REPORTS
IN
INFORMATICS

ISSN 0333-3590

H-join and algorithms on graphs of bounded rankwidth (SODA submission)[a]

Binh-Minh Bui-Xuan, Jan Arne Telle, Martin Vatshelle

REPORT NO 378 november 2008

[a]The content of this technical report is identical to the paper submitted to SODA on July 3rd 2008, using the EasyChair Conference System. The factor 1/4 in the exponent of the algorithm for q-coloring should be 1/2.

Department of Informatics

UNIVERSITY OF BERGEN

Bergen, Norway

H-join and algorithms on graphs of bounded rankwidth

Binh-Minh BUI-XUAN[a,†] Jan Arne TELLE[b,‡] Martin VATSHELLE[b,‡]

[a] CNRS-LIRMM-Université Montpellier II, France.
BinhMinh.BuiXuan@lirmm.fr

[b] Department of Informatics, University of Bergen, Norway.
[Jan.Arne.Telle,Martin.Vatshelle]@ii.uib.no

Abstract

Rankwidth is a graph parameter introduced by Oum and Seymour, based on ranks of adjacency matrices over GF(2). We propose an alternative definition of rankwidth, based on the graph-theoretical notion of H-joins, and give fast dynamic programming algorithms to solve optimization problems on graphs of bounded rankwidth. Such algorithms are interesting since graphs of rankwidth at most k encompass large classes of graphs, e.g., all graphs of treewidth $k+1$ or branchwidth k or cliquewidth k, some graphs of unbounded treewidth and branchwidth, and some graphs of cliquewidth $2^{k/2-1}-1$.

We introduce a graph composition operation called H-join, indexed by a fixed bipartite graph H, and define the H-join decomposable graphs to be those having an H-join decomposition. We show that any problem expressible in MSO_1-logic is fixed parameter tractable on an H-join decomposable graph when parameterized by $\rho(H)$, the rank of the adjacency matrix of H. Given an H-join decomposition of an n-vertex m-edge graph G we solve the Maximum Independent Set and Minimum Dominating Set problems on G in time $O(n(m+2^{O(\rho(H)^2)}))$, and the q-Coloring problem in time $O(n(m+2^{O(q\rho(H)^2)}))$.

For any positive integer k we define a bipartite graph R_k and show that the graphs of rankwidth at most k are exactly the R_k-join decomposable graphs. Moreover, a rank-decomposition of width k of a graph is also an R_k-join decomposition of the graph. For a graph G of rankwidth k, given with its width k rank-decomposition, this results in algorithms which, in $O(n(m+2^{\frac{1}{2}k^2+\frac{9}{2}k}\times k^2))$ time solve the Maximum Independent Set problem on G, in $O(n(m+2^{\frac{5}{4}k^2+\frac{29}{4}k}\times k^6))$ time solve the Minimum Dominating Set problem on G, and in $O(n(m+2^{\frac{q}{4}k^2+\frac{5q+8}{4}k}\times k\times q))$ time solve the q-Coloring problem on G. These are the first algorithms for NP-hard problems whose runtimes are less than double exponential in the rankwidth k.

1 Introduction

Many variants of graph decompositions have been studied, like tree decompositions, modular decompositions, rank decompositions and split decompositions, see the recent survey paper by Hliněný et al [16] for a good overview. Several decompositions define a graph "width" parameter, with the most important from an algorithmic point of view being, in order of discovery: treewidth, branchwidth, cliquewidth and rankwidth. The first two of these parameters are "less powerful" than the last two, in the sense that a graph class has bounded treewidth iff it has bounded branchwidth [23], it has bounded cliquewidth iff it has bounded rankwidth [21], and if it has bounded treewidth then it has bounded cliquewidth but not the other way around [4]. The rankwidth of a graph is never larger than its cliquewidth, nor its branchwidth, nor its treewidth plus one [20]. In this sense rankwidth, which has been investigated quite heavily in recent years [6, 7, 15, 19, 20] and which is the focus of this paper, is the most powerful of the four parameters.

Many NP-hard graph optimization problems have fixed-parameter tractable (FPT) algorithms when parameterized by these graph width parameters, see [16] for an overview. Such algorithms usually have two

[†] Supported by the French National Research Agency, project GRAAL.
[‡] Supported by the Norwegian Research Council, project PARALGO.

stages, a first stage computing the right decomposition of the input graph and a second stage solving the problem using the decomposition. For a long time there was no good first stage algorithm for cliquewidth, and rankwidth was in fact introduced originally as a tool to help compute a decomposition for cliquewidth [19]. Recently, Hliněný and Oum found an FPT algorithm that given a graph G on n vertices and a parameter k will decide if G has rankwidth at most k and if so output a rank decomposition of width k in time $O(f(k)n^3)$ [15]. Between rankwidth $rw(G)$ and cliquewidth $cw(G)$ we have the connection $rw(G) \leq cw(G) \leq 2^{rw(G)+1}$ [21]. Moreover, a rank decomposition of width k of G can be turned into a (2^{k+1})-expression that is then used as the cliquewidth decomposition of G. Note that in going from rankwidth to cliquewidth some exponential jump is required, as it follows by results of Corneil and Rotics [4] that for any k there is a graph G with rankwidth k and cliquewidth at least $2^{k/2-1} - 1$. Because of this exponential jump from rankwidth to cliquewidth it is clear that based on the first stage of the usual FPT process rankwidth should be better suited than cliquewidth for getting the fastest possible algorithms. The problem with rankwidth has been that the rank decomposition has until now not been known to be amenable to the second stage that involves dynamic programming along the decomposition. In this paper we rectify this situation, by showing how to enhance a rank decomposition with additional information that will make it amenable to dynamic programming.

Our main tool is a new graph composition operation that we call H-join, indexed by a fixed bipartite graph H. This operation adds edges between two given graphs by taking partitions of their two vertex sets, identifying the classes of the partitions with vertices of H, and connecting classes by the pattern H. The formal definitions are given in Section 2. We also define an H-join decomposition (T, δ) of a graph, where T is a subcubic tree and δ is a bijection between the leaves of T and the vertex set of G, as in a rank decomposition and in branch decomposition. A graph having an H-join decomposition will be called an H-join decomposable graph. We define, for any positive integer k, a bipartite graph R_k having 2^k vertices in each color class, and show that the graphs of rankwidth at most k are exactly the R_k-join decomposable graphs and that a rank decomposition of width k is an R_k-join decomposition.

In Section 3 we give dynamic programming algorithms that follow an H-join decomposition of a graph G to solve various problems on G. We express the runtime of these algorithms as a function of the rank $\rho(H)$ of the adjacency matrix of H. Using the connection to rank decompositions this gives us algorithms for graphs of bounded rankwidth that work directly on the rank decomposition. This allows designing algorithms for various optimization problems on graphs of bounded rankwidth that are faster than those we get by exploiting the connection between rankwidth and cliquewidth. The generic result of Courcelle et al stating that any MSO_1-logic problem is FPT for bounded cliquewidth [5, 13] results in running times that are at least double exponential in cliquewidth. For some problems faster FPT algorithms have been designed, see e.g. [10, 12, 18, 22], but apart from the algorithm for Minimum Dominating Set in [18] they all have a factor which is double exponential in cliquewidth, and thereby triple exponential in rankwidth. For a graph of rankwidth k given with its rank decomposition, the result of [18] yields an algorithm solving Minimum Dominating Set whose runtime will have a factor $(2^{2^k})^4$. For Maximum Independent Set one can without too much effort design an algorithm with factor $2^{2^{k+1}}$. Our result significantly improves on these algorithms by replacing the exponential factors in the exponent of the runtimes with a quadratic factor:

