (E(G) \cap Y)$ (uv is an edge of G and moreover uv and u belong to the same side of the cut of T given by e).

Let $u \in X$ and $v \in X^c$. If $uv \notin E(G)$, then, in any of the matrices $M_{(X,X^c)}$, M_1 and M_2, there is a 0 in the intersection of the row corresponding to u and the column corresponding to v. If $uv \in E(G)$, namely if uv belongs to either Y or Y^c but not both, then, beside $M_{(X,X^c)}$, we can find in one and only one among M_1 and M_2 a 1 in the intersection of the row corresponding to u and the column corresponding to v. Hence, both $GF(2)$-sum and Boolean-sum of M_1 and M_2 are equal to $M_{(X,X^c)}$.

Let $u \in X$ be a vertex such that there is at least one 1 in the row corresponding to u in M_1, namely $\exists v \in X^c$, $uv \in E(G) \cap Y^c$. Then, there is exactly one 1 in the column of $M_{(Y,Y^c)}$ corresponding to uv and at least one 1 in the row corresponding to u, and therefore there is still a column corresponding to uv and a row corresponding to u in $M'_{(Y,Y^c)}$. Hence, there are at most $|V_{A'}|$ non-zero rows in M_1. A similar argument holds for $|V_{B'}|$ and M_2. $\square$

Proof of Lemma 6

For the proof of Lemma 6 we need two lemmas on induced matchings. An *induced matching* in a graph G is a set $Y \subseteq E(G)$ of pairwise non-adjacent edges, such that no edge of $E(G) \setminus Y$ connects two vertices that are both incident to edges in Y. A graph G is *bipartite*, if $V(G)$ can be partitioned into two sets $V(G) = A \dot\cup B$ such that every edge of G connects a vertex in A to a vertex in B. Note that an induced matching in G corresponds to a submatrix S of $M_{(A,B)}$, which is a permutation matrix.

For an integer $r \geq 0$, let I_r denote the identity matrix with r rows and columns.

Lemma 10. *Let $r \geq 0$ be an integer. Let G be a bipartite graph witnessed by the partition $V(G) = A \dot\cup B$. Assume $M_{(A,B)}$ has at most two 1s in each column and at least r non-zero rows. Then G has an induced matching of cardinality $\lceil \frac{r}{2} \rceil$.*

Proof. We may assume that $M := M_{(A,B)}$ has no zero rows. It suffices to show that, by permuting the rows, permuting the columns and possibly deleting some rows and columns of M, we can obtain the matrix $I_{\lceil \frac{r}{2} \rceil}$. This is trivial for $r \leq 2$. Let $r \geq 3$. We choose a non-zero row with as few 1's as possible. We delete each column which has a 1 on this row except one. We move the row we chose to the bottom of the matrix, and we move the column which has a 1 on this row to the right of the matrix. All remaining rows are non-zero. Then we delete the other row which has a 1 on this column if it exists. We obtain a matrix

$$M^* = \begin{pmatrix} & & 0 \\ & M' & \vdots \\ & & 0 \\ 0 \dots 0 & & 1 \end{pmatrix},$$

where M' is a boolean matrix with at most two 1s in each column and at least $r-2$ non-zero rows. Applying the inductuve hypothesis ti M' we obtain the matrix $I_{\lceil \frac{r-2}{2} \rceil}$ by permuting the rows, permuting the columns and deleting some rows and columns of M'. Then, by applying the same transformation on M^*, we obtain $I_{\lceil \frac{r-2}{2} \rceil + 1} = I_{\lceil \frac{r}{2} \rceil}$. □

Lemma 11. *Let G be a bipartite graph witnessed by the partition $V(G) = A \dot\cup B$. Suppose G contains an induced matching of cardinality k. Then* $\mathbf{cut}\text{-}\mathbf{rk}(A) \geq k$ *and* $\mathbf{cut}\text{-}\mathbf{bool}(A) \geq k$.