Theorem 1.1 *For a graph G of rankwidth k, given with its rank-decomposition, we can solve the Maximum Independent Set problem in time $O(n(m + 2^{\frac{1}{2}k^2+\frac{9}{2}k} \times k^2))$, the Minimum Dominating Set problem in time $O(n(m + 2^{\frac{5}{4}k^2+\frac{29}{4}k} \times k^6))$, and the q-Coloring problem in time $O(n(m + 2^{\frac{q}{4}k^2+\frac{5q+8}{4}k} \times k \times q))$.*

2 H-join decomposable graphs and rankwidth

Definition 2.1 *Let H be a bipartite graph with color classes V_1 and V_2, thus $V(H) = V_1 \cup V_2$. Let G be a graph and $S \subseteq V(G)$ a subset of its vertices. We say that G is an H-join across the ordered cut $(S, V(G) \setminus S)$ if there exists a partition of S with set of classes P and a partition of $V(G) \setminus S$ with set of classes Q, and injective functions $f_1 : P \rightarrow V_1$ and $f_2 : Q \rightarrow V_2$, such that for any $x \in S$ and $y \in V(G) \setminus S$ we have x adjacent to y in G if and only if x belongs to a class P_i of P and y to a class Q_j of Q with $f_1(P_i)$ adjacent to $f_2(Q_j)$ in H. We say that G is an H-join across the non-ordered cut*

$\{S, V(G) \setminus S\}$ if G is an H-join across either $(S, V(G) \setminus S)$ or $(V(G) \setminus S, S)$.

Twins in a bipartite graph are vertices in the same color class having exactly the same neighbourhood. A twin contraction is the deletion of a vertex when it has a twin. Notice that H-joins are insensitive to twin contractions: if H' is obtained from H by a twin contraction then G is an H-join across some cut if and only if G is an H'-join across the same cut. In the remainder of the paper we therefore assume that H is a graph with no twins in the same color class. However, note that we do allow one isolated vertex in each color class. We will decompose graphs by H-joins in a way analogous to branch decompositions. A subcubic tree is an unrooted tree where all internal nodes have degree three.

Definition 2.2 *Let T be a subcubic tree and δ a bijection between the leaf set of T and the vertex set of a graph G. We say that (T, δ) is an H-join decomposition of G if for any edge uv of T we have G being an H-join across the cut $\{S_u, S_v\}$ we get from the 2-partition of $V(G)$ induced by the leaf sets of the two subtrees we get by removing uv from T. A graph having an H-join decomposition will be called an H-join decomposable graph.*

One feature of studying such a tree-like decomposition is that we can think of the decomposition as the collection of cuts of the initial graph given by the collection of edges of the subcubic tree. Under this standpoint, H-join decomposition is related to both modular decomposition [11] and split decomposition [9]. Indeed, saying $M \subseteq V(G)$ is a module of G is exactly equivalent to saying that G is a P_2^+-join across the ordered cut $(M, V(G) \setminus M)$, where P_2^+ is obtained by adding an isolated vertex to the second colour class of the bipartite graph P_2. Therefore we have for instance that a cograph – a graph where every induced subgraph of at least four vertices has a non-trivial module – is always P_2^+-join decomposable, and that a P_2^+-join decomposition of the cograph can be obtained from its modular decomposition tree by unrooting the tree and subdividing arbitrarily all internal nodes of degree more than three. The link between modular decomposition and H-join decomposition is reflected also in the definition of external module partitions that will be fundamental when studying the partitions used for an H-join across a cut:

Definition 2.3 *Let G be a graph and let $S \subseteq V(G)$ be a vertex subset. An* external module partition of S *is a partition P of S such that, for every $z \in V(G) \setminus S$ and pair of vertices x, y belonging to the same class in P, we have x adjacent to z if and only if y adjacent to z.*

As for split decomposition, saying that a cut $\{S, V(G) \setminus S\}$ is a split is exactly equivalent to saying that G is a P_2^{++}-join across $\{S, V(G) \setminus S\}$, where P_2^{++} is obtained by adding one isolated vertex to each colour class of the bipartite graph P_2. Here, we have a stronger fact than that with modular decomposition: a graph is distance hereditary – meaning a graph where every induced subgraph of at least five vertices has a non-trivial split – if and only if it is P_2^{++}-join decomposable. Moreover, there is a straightforward manner to obtain a P_2^{++}-join decomposition from the split decomposition tree of the distance hereditary graph, and conversely.

Other particular cases of H-join decompositions include the so-called 2-join [8], and also the so-called generalized join, itself a particular case of so-called 1-separations [17]. More precisely, 2-joins are related to H-joins when H is equal to $2P_2^{++}$, the graph we obtain by adding one isolated vertex to each colour class of the bipartite graph made by a disjoint union of two P_2. At the same time, Hsu's generalized joins are related to H-joins as soon as H admits orderings of colour classes $V_1 = (v_1^1, v_1^2, \ldots, v_1^{k+1})$ and $V_2 = (v_2^1, v_2^2, \ldots, v_2^{k+1})$ such that $N_H(v_i^1) = \{v_2^1, v_2^2, \ldots, v_2^{k+1-i}\}$. Both 2-join and Hsu's generalized join decompositions are important for decomposing perfect graphs, with the former decomposition playing a central role in the recent proof of the strong perfect graph theorem by Chudnovsky et al [3]. This result has been known as one of the major challenges in graph theory, and was conjectured by C. Berge half a century ago.

We now turn to the strong connections between H-join decompositions and rank decompositions. We first recall the definition of rankwidth. For any graph G, the cut-rank function ρ_G is defined over every vertex subset $X \subseteq V(G)$ as the rank of the $X \times V(G) \setminus X$ submatrix of the adjacency matrix of G. For any pair (T, δ) with T a subcubic tree and δ a bijection between vertices of G and leaves of T, (T, δ) is defined as a width r rank decomposition of G if for all edge uv in T, the cut-rank of S_u is at most r, where $\{S_u, S_v\}$ is the 2-partition of $V(G)$ induced by the leaf sets of the two subtrees we get by removing uv from T. The rankwidth of G is the minimum r such that there exists a width r rank decomposition of G.

Definition 2.4 *For a positive integer k we define a bipartite graph R_k having for each subset S of $\{1, 2, ..., k\}$ a vertex $a_S \in A$ and a vertex $b_S \in B$, with $V(R_k) = A \cup B$. This gives 2^k vertices in each of the color classes. Two vertices a_S and $b_{S'}$ are adjacent iff $|S \cap S'|$ is odd.*

Lemma 2.5 *The function $\sigma_G : 2^{V(G)} \to \mathbb{N}$ defined by*

$$\sigma_G(X) = \min\{k : G \text{ is an } R_k\text{-join of } G \text{ across the cut } \{X, V(G) \setminus X\}\}$$

is equal to the cut-rank function ρ_G.

Theorem 2.6 *(T, δ) is a width k rank decomposition of G if and only if (T, δ) is an R_k-join decomposition of G. Thus G is a graph of rankwidth at most k if and only if G is an R_k-join decomposable graph.*

The proof of Lemma 2.5 has been moved to the appendix. Theorem 2.6 follows directly from that lemma. Now, the following straightforward observation shows how, on the other hand, H-join decompositions can be embedded in a rank decomposition of reasonable width.

Theorem 2.7 *To any bipartite graph H we can apply twin contractions to get an induced subgraph of $R_{\rho(H)}$, where $\rho(H)$ is the rank of the bipartite adjacency matrix of H. A consequence is that if G is an H-join decomposable graph then G is also an $R_{\rho(H)}$-decomposable graph. In other words, the rankwidth of an H-join decomposable graph is at most $\rho(H)$.*

The above observation, though simple, implies that H-join decompositions inherit algorithmic results of rank decompositions. For instance, we immediately get the following.

Theorem 2.8 *Any problem expressible in monadic second-order logic with quantifications over vertex sets (MSO$_1$-logic) can be solved in FPT time for H-join decomposable graphs when parameterized by $\rho(H)$.*

This follows since it is true when parameterized by cliquewidth [5], hence when parameterized by rankwidth because of the bound between cliquewidth and rankwidth, hence when parameterized by $\rho(H)$ by Theorem 2.7. More generally, any FPT algorithm on an H-join decomposable graph that is parameterized by the rankwidth of the graph is also an FPT algorithm when parameterized by $\rho(H)$. Examples of problems outside of MSO$_1$-logic include those addressed in [10, 12, 18, 22], however, note that some of the solutions given therein do not have FPT runtime.

Applying the algorithm of [15] to an H-join decomposable graph G will in time $O(f(k)n^3)$ give an $R_{\rho(H)}$-join decomposition of G with the property that every H'-join across the cut defined by any edge of the subcubic tree satisfies $\rho(H') \leq \rho(H)$. In the next section we give dynamic programming algorithms that work along H-join decompositions, which by Theorem 2.6 are more general than rank decompositions. For some H, like $H = P_2^+$ and $H = P_2^{++}$ the H-join decomposition is easy to compute, while for other H we can assume a rank decomposition is given and apply Theorem 2.7.

3 Dynamic Programming

In this section we give dynamic programming algorithms to solve various problems on an H-join decomposable graph G, given with its H-join decomposition (T, δ). A potential drawback of defining H-join decompositions simply as the pair (T, δ) is that for an edge uv of T we *a priori* do not know the partition classes P of S_u and Q of S_v mentioned in Definition 2.1, to confirm that G is an H-join across the cut $\{S_u, S_v\}$. We now show how to compute this information.

Given as input a graph G and a vertex subset $S \subseteq V(G)$, we can compute a maximum external module partition (see Definition 2.3) of S, which is well-defined by Lemma 3.1, using partition refinement techniques: just initialize a partition as $P = \{S\}$; then, for every exterior vertex $z \in V(G) \setminus S$, refine P using the neighbourhood of z as pivot. These operations can be done in $O(m)$ time, with $m = |E(G)|$, since each refinement operation can be done in time proportional to the size of the pivot set (for more details refer to [14]). The correctness is stated in the following lemma, whose simple proof has been moved to the appendix.

Lemma 3.1 *Let G be a graph and S be a vertex subset of $V(G)$. The maximum (coarse-wise) external module partition of S is well-defined and can be computed in $O(|E(G)|)$ time.*

Maximum external module partitions have the following property, essential for computational purposes on H-join decompositions:

Proposition 3.2 *Let (T, δ) be an H-join decomposition of G. Let P_u, P_v be the maximum external module partitions of respectively S_u and S_v, where $\{S_u, S_v\}$ is the 2-partition of $V(G)$ we get by deleting an edge uv in T. Let R_u and R_v be two sets containing exactly one vertex per part in respectively P_u and P_v and let H' be the bipartite graph defined by the bipartite adjacency in G between R_u and R_v. Then H' is an induced subgraph of H, and therefore G is an H-join across the cut $\{S_u, S_v\}$ using partitions P_u and P_v.*

Proposition 3.2 follows mainly from the maximality of P_u and P_v, and the fact that if G is an H-join across $\{S_u, S_v\}$ using partitions Q_u and Q_v, then Q_u is an external module partition of S_u. A more detailed proof can be found in the appendix. For the dynamic programming, we subdivide an arbitrary edge of T to get a new root node r, and denote by T_r the resulting rooted tree. The algorithms will follow a bottom-up traversal of T_r. With each node w of T_r we associate a data structure *table*, that will store optimal solutions to subproblems restricted to the graph $G[V_w]$, where V_w are the vertices of G mapped to leaves of the subtree of T_r rooted at w. Each index of the table will be associated with a class of subproblems. These classes of subproblems will be related to the following equivalence classes of vertex subsets.

Definition 3.3 *For a fixed graph G and vertex subset $A \subseteq V(G)$, consider two vertex subsets $X \subseteq A$ and $Y \subseteq A$ and define $X \equiv_A Y$ if and only if $N(X) \setminus A = N(Y) \setminus A$.*

Note that $\equiv_A$ is an equivalence relation on subsets of A, with two equivalent sets having the same neighbors outside A. For a node a of T_r we will be interested in the equivalence relation $\equiv_{V_a}$ on V_a, the subset of vertices of G mapped to leaves of the subtree of T_r rooted at a. We begin with showing how to compute canonical representatives for the equivalence classes of $\equiv_{V_a}$. By applying the previous computation we have that the graph G is an H-join across the cut $\{V_a, V(G) \setminus V_a\}$ using partitions P_a of V_a and Q_a of $V(G) \setminus V_a$, with P_a and Q_a being maximum external module partitions. Consider arbitrary orderings $P_a(1), P_a(2), ..., P_a(h1)$ of the classes of P_a, and $Q_a(1), Q_a(2), ..., Q_a(h2)$ of the classes of Q_a. Let v_i be an arbitrary element of $P_a(i)$, for $1 \leq i \leq h1$, and let u_i be an arbitrary element of $Q_a(i)$, for $1 \leq i \leq h2$. Let H' be the bipartite graph induced by edges between the two vertex sets $\{v_1, v_2, ..., v_{h1}\}$ and $\{u_1, u_2, ..., u_{h2}\}$. Thus, $N_{H'}(v_i)$ will denote the neighbors of v_i in H'. Besides, we will also denote the neighbourhood of a vertex subset by $N_{H'}(X) = \bigcup_{x \in X} N_{H'}(x) \setminus X$. ¿From Proposition 3.2 we have H' an induced subgraph of H.

Given $X \subseteq V_a$ we compute a canonical representative $can_{V_a}(X)$ for $[X]_{\equiv_{V_a}}$, the equivalence class of $\equiv_{V_a}$ containing X, as follows. First, compute the set $X_{H'} = \{v_i : X \cap P_a(i) \neq \emptyset\}$. Then, initialize $can_{V_a}(X)$ and W to the emptyset. Finally, for $i = 1$ to $h1$, if $N_{H'}(v_i) \subseteq N_{H'}(X_{H'})$ and $N_{H'}(v_i) \setminus W \neq \emptyset$ then add v_i to $can_{V_a}(X)$ and add the vertices in $N_{H'}(v_i) \setminus W$ to W. Note that we could have broken out of the for loop as soon as $W = N_{H'}(X_{H'})$.

Lemma 3.4 *For $X \subseteq V_a$ the algorithm given above computes a canonical representative $R = can_{V_a}(X)$ for the class $[X]_{\equiv_{V_a}}$ in time $O(|X| + |E(H)|)$.*

Proof We first prove that X and R are in the same class, namely that $N_G(X) \setminus V_a = N_G(R) \setminus V_a$. Indeed, P_a is an external module partition of V_a and thus for any $v \in X$ with $v \in P_a(i)$ we have

$N_G(v) \setminus V_a = N_G(v_i) \setminus V_a$. Thus $N_G(X) \setminus V_a = N_G(X_{H'}) \setminus V_a$. ¿From the test in the for loop we have immediately that $N_{H'}(X_{H'}) = N_{H'}(R)$. Since Q_a is an external module partition of $V(G) \setminus V_a$ we have that $u_j \in N_{H'}(v_i)$ if and only if $Q_a(j) \subseteq N_G(v_i)$ and thus $N_{H'}(X_{H'}) = N_{H'}(R)$ implies that $N_G(X_{H'}) \setminus V_a = N_G(R) \setminus V_a$ which proves the claim. As for the uniqueness of R, for any $Y \subseteq V_a$ with $X \equiv_{V_a} Y$, we get $can_{V_a}(X) = can_{V_a}(Y)$ since the algorithm loops through the vertices $v_1, v_2, ...v_{h1}$ in fixed order. Finally, for complexity issues note that the first action of the algorithm takes time $O(|X|)$, and in the for loop we check every edge of the graph H at most once. □

When expressing the complexity analysis in function of $|E(H)|$ as in the above, it is important to check that the assumption where H has no twins in the same colour class is well-behaved w.r.t. the analysis. Basically, all situations in the paper where we need to take $|E(H)|$ into account start with computing the corresponding maximum external module partitions, e.g., P_a and Q_a in the above, then take one representative vertex per class of the partitions, e.g., $\{v_1, v_2, ..., v_{h1}\}$ and $\{u_1, u_2, ..., u_{h2}\}$ in the above, and after those steps the adjacency in G between the representative vertices can be used in order to express the adjacency in H. Then, not only Proposition 3.2 states that the adjacency in G between the representative vertices then defines an induced subgraph of H, but also one can check that the adjacency in G then defines a bipartite graph having no twins in the same colour class. Accordingly, the part of the runtime denoted by $|E(H)|$ does not change whether we suppose H to be twin-free or not and w.l.o.g. such a supposition can be made. Situations where this consideration is important include the complexity analysis given in the abstract where we assume H has no twins in the same colour class and use the naive bound $|E(H)| \leq |V(H)|^2 \leq 2^{2\rho(H)}$ in order to give a bound for the analysis.