Proof. Since an induced matching corresponds to a submatrix S of $M_{(A,B)}$, which is a permutation matrix, the lemma follows easily. □

Proof of Lemma 6. We prove $\mathbf{bw}(G) \leq 2 \cdot \mathbf{boolw}(I(G))$. Let (T, λ) be a decomposition tree of cutbool-width k of $I(G)$. Let (T', λ') be the decomposition tree of G, where T' is the subtree obtained from T by suppressing the leaves labeled by elements of $V(G)$, and λ' is the restriction of λ to $L(T')$. Let $e \in E(T')$ be an edge. Let (X, X^c) be the partition of $V(I(G))$ obtained by removing e from T in the decomposition tree (T, λ).

Since $I(G)$ is a bipartite graph witnessed by $V(I(G)) = V(G) \dot\cup E(G)$, we can write

$$M_{(X,X^c)} = \begin{pmatrix} 0 & M_1 \\ M_2 & 0 \end{pmatrix}$$

where the rows of M_1 correspond to $X \cap V(G)$, the columns of M_1 correspond to $X^c \cap E(G)$, the rows of M_2 correspond to $X \cap E(G)$ and the columns of M_2 correspond to $X^c \cap V(G)$. Thus we have $k \geq \mathbf{cut}\text{-}\mathbf{bool}(X) = \mathbf{cut}\text{-}\mathbf{bool}(X \cap V(G)) + \mathbf{cut}\text{-}\mathbf{bool}(X^c \cap V(G))$. Since the elements of $E(G)$ have degree 2 in $I(G)$, there are at most two 1s in any column of M_1 and in any row of M_2. Let r_1 be the number of non-zero rows of M_1. From Lemmas 10 and 11 it follows that $\lceil \frac{r_1}{2} \rceil \leq \mathbf{cut}\text{-}\mathbf{bool}(X \cap V(G))$ and hence $r_1 \leq 2 \cdot \mathbf{cut}\text{-}\mathbf{bool}(X \cap V(G))$. Symmetrically, letting r_2 be the number of non-zero columns of M_2, we have $r_2 \leq 2 \cdot \mathbf{cut}\text{-}\mathbf{bool}(X^c \cap V(G))$. Now, every vertex in the border $\partial(E(G) \cap X)$ is incident to an element of $E(G) \cap X$ and an element of $E(G) \cap X^c$, so it corresponds either to a non-zero row of M_1, or to a non-zero column of M_2. Hence $\delta(E(G) \cap X) \leq r_1 + r_2 \leq 2\big(\mathbf{cut}\text{-}\mathbf{bool}(X \cap V(G)) + \mathbf{cut}\text{-}\mathbf{bool}(X^c \cap V(G))\big) \leq 2k$. It follows that $\delta\text{-w}(T', \lambda') \leq 2k$, and hence $\mathbf{bw}(G) \leq 2 \cdot \mathbf{boolw}(I(G))$.

For a proof of $\mathbf{bw}(G) \leq 2 \cdot \mathbf{rw}(I(G))$, we merely replace the word ‘cutbool’ by ‘cutrk’ in the proof above. □

Proof of Claim 1

First, we claim that for a fixed I of size k, $\Pr[\, I \text{ is bad} \,] < e^{-\Omega(m^{0.75})}$. Let X_i be 1 if $i \notin S_I$, and 0 otherwise. Clearly, the random variables $\{X_i\}_{i=1}^m$ are independent, and $\Pr[\, X_i = 1 \,] = m^{-0.25}$. The expectation of $X = \sum_i X_i$ is $m^{0.75}$. Using the following Chernoff Bound for the left side tail (see e.g., [18]):

$$\Pr[\, X \leq (1-\delta) \cdot E[X] \,] \leq e^{\frac{-\delta^2 E[X]}{2}},$$

we conclude that
$$\Pr[\, I \text{ is bad} \,] \;=\; \Pr[\, X < m^{0.5} \,] \;<\; e^{-\Omega(m^{0.75})} .$$