We now use linear algebra and show in the following result that the number of vertices in a canonical representative of an equivalence class of $\equiv_{V_a}$ is at most $\rho(H)$, the rank of the adjacency matrix of H. We also bound neq, the number of equivalence classes of $\equiv_{V_a}$, using the number of pairwise distinct subspaces in a given space over $GF(2)$. The latter number can be expressed using q-binomial coefficients. Then, by using the $q-$analog of Pascal triangles, a bound for neq can also be obtained. The proof of the proposition has been moved to the appendix.

Proposition 3.5 *For any vertex subset $X \subseteq V_a$, the number of vertices belonging to the canonical representative $R = can_{V_a}(X)$ for the class $[X]_{\equiv_{V_a}}$ is at most $\rho(H)$, the rank of the bipartite adjacency matrix of H. Moreover, the number neq of equivalence classes of $\equiv_{V_a}$ is at most $neq \leq 2^{\frac{1}{4}\rho(H)^2 + \frac{5}{4}\rho(H)}\rho(H)$.*

Roughly, our algorithmic idea behind the classification of subsets of V_a w.r.t. the equivalence classes of $\equiv_{V_a}$ is to use the above upper bound for speeding up purposes. However, a potential drawback could come if we would have to parse the subsets of V_a in order to look for some equivalence class of $\equiv_{V_a}$. Fortunately enough, the previous definition of canonical representatives comes in handy for such a situation since there is a fast and simple manner to output the list C_a containing all of them:

```
initialize the list $C_a$ to contain $\{\emptyset\}$
for $i = 1$ to $h1$
    for all $R \in C_a$
        if ($R \cup v_i == can_{V_a}(R \cup v_i)$)
            add the set $R \cup v_i$ to $C_a$
```

Theorem 3.6 *R belongs to C_a if and only if R is a canonical representative of $\equiv_{V_a}$. Moreover, C_a can be outputted in $O(h_1 * neq * |E(H)|)$ time, where neq is the number of classes of $\equiv_{V_a}$.*

Proof ($\Rightarrow$) Since R is in C_a some $R' \cup v_i = R$ must have passed the check "if ($R' \cup v_i = can_{V_a}(R \cup v_i)$)" hence, R is a canonical representative.
($\Leftarrow$) Assume R is a canonical representative and v_i is the element in R with highest index i. If $R = \{v_i\}$, then $R \in C_a$. Assume inductively that this is true for all representatives of size less than $|R|$, then R would be added to C_a iff $R \setminus v_i$ is in C_a and hence is a canonical representative. By definition of canonical representatives $N_{H'}(R \setminus v_i) \neq N_{H'}(R)$, the only vertex that sees any nodes of $X = N_{H'}(R) \setminus N_{H'}(R \setminus v_i)$ is v_i. The algorithm computing $can_{V_a}(R \setminus v_i)$ goes through the nodes $v_1, v_2, ...v_{i-1}$ in the same way as

for $can_{V_a}(R)$, they both only pick vertices in $can_{V_a}(R \setminus v_i)$ since no node before v_i sees any node in X. This means $can_{V_a}(R \setminus v_i) = can_{V_a}(R) \setminus v_i = R \setminus v_i$ hence $R \in C_a$. By induction the result follows.

The runtime follows from Lemma 3.4 since the calls to $can_{V_a}(X)$ always satisfy $|X| = O(|E(H)|)$. □

3.1 Maximum Independent Set

We consider the problem of computing the size of a maximum independent set. The *table* data structure Tab_a associated with node a of T_r will then have an index set that contains all indices $\{can_{V_a}(X) : X \subseteq V_a\}$, i.e. the elements of the list C_a. Recall from Proposition 3.5 that no canonical representative has more than $\rho(H)$ elements from $v_1, v_2, ..., v_{h1}$. The table Tab_a associated with node a of T_r will actually have index set consisting of all subsets of at most $\rho(H)$ elements from $1, 2, ..., h1$, and an index corresponding to a canonical representative R will have a pointer to R in the list C_a. This way we achieve $O(1)$ accesses to the table and also we can loop through all canonical representatives in $|C_a|$ time.

For $R = can_{V_a}(X)$ the contents of $Tab_a[R]$ after processing a should be the size of the largest independent set contained in the equivalence class of R, in other words

$$Tab_a[R] \stackrel{\text{def}}{=} \max_{S \subseteq V_a}\{|S| : S \equiv_{V_a} R \wedge \forall x, y \in S \Rightarrow xy \notin E(G)\}$$

At a leaf w of T_r associated to a node x of G we have a partition of $V_w = \{x\}$ into two equivalence classes and set $Tab_w[\emptyset] = 0$ and $Tab_w[\{x\}] = 1$. For an internal node w of T_r with children a and b whose tables have already been processed, we process the table of w as follows:

```
initialize all values of $Tab_w$ to 0
for all indices $R_a$ in $Tab_a$ and $R_b$ in $Tab_b$
    if ($Ra \cup R_b$ is an independent set) then
        $R_w := can_{V_w}(R_a \cup R_b)$
        $Tab_w[R_w] := \max\{Tab_w[R_w], Tab_a[R_a] + Tab_b[R_b]\}$
```

Lemma 3.7 *The table of an internal node w having children a, b is updated correctly.*

Proof Let R_w be a canonical representative of $\equiv_{V_w}$. Assume $I \subseteq V_w$ is an independent set such that $I \equiv_{V_w} R_w$, we first show that $Tab_w[R_w] \geq |I|$. Let $I_a = I \cap V_a$ and $I_b = I \cap V_b$. Clearly, I_a and I_b are independent sets, and therefore by an inductive argument on the correctness of the tables of a and b we have that $Tab_a[can_{V_a}(I_a)] \geq |I_a|$ and $Tab_b[can_{V_b}(I_b)] \geq |I_b|$. The update procedure at node w will thus set $Tab_w[can_{V_w}(can_{V_a}(I_a) \cup can_{V_b}(I_b))] \geq |I_a| + |I_b| = |I|$. To conclude, we simply need to prove the claim that $R_w = can_{V_w}(can_{V_a}(I_a) \cup can_{V_b}(I_b))$. By expressing $can_{V_a}(I_a) \equiv_{V_a} I_a$ and $can_{V_b}(I_b) \equiv_{V_b} I_b$, we have $N(can_{V_a}(I_a) \cup can_{V_b}(I_b)) \setminus V_w = N(I_a \cup I_b) \setminus V_w = N(I) \setminus V_w = N(R_w) \setminus V_w$, In other words, $R_w \equiv_{V_w} can_{V_a}(I_a) \cup can_{V_b}(I_b)$, and this proves the claim since R_w is a canonical representative.

To conclude the lemma, we need to prove that if $Tab_w[R_w] = k$ then there exists an independent set $I \subseteq V_w$ with $|I| = k$ and $I \equiv_{V_w} R_w$. For this, note that the algorithm increases the value of $Tab_w[R_w]$ only if there exist indices R_a in Tab_a and R_b in Tab_b such that $R_a \cup R_b$ is an independent set. Moreover, if $I_a \equiv_{V_a} R_a$ and $I_b \equiv_{V_b} R_b$, then the fact $R_a \cup R_b$ is an independent set can be used to prove that $I_a \cup I_b$ is also an independent set. □

At the root r of T_r we have $V_r = V(G)$ and thus no outside neighbors to distinguish vertices into distinct equivalence classes, so that the single entry of its table will store the size of the maximum independent set of G.

For the runtime note that we first computed, for each of the $O(n)$ nodes of T_r, the maximum external module partitions in $O(m)$ time, the canonical representatives in time $O(h1 \times neq \times |E(H)|)$ and filled the table at this node in time $O(neq^2|E(H)|)$, where neq is the number of equivalence classes (loop over all pairs). This gives a total runtime of $O(n(m+neq^2|E(H)|))$. Using the bound on neq from Proposition 3.5 we thus get:

Theorem 3.8 *Given a graph G on n nodes and m edges, and an H-join decomposition (T, δ) of G, we can in $O(n(m + 2^{\frac{1}{2}\rho(H)^2 + \frac{5}{2}\rho(H)}\rho(H)^2|E(H)|))$ time solve the Maximum Independent Set problem on G, where $\rho(H)$ is the rank of the adjacency matrix of H.*

Notice that the problem of finding a maximum clique of a given graph G can be solved by finding a maximum independent set in $\overline{G}$, the complement of G. Moreover, any H-join decomposition of G can be used as an $\overline{H}$-join decomposition of $\overline{G}$, and besides $\rho(\overline{H}) \leq \rho(H) + 1$. Therefore, the Maximum Clique problem can be solved using the "Independent Set" algorithm with a runtime bounded by $O(n(m + 2^{\frac{1}{2}\rho(H)^2 + \frac{7}{2}\rho(H)}\rho(H)^2|E(\overline{H})|))$.

3.2 Minimum Dominating Set and q-Coloring

We consider the problem of computing the size of a minimum dominating set. Naively generalizing from the independent set algorithm we may think that the table at a node w of T_r should store the size of a smallest dominating set D for $G[V_w]$. However, unlike the case of independent sets we note that a dominating set D will include also vertices of $V(G) \setminus V_w$ that dominate vertices of V_w 'from the outside'. This complicates the situation. Denote $V(G) \setminus V_w$ by $\overline{V_w}$. The main idea for dealing with this complication is to index the table at w by two sets, one that represents the equivalence class under $\equiv_{V_w}$ of $D \cap V_w$ that dominate 'from the inside', and one that represents the equivalence class under $\equiv_{\overline{V_w}}$ of $D \cap \overline{V_w}$ that help dominate the rest of V_w 'from the outside'. In the table update procedure when we join two subgraphs to form a bigger subgraph we use the union of 'the inside' dominators as the 'inside' dominator, and what remains to be dominated 'from the outside' is the union of what needed to be dominated 'from the outside' in the children minus the parts that the 'inside' in one child dominates in the other child.

Definition 3.9 *For $X \subseteq V_w$ and $Y \subseteq \overline{V_w}$ we say that (X, Y) dominates $G[V_w]$ if V_w is a subset of $X \cup N(X) \cup N(Y)$.*

For this algorithm, the table Tab_w associated with a node w of T_r will have index set $\{can_{V_w}(X) \times can_{\overline{V_w}}(Y) : X \subseteq V_w, Y \subseteq \overline{V_w}\}$. We define the contents of $Tab_w[R_X][R_Y]$ where $R_X = can_{V_w}(X)$ and $R_Y = can_{\overline{V_w}}(Y)$ as:

$$Tab_w[R_X][R_Y] \stackrel{\text{def}}{=} min_{S \subseteq V_w}\{|S| : S \equiv_{V_w} R_X \wedge (S, R_Y)\ dominates\ G[V_w]\}.$$

At a leaf w of T_r corresponding to a vertex x of G, there are at most four entries in Tab_w. Let $R = can_{\overline{V_w}}(\overline{V_w})$ then $Tab_w[\{x\}][R] = 1, Tab_w[\{x\}][\emptyset] = 1, Tab_w[\emptyset][R] = 0$ and $Tab_w[\emptyset][\emptyset] = \infty$. For simplicity we assume that G is connected. Let a and b be two nodes of T_r with w their common parent. Let H'_a, H'_b and H'_w be the induced subgraphs of H across the cuts $\{V_a, \overline{V_a}\}, \{V_b, \overline{V_b}\}$ and $\{V_w, \overline{V_w}\}$ respectively. Similarly as in the computation of a canonical representative, let P_w be the maximum external module partition of V_w with $P_w(1), P_w(2), \ldots, P_w(h1)$ arbitrary ordering of its classes, and v_i an arbitrary element of $P_w(i)$, for $1 \leq i \leq h1$. Given the two tables Tab_a, Tab_b we compute Tab_w as follows:

```
initialize all values of $Tab_w$ to $\infty$
for all indices $(R_a, R_{\overline{a}})$ in $Tab_a$ and $(R_b, R_{\overline{b}})$ in $Tab_b$ do:
    $R_w := can_{V_w}(R_a \cup R_b)$
    $N_a := N_{H'_a}(R_{\overline{a}}) \setminus N_{H'_a}(can_{\overline{V_a}}(R_b))$ and $N_b = N_{H'_b}(R_{\overline{b}}) \setminus N_{H'_b}(can_{\overline{V_b}}(R_a))$
    if $\nexists x \in (N_a \cup N_b)$ such that $N(x) \setminus V_w == \emptyset$ then
        $N_w = \{v_i : (N_a \cup N_b) \cap P_w(i) \neq \emptyset\}$
        (dominating $N_a \cup N_b$ in $G$ from $\overline{V_w}$ is equivalent to dominating $N_w$ in $H'_w$)
        for all $R_{\overline{w}} \in C_{\overline{w}}$, namely $R_{\overline{w}}$ a canonical representative of $\equiv_{\overline{V}_w}$, do:
            if ($N_w \subseteq N_{H'_w}(R_{\overline{w}})$) then
                $Tab_w[R_w][R_{\overline{w}}] := \min(Tab_w[R_w][R_{\overline{w}}], Tab_a[R_a][R_{\overline{a}}] + Tab_b[R_b][R_{\overline{b}}])$
```

To see why these steps are performed note that $R_{\overline{w}}$ must dominate all nodes in $G[V_w]$ that are not dominated by R_w. In $G[V_w]$ the set of nodes not dominated by R_a or R_b will be dominated by $R_{\overline{w}}$ iff

$(N_a \cup N_b)$ is dominated by $R_{\overline{w}}$. We look for $R_{\overline{w}}$ that dominate $N_a \cup N_b$ by looping over all canonical representatives in the equivalence class of $\equiv_{\overline{V_w}}$ and update the value of $Tab_w[R_w][R_{\overline{w}}]$. Those ideas are formalized in the following theorem, whose proof has been moved to the appendix.

Theorem 3.10 *The table at node w is updated correctly, namely $Tab_w[R_w][R_{\overline{w}}] \leq s \leq n$ iff $\exists S_w : |S_w| = s \wedge R_w \equiv_{V_w} S_w$ and $(S_w, R_{\overline{w}})$ dominates $G[V_w]$.*

At the end we have a table Tab_r at the root of T_r where $G[V_r] = G$. We thus find the size of the minimum dominating set of G stored in $Tab_r[can_{V_r}(V_r)][\emptyset]$. For $O(1)$ access to tables we can use a technique similar to the one used for Max Independent Set, but we leave out details.

Theorem 3.11 *Given an H-join decomposition of an n-vertex m-edge graph G we can solve the Minimum Dominating Set problem in $O(n(m+2^{\frac{5}{4}\rho(H)^2+\frac{25}{4}\rho(H)}\rho(H)^6|V(H)|))$. time, where $\rho(H)$ is the rank of the adjacency matrix of H.*

Proof First, the computation of the maximum external module partitions associated with every node of T_r takes $O(nm)$ time. Also, the tables of all leaves are initialized in $O(n)$ time.