Next, consider a random permutation $\sigma \in S_m$, and define the sets $I_j = \sigma\{(j-1)k+1, \ldots, jk\}$ where $j = 1, 2, \ldots\; t = m/k$. (To avoid messy calculations, assume for simplicity that k devides m.) We claim that the probability that the fraction of the bad sets in this family is 0.5 or more, is at most $e^{-\Omega(m^{1.74})}$. Let Y_j, $j = 1, 2, \ldots, t$, be an indicator random variable taking the value 1 if Y_j is bad, and 0 otherwise. Observe that Y_j's are independent, and that, by the previous discussion, $\Pr[\, Y_j = 1 \,] \;=\; e^{-\Omega(m^{0.75})}$. Let $Y = \sum_{j=1}^{t} Y_j$. Applying the Chernoff Bound we conclude that
$$\Pr\left[\, Y \geq t/2 \,\right] \;<\; \left(\frac{e \cdot E(Y)}{t/2}\right)^{t/2} \;<\; \left(2e \cdot e^{-\Omega(m^{0.75})}\right)^{\frac{m}{0.25 \cdot \log_2 m}} \;\leq\; e^{-\Omega(m^{1.74})} .$$

Observe that the number of permutations σ is only $m!$, and thus, applying the union bound, we conclude that with probability $1 - e^{-\Omega(m^{1.74})}$, for *all* σ's, the fraction of the bad sets in the corresponding family $\{I_j\}_{j=1}^{t}$ is less than 0.5. To conclude the proof of the claim, observe that the random family $\{I_j\}_{j=1}^{t}$ obtained by randomly choosing σ, uniformly covers the family of all I's of size k. Therefore, with the same probability, the fraction of the bad sets in the latter family is less than 0.5 as well. □

Proof of Claim 2

For $i \notin I$, let Z_i be an indicator random variable being 1 if $S_i \subseteq S_I$, and 0 otherwise. Clearly, Z_i's are independent, and $\Pr[\, Z_i = 1 \,] \;<\; 2^{-m^{0.5}}$. Let $Z = \sum_{i \notin I} Z_i$. Applying once more the Chernoff Bound, we conclude that
$$\Pr\left[\, Z \geq m^{0.9} \,\right] \;<\; \left(\frac{e \cdot E(Z)}{m^{0.9}}\right)^{m^{0.9}} \;<\; O\left(m^{0.1} \cdot 2^{-\Omega(m^{0.5})}\right)^{m^{0.9}} \;=\; e^{-\Omega(m^{1.3})}.$$

□

Proof of Corollary 4

Let B be the event that the number of good I's is at least $0.5 \cdot \binom{m}{k}$, and let C be the event that all good I's are thin (that is, not thick). Using Claim 1, Claim 2, and the union bound, we conclude that
$$\Pr[\overline{C}\,] \;\leq\; \sum_{I} \Pr[\, I \text{ is good and thick} \,] \;=\; \sum_{I} \Pr[\, I \text{ is thick} \mid I \text{ is good}] \;\cdot\; \Pr[\, I \text{ is good}]]$$
$$\leq\; \binom{m}{0.25 \cdot \log_2 m} \cdot e^{-\Omega(m^{1.3})} \;=\; e^{-\Omega(m^{1.3})} .$$

Let D be the event that there are at least $0.5 \cdot \binom{m}{k}$ thin I's. Clearly, $B \cap C \subseteq D$. Thus, by Claim 1 and Claim 2, $\Pr[\, D \,] \;\geq\; \Pr[\, B \cap C \,] \;\geq\; 1 - \Pr[\,\overline{B}\,] - \Pr[\,\overline{C}\,] \;\geq\; 1 - e^{-\Omega(m^{1.3})}$. □