Now, in the bottom-up process for each of the $O(n)$ other tables, we compute the list of canonical representatives in time $O(|V(H)| \times neq \times |E(H)|)$ and we initialize to ∞ (or something higher than n) in $O(neq^2)$ time where neq is the number equivalence classes. Then, we go through all the $O(neq^4)$ pairs of entries in Tab_a, Tab_b, and for each of them, we perform the following. We find R_w, N_a, and N_b in $O(|E(H)|)$ time. The size of $N_a \cup N_b$ is bounded by $2|V(H)|$ so that we compute N_w by identifying the partition class of at most $2|V(H)|$ elements, at the same time we check if some node belong to the partition class that have no neighbours. This means the first if test and the computation of N_w takes $O(|V(H)|)$ time. After this, we try all $O(neq)$ values for $R_{\overline{w}}$ and check if $N_w \subseteq N_{H'_w}(R_{\overline{w}})$ in $O(|V(H)|\rho(H))$ time since $|R_{\overline{w}}| \leq \rho(H)$, the subsequent update takes constant time. Note that $neq \geq V(H)$ hence $neq|V(H)|\rho(H) \geq |E(H)|$. In summary, the bottom-up process takes $O(neq^5|V(H)|\rho(H))$ time. Eventually, applying Proposition 3.5 allows to conclude. □

For q-Coloring we can use the same ideas as for Maximum Independent Set. We want to find q disjoint independent sets such that their union contains all vertices of the graph. For each color it suffices to keep track of all vertices that are forbidden for that color. For this, we maintain for each color of the q colors a canonical representative for the forbidden set. Then, in the table of node w of T_r, we store $true$ if all vertices of V_w is in one of the q independent sets. Therefore the table will have neq^q entries and the combining can be done in $O(|E(H)| \times q)$ time, in total this gives $O(neq^q|E(H)|q)$ time for each node of T_r. Applying Proposition 3.5 we get

Theorem 3.12 *Given an H-join decomposition of an n-vertex m-edge graph G we can solve the q-Coloring problem in $O(n(m + 2^{\frac{q}{4}\rho(H)^2+\frac{5q}{4}\rho(H)} \times \rho(H)q|E(H)|))$ time, where $\rho(H)$ is the rank of the adjacency matrix of H.*

4 Conclusion

Note that Theorem 1.1 follows from Theorems 2.7, 3.8, 3.12 and 3.11 since $|E(R_k)| \leq 2^{2k}$. Graphs of rankwidth at most k encompass large classes of graphs, e.g. all graphs of branchwidth or cliquewidth k, some graphs of unbounded branchwidth and some of cliquewidth $2^{k/2-1} - 1$. Besides, they are well-behaved in the sense of having a recognition algorithm, based on geometric properties of the graphs, that is fixed-parameter tractable when parameterized by k [15]. However, algorithms for solving NP-hard problems on graphs of bounded rankwidth had until now not been directly investigated. The alternative definition of rankwidth given in this paper, using the notion of R_k-join decompositions, is useful both for visualization purposes and for developing such algorithms. For these algorithms we used the more general notion of H-join decompositions that has connections to modular decompositions, split decompositions, perfect graph decompositions, and rank decompositions. We have shown that given an H-join decomposition of a graph G we can solve several NP-hard problems on G in FPT time when parameterized by the rank $\rho(H)$ of the adjacency matrix of H. For graphs of rankwidth k this gives algorithms having runtimes with a factor exponential in the square of k which means that they could be practical for low values of k. Finally, let us remark that the graph R_k has many interesting properties, and that graphs with a similar definition based on a parity check appear in the book of Alon and Spencer [1] and recently also in [2].

References

[1] N. Alon and J. Spencer. The probabilistic method. Second edition. Wiley-Interscience Series in Discrete Mathematics and Optimization. New York (2000). 4

[2] P. Charbit, S.Thomassé, A. Yeo. The minimum feedback arc set problem is NP-hard for tournaments. *Combinatorics, Probability and Computing*, 16 (2007), 1–4. 4

[3] M. Chudnovsky, N. Robertson, P. Seymour, and R. Thomas. The strong perfect graph theorem. *Ann. Math.*, 164: 51-229 (2006). 2

[4] D. Corneil, U. Rotics. On the relationship between cliquewidth and treewidth. *SIAM Journal of Computing* 34(4): 825-847 (2005) 1

[5] B. Courcelle, J.A. Makowsky, U. Rotics. Linear time solvable optimization problems on graphs of bounded clique width. *Theory of Computing Systems*, 33(2):125150, 2000 1, 2

[6] B. Courcelle, M.Kanté. Graph Operations Characterizing Rank-Width and Balanced Graph Expressions. Proceedings WG 2007: 66-75 1

[7] B. Courcelle, S.Oum. Vertex-minors, monadic second-order logic, and a conjecture by Seese. *J. Combinatorial Theory, Ser. B* 97(1): 91-126 (2007) 1

[8] G. Cornuéjols and W. Cunningham. Compositions for perfect graphs. *Discrete Mathematics* 55: 245-254 (1985) 2

[9] W. Cunningham and J. Edmonds. A combinatorial decomposition theory. *Canadian Journal of Mathematics* 32: 734-765 (1980) 2

[10] W. Espelage, F. Gurski, E. Wanke. How to Solve NP-hard Graph Problems on Clique-Width Bounded Graphs in Polynomial Time. Proceedings WG 2001: 117-128 1, 2

[11] T. Gallai. Transitiv orientierbare Graphen. *Acta Mathematica Academiae Scientiarum Hungaricae* 18: 25-66 (1967) 2

[12] M. Gerber, D. Kobler. Algorithms for vertex-partitioning problems on graphs with fixed clique-width. *Theoretical Computer Science* 299 (2003) 719734 1, 2

[13] M. Grohe. Logic, Graphs, and Algorithms. In J.Flum, E.Grdel, T.Wilke (Eds), Logic and Automata-History and Perspectives Amsterdam University Press, 2007. 1

[14] M. Habib, C. Paul, L. Viennot. Partition Refinement Techniques: An Interesting Algorithmic Tool Kit. *International Journal of Foundations on Computer Science*, vol 10, 2, 147–170, 1999 3

[15] P. Hliněný, S. Oum. Finding Branch-Decompositions and Rank-Decompositions. Proceedings ESA 2007: 163-174 1, 2, 4

[16] P. Hliněný, S. Oum, D. Seese, G. Gottlob. Width Parameters Beyond Tree-width and Their Applications. *Computer Journal*, to appear 2008. 1

[17] W.-L. Hsu. Decomposition of Perfect Graphs. *J. Combinatorial Theory, Ser. B* 43: 70-94 (1987) 2

[18] D. Kobler, U. Rotics. Edge dominating set and colorings on graphs with fixed clique-width. *Discrete Applied Mathematics* 126(2-3): 197-221 (2003) 1, 2

[19] S. Oum. Graphs of bounded rank-width. PhD thesis, Princeton University, 2005 1

[20] S. Oum. Rank-width is less than or equal to branch-width. *J. Graph Theory* 57(2008)(3), pp. 239-244. 1

[21] S. Oum, P. Seymour. Approximating clique-width and branch-width. *J. Combin.Theory Ser. B* 96(4):514528, 2006. 1

[22] M. Rao. MSOL partitioning problems on graphs of bounded treewidth and clique-width. *Theoretical Computer Science,* 377(1-3):260-267, 2007. 1, 2

[23] N. Robertson, P. Seymour. Graph minors X: Obstructions to tree-decomposition. *Journal on Combinatorial Theory Series B,* 52:153 190, 1991. 1

Appendix

Lemma 2.5 *The function* $\sigma_G : 2^{V(G)} \to \mathbb{N}$ *defined by*

$$\sigma_G(X) = \min\{k : G \text{ is an } R_k\text{-join of } G \text{ across the cut } \{X, V(G) \setminus X\}\}$$

is equal to the cut-rank function ρ_G.

Proof Let $k = \rho_G(X)$. There are several ways to view the graph R_k. Before proving the lemma, note the following, where we slightly abuse the notation of Definition 2.4 by denoting the vertices arising from a one-element subset $S = \{i\}$ simply as a_i and b_i. We denote by M_k the bipartite adjacency matrix of the bipartite graph R_k, meaning that its rows correspond to the vertices of one color class and the columns to those of the other color class. Suppose that the vertices $a_1, a_2, \dots, a_k$ are mapped to rows in M_k: again by abuse of notation, we can view vertex of R_k as the row/column it is mapped to in M_k. Clearly, a_S with $S = \emptyset$ is a linear combination of $a_1, a_2, \dots, a_k$: choose scalar 0 for every vector. Let a_S be a vertex of R_k with $S = i_1, i_2, \dots, i_p$. We can prove that in M_k, the row a_S is the $GF2$-sum of the rows $a_{i_1}, a_{i_2}, \dots, a_{i_p}$: for every column $b_{S'}$ of M_k, $|S \cap S'|$ is odd iff there is an odd number of the i_q $(1 \leq q \leq p)$ which belong to S', that is M_k has a 1 in the row a_{i_p} and column $b_{S'}$. The same holds for $b_1, b_2, \dots, b_k$. Note also that an arbitrary bipartite adjacency matrix is not necessarily symmetric but it is clear here that

Claim: There is a way to swap the columns and rows of M_k to result in a symmetric matrix. Also, M_k is of rank k and has the maximum size among the $GF2$-matrices of rank k.

Moreover, let us w.l.o.g. define M_k in such a way that $\{a_1, a_2, \dots, a_k\}$ are mapped (in this order) to the first k rows of M_k while $\{b_1, b_2, \dots, b_k\}$ are mapped to the k first columns. This way, the first $k \times k$ block of M_k is equal to the identity matrix of size k. We define L_k as the block of M_k made of the first k rows. Clearly, L_k has 2^k columns and has one column with only 0's.

We now come to the actual proof of the lemma. We first prove that $\sigma_G(X) \leq k$. Let M be the bipartite adjacency matrix induced by X and $V(G) \setminus X$ in G. A *valid elimination* in a matrix is a deletion of a column (resp. a row) when the matrix has another column (resp. row) identical to the one we delete. This corresponds to twin contractions in the graph defined by the matrix. Let us obtain N from M through a maximal sequence of valid eliminations. This operation corresponds to the contraction with respect to some external module partition. Then, in order to prove that G is an R_k-join across $\{X, V(G) \setminus X\}$, it suffices to prove that the bipartite graph G_N with bipartite adjacency matrix N is an induced subgraph of R_k. This will be proved in two steps.

There can not be less than k rows in N. If the number of rows in N is exactly k, then we look at N as a collection of columns. By maximality of the sequence of valid eliminations, all the latter columns are pairwise distinct. Besides, if we look at L_k as a collection of columns, then by definition L_k contains all possible k-bit vectors. Therefore, N (as a collection of columns) is a subset of L_k. Hence, G_N is an induced subgraph of the bipartite graph defined by L_k, and consequently it is an induced subgraph of R_k. If the number of columns in N is exactly k, then by transposition we can conduct a similar argument to conclude.

Otherwise we take k rows of N which induce a k-basis of the matrix N. Putting those k rows together results in a matrix Z of k rows. Besides, the other rows of N are linear combinations of those k rows. Therefore, the columns of Z are pairwise distinct otherwise there would be identical columns in N, which contradicts the maximality of the sequence of valid eliminations. Then, the previous argument applies, and every column of Z is a column of L_k: w.l.o.g. suppose Z is a block of L_k (otherwise swap columns). Let T be a set of rows which contains all linear combinations of rows of Z. Now, the set of rows of M_k contains every linear combination of rows of L_k, and Z is a block of L_k. Consequently, we can suppose w.l.o.g. that T is a block of M_k (otherwise just swap rows). Then, the bipartite graph G_T defined by T is an induced subgraph of R_k. Besides, it is clear that every row of N belongs to T and G_N is an induced subgraph of G_T. Hence, G_N is an induced subgraph of R_k.

We now prove that $\rho_G(X) \leq \sigma_G(X)$. Let $l = \sigma_G(X)$. We know there exists external module partitions P and Q of X and $V(G) \setminus X$ such that G is an R_l-join across $\{X, V(G) \setminus X\}$. Let Y and Z contain one representative vertex per part in respectively P and Q. Then, the cut-rank value $\rho_G(X)$ is equal to the rank of the bipartite adjacency matrix M between Y and Z. Clearly, the graph defined by M is an induced subgraph of R_l from Proposition 3.2. Hence, the cut-rank value $\rho_G(X)$ can not exceed that of R_l, which is equal to l. $\square$

Lemma 3.1 *Let G be a graph and S be a vertex subset of $V(G)$. The maximum (coarse-wise) external module partition of S is well-defined and can be computed in $O(|E(G)|)$ time.*

Proof Let P be a maximal external module partition of S. Suppose it is not maximum, then there exists an external module partition Q of S such that there are some parts $X \in P$ and $Y \in Q$ such that X and Y overlap. Then, replace all X_i in P which overlap (or included in) Y by $\bigcup_i X_i \cup Y$, and obtain P'. Using the transitivity of the relation on x, y for a given z: "x and y are linked to z the same way" , we can prove that P' is an external module partition that is coarser than P. Contradiction.

To achieve the proof, it suffices to prove that the above description computes correctly a maximum external module partition. That the computation results in an external module partition is straightforward from an argument by contradiction. Moreover, the partition is maximum since, for every external module partition of S, for every exterior vertex $z \in V(G) \setminus S$, the neighbourhood of z does not overlap any part in the external module partition. $\square$

Proposition 3.2 *Let (T, δ) be an H-join decomposition of G. Let P_u, P_v be the maximum external module partitions of respectively S_u and S_v, where $\{S_u, S_v\}$ is the 2-partition of $V(G)$ we get by deleting an edge uv in T. Let R_u and R_v be two sets containing exactly one vertex per part in respectively P_u and P_v and let H' be the bipartite graph defined by the bipartite adjacency in G between R_u and R_v. Then H' is an induced subgraph of H, and therefore G is an H-join across the cut $\{S_u, S_v\}$ using partitions P_u and P_v.*

Proof Straight from Definition 2.1 we have that G is an H'-join across $\{S_u, S_v\}$ using P_u and P_v. Besides, since (T, δ) is an H-join decomposition of G, G is also an H-join across $\{S_u, S_v\}$. This latter H-join uses some partitions of S_u and S_v, say Q_u and Q_v. Let F be the subgraph of H which is induced by the image of Q_u and Q_v by the injections f_1 and f_2 as defined in Definition 2.1. Clearly G is an F-join across $\{S_u, S_v\}$ using partitions Q_u and Q_v and F is an induced subgraph of H'. It is straightforward to check that Q_u and Q_v are external module partitions. ¿From Lemma 3.1, P_u (resp. P_v) is coarser than Q_u (resp. Q_v). We deduce that H' is an induced subgraph of F, and thus also of H, which is obtained by some successive twin contractions of F. ¿From the fact that G is an H'-join across $\{S_u, S_v\}$ using P_u and P_v, for an induced subgraph H' of H, it follows by Definition 2.1 that G is an H-join across $\{S_u, S_v\}$ using P_u and P_v. $\square$

Proposition 3.5 *For any vertex subset $X \subseteq V_a$, the number of vertices belonging to the canonical representative $R = can_{V_a}(X)$ for the class $[X]_{\equiv_{V_a}}$ is at most $\rho(H)$, the rank of the bipartite adjacency matrix of H. Moreover, the number neq of equivalence classes of $\equiv_{V_a}$ is at most $neq \leq 2^{\frac{1}{4}\rho(H)^2+\frac{5}{4}\rho(H)}\rho(H)$.*

Proof Consider the bipartite adjacency matrix of the cut $\{V_a, V(G) \setminus V_a\}$ of G: it defines a vector space $\mathcal{S}$ of dimension $\rho(H)$. We first prove that the vertices belonging to R, when considered as vectors of $\mathcal{S}$, are linearly independent and therefore they span a subbasis of $\mathcal{S}$ which in particular implies their number is at most $\rho(H)$. For this just notice that, during the process of the algorithm to compute $R = can_{V_a}(X)$, whenever a vertex v_i is added to $can_{V_a}(X)$, it has to pass the check where $N_{H'}(v_i) \setminus W \neq \emptyset$. This check ensures that v_i cannot be obtained as a linear combination of the vertices already present in the current $can_{V_a}(X)$. Therefore, at the end of the process, all vertices belonging to R are linearly independent. Hence, R is a subbasis of $\mathcal{S}$, and its cardinality is at most $\rho(H)$.

We now prove the claim that

$$neq \leq \sum_{i=1}^{\rho(H)} \binom{\rho(H)}{i}_2, \text{ where } \binom{n}{m}_q = \prod_{i=1}^{m} \frac{1-q^{n-i+1}}{1-q^i}.$$

For this we use the folklore fact that $\binom{n}{m}_q$, which is known under the name of the q-binomial coefficient of n and m, is exactly the number of different subspaces of dimension m of a given space of dimension n over a finite field of q elements (roughly, $\frac{1-q^{n-i+1}}{1-q^i}$ is the number of choices of an i^{th} vector that is linearly independent from the previously chosen ones). In other words $\binom{n}{m}_q$ is the number of subbasis which span pairwise distinct subspaces of dimension m in a given space of dimension n over $GF(q)$. Now, another consequence of the above proven fact on the canonical representative R is that every equivalence class of $\equiv_{V_a}$ contains at least one subbasis of $\mathcal{S}$ of dimension at most the dimension of $\mathcal{S}$. There are $\sum_{i=1}^{\rho(H)} \binom{\rho(H)}{i}_2$ choices for such a subbasis. Hence that many choices for an equivalence class of $\equiv_{V_a}$, which proves the claim.

Let $neq = a(\rho(H))$. In order to conclude we can use the $q-$analog of Pascal triangles: $\binom{n}{m}_q = 2^m\binom{n-1}{m}_q + \binom{n-1}{m-1}_q$, for all $m \leq n$, with the convention that $\binom{n}{m}_q = 0$ if $m < 0$ or $m > n$. ¿From this we firstly have that the highest number among $\binom{n}{m}_q$, for all $0 \leq m \leq n$, is when $m = \lceil \frac{n}{2} \rceil$. Therefore, $a(n) \leq n \times b(n)$ with $b(n) = \binom{n}{\lceil \frac{n}{2} \rceil}_q$. Finally, still using the q-analog of Pascal triangles, one can check that $b(n) \leq \left(2^{\lceil \frac{n}{2} \rceil} + 1\right) \times b(n-1) \leq 2^{\frac{1}{4}n^2 + \frac{5}{4}n}$. □

Theorem 3.10 *The table at node w is updated correctly, namely $Tab_w[R_w][R_{\overline{w}}] \leq s \leq n$ iff $\exists S_w$: $|S_w| = s \wedge R_w \equiv_{V_w} S_w$ and $(S_w, R_{\overline{w}})$ dominates $G[V_w]$.*

Proof Suppose w is a leaf of T_r. Let x be the corresponding vertex in G. Then $V_w = \{x\}$, therefore there are only two possibilities, either x is part of the dominating set, or it needs to be dominated by some other vertex, if neither happens we give the value ∞ to the entry. Suppose now that w is an internal node of T_r with children a and b. Assuming Tab_a and Tab_b are correctly filled, we show that Tab_w is correctly filled.

($\Rightarrow$) Each value in Tab_w is a sum of one value from Tab_a and one value from Tab_b. We therefore know there exist S_a and $R_{\overline{a}}$ such that $(S_a, R_{\overline{a}})$ dominates G_a and S_b and $R_{\overline{b}}$ such that $(S_b, R_{\overline{b}})$ dominates G_b such that $S_a \cup S_b = S_w$ and $|S_w| = s$ is stored in Tab_w. Now we need to show $(S_w, R_{\overline{w}})$ dominates $G[V_w]$:
For any node x in V_a we show that x is dominated. We know x is dominated by $(S_a, R_{\overline{a}})$. There are two types of nodes in V_a, those dominated by S_a and those not dominated by S_a. Those dominated by S_a will also be dominated by S_w , the rest are in $N(R_{\overline{a}})$. If x is also in $N(S_b)$ it is dominated by S_w else it will be in N_a, and hence a node v_i with identical neighbourhood will be in N_w. So we have $x \in N(R_{\overline{w}}) \Leftrightarrow v_i \in N(R_{\overline{w}})$. Since $N_w \subseteq N(R_{\overline{w}})$, x is dominated. By a symmetric argument all nodes in V_b are also dominated, hence $(S_w, R_{\overline{w}})$ dominates $G[V_w]$.

($\Leftarrow$) Assume $(S_w, R_{\overline{w}})$ dominates $G[V_w]$ with $R_w = can_{V_w}(S_w)$ and $|S_w| = s$. We show how $Tab_w[R_w][R_{\overline{w}}]$ was updated when parsing some entries R_a and R_b in the tables Tab_a and Tab_b. We first need to properly define R_a and R_b. Let $S_a = S_w \cap V_a, S_b = S_w \cap V_b$ and $R_a = can_{V_a}(S_a)$ and $R_b = can_{V_b}(S_b)$. Let $R_{\overline{a}} = can_{\overline{V_a}}(R_{\overline{w}} \cup R_b)$ and $R_{\overline{b}} = can_{\overline{V_b}}(R_{\overline{w}} \cup R_a)$. Clearly, $S_w = S_a \cup S_b$ and therefore $R_w = can_{V_w}(R_a \cup R_b)$. By expressing the definition of $R_{\overline{a}} \equiv_{\overline{V_a}} R_{\overline{w}} \cup R_b$ and $R_{\overline{b}} \equiv_{\overline{V_b}} R_{\overline{w}} \cup R_a$, we obtain both

$$\begin{cases} N_a = (N(R_{\overline{a}}) \setminus N(R_b)) \setminus \overline{V_a} = (N(R_{\overline{w}} \cup R_b) \setminus N(R_b)) \setminus \overline{V_a} \subseteq N(R_{\overline{w}}) \\ N_b = (N(R_{\overline{b}}) \setminus N(R_a)) \setminus \overline{V_b} = (N(R_{\overline{w}} \cup R_a) \setminus N(R_a)) \setminus \overline{V_b} \subseteq N(R_{\overline{w}}) \end{cases}$$

Therefore, $N_a \cup N_b \subseteq N(R_{\overline{w}})$.
What remains to prove is that $(S_a, R_{\overline{a}})$ dominates G_a and $(S_b, R_{\overline{b}})$ dominates G_b. Let x be a node of V_a, then we show that x is dominated by $(S_a, R_{\overline{a}})$. There are three ways for x to be dominated in $G[V_w]$, by S_a, by S_b or $x \in N(R_{\overline{w}})$. If x was dominated by S_a it still is dominated. Since $N(R_{\overline{w}} \cup R_b) \setminus \overline{V_a} = N(R_{\overline{a}}) \setminus \overline{V_a}$ then x will be in $N(R_{\overline{a}})$ hence $(S_a, R_{\overline{a}})$ dominates G_a. A symmetric argument holds for V_b, therefore $(S_b, R_{\overline{b}})$ dominates G_b. □