H-join decomposable graphs and algorithms with runtime single exponential in rankwidth

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

[a] Department of Informatics, University of Bergen, Norway.
[buixuan,telle,martinv]@ii.uib.no

Abstract

We introduce H-join decompositions of graphs, indexed by a fixed bipartite graph H. These decompositions are based on a graph operation that we call H-join, which 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. H-join decompositions are related to modular, split and rank decompositions.

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 $O(n(m+2^{O(q\rho(H)^2)}))$, where $\rho(H)$ is the rank of the adjacency matrix of H over GF(2).

Rankwidth is a graph parameter introduced by Oum and Seymour, based on ranks of adjacency matrices over GF(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 graphs having an R_k-join decomposition, thereby giving an alternative graph-theoretic definition of rankwidth that does not use linear algebra.

Combining our results we get algorithms that, for a graph G of rankwidth k given with its width k rank-decomposition, solves 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 $O(n(m+2^{\frac{3}{4}k^2+\frac{23}{4}k}\times k^3))$ and the q-Coloring problem in time $O(n(m+2^{\frac{q}{2}k^2+\frac{5q+4}{2}k}\times k^{2q}\times q))$. These are the first algorithms for NP-hard problems whose runtimes are single exponential in the rankwidth[1].

1 Introduction

A key tool in the area of graph algorithms is the concept of decomposing a graph into a tree structure. Many variants have been studied, like modular, split and rank decompositions. In this paper we introduce H-join decompositions, another tree-like decomposition of graphs based on a graph operation that we call H-join. The H-join operation is indexed by a fixed bipartite graph H and 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 and a discussion of relations to some other graph decompositions are given in Section 2. For this to be a good algorithmic tool we should choose a graph H that satisfies the following desiderata list:

†Part of this work done while the first author was a PhD student at LIRMM, Univ. Montpellier II, CNRS, and supported by the French National Research Agency, project GRAAL.

‡Supported by the Norwegian Research Council, project PARALGO.

[1] For a polynomial function $poly$ we call $2^{poly(k)}$ single exponential in k.

- given an H-join decomposition of a graph G several important NP-hard problems should be solvable fast on G
- there should be a relatively fast algorithm that finds an H-join decomposition of an input graph G, if it exists
- interesting classes of graphs should have H-join decompositions, alternatively we could use a family of graphs $H_1, H_2, H_3, ...$ such that every G is H_i-decomposable for some i

In Section 3 we address the first item and show that regardless of which graph H is chosen we can solve several NP-hard optimization problems on G by dynamic programming along an H-join decomposition of G. We will show this for the problems of computing a Maximum Independent Set, Maximum Clique, Minimum Dominating Set and Vertex q-Coloring. The runtime of these algorithms will depend single exponentially on $\rho(H)$, the rank of the adjacency matrix of H over GF(2).

In Section 4 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 graphs having an R_k-join decomposition. Combining our algorithms from Section 2 with the powerful results that hold for rankwidth [18, 5] this means that all three items on the desiderata list are satisfied for the family $R_1, R_2, R_3, \ldots$.

Let us say a few words about rankwidth. 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 [27], it has bounded cliquewidth iff it has bounded rankwidth [24], 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 [23]. In this sense rankwidth, which has been investigated quite heavily in recent years [6, 7, 18, 22, 23] 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 the recent paper by Hliněný et al [19] for an overview. As reflected by the two first items in our desiderata list, these FPT algorithms usually have two 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 by Oum and Seymour [24] as a tool to help compute a decomposition for cliquewidth.

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 $O(f(k)n^3)$ [18]. Between rankwidth $rw(G)$ and cliquewidth $cw(G)$ we have the connection $rw(G) \leq cw(G) \leq 2^{rw(G)+1}$ [24]. 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, if we want algorithms with runtime single exponential in rankwidth, we cannot go via a cliquewidth decomposition. Concerning FPT algorithms for problems parameterized by cliquewidth or rankwidth we have a recent negative result by Fomin et al [12] showing that various graph problems are W[1]-hard when parameterized by cliquewidth, thus also when parameterized by rankwidth, and hence unlikely to have FPT algorithms at all. The main positive result is by Courcelle, Makowsky and Rotics [5] who have shown that any MSO_1-logic problem is FPT when parameterized by cliquewidth. Courcelle and Kanté [6] gave an alternative algebraic characterization of graphs of bounded rankwidth, based on vertex colors that are manipulated by linear transformations over the GF[2] vector space, that will allow a result like the one in [5] for graphs of bounded rankwidth without transforming into a cliquewidth expression. Ganian and Hliněný [14] give another alternative characterization of rankwidth by using labelling parse trees and an automata-approach in order to explicitly solve all MSO_1 problems directly on these parse trees. However, these results that hold for all MSO_1 problems do not have practical runtime, as the exponential dependency on the parameter is a tower of powers depending on the logical expression. For practical runtime a more refined analysis is necessary. Finding FPT algorithms with low dependency on the parameter is a main goal of research in parameterized algorithms, see e.g. Downey and Fellows [10]. Applying the algorithms developed in Section 3 to the family $R_1, R_2, R_3, ...$ will give the first algorithms for NP-hard problems that are single exponential in rankwidth.

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 Maximum Clique problem in time $O(n(m + 2^{\frac{1}{2}k^2+\frac{11}{2}k} \times k^2))$, the Minimum Dominating Set problem in time $O(n(m + 2^{\frac{3}{4}k^2+\frac{23}{4}k} \times k^3))$, and q-coloring in time $O(n(m + 2^{\frac{q}{2}k^2+\frac{5q+4}{2}k} \times k^{2q} \times q))$.*

Let us mention that using the connection between the family $R_1, R_2, R_3, ...$ and rankwidth it easily follows that any MSO_1 problem can be solved in FPT time for H-join decomposable graphs when parameterized by the rank of the adjacency matrix of H over GF(2).

2 H-join decomposable graphs

In this section we introduce H-join decompositions and discuss its relations to other well-known graph decompositions. However, the main result showing the tight connection to rank decompositions is postponed to Section 4.

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 $f_1 : P \to V_1$ and $f_2 : Q \to V_2$, 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, unless otherwise explicited, 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. With some abuse in terminology, 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 modular decomposition [13]. 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 will also be reflected in the upcoming section via the notion of an external module partition, our main tool for the study of the partitions used for an H-join across a cut (cf. Definition 3.1 and Proposition 3.3).

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 [20]. 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.

Finally, note that in the above list of connections between H-joining (for some H) and particular vertex partitions, namely modules, splits, 2-join and Hsu's generalized join, the two first cases, namely when $H = P_2^+$ and $H = P_2^{++}$, are known to own polynomially computable decomposition trees. Then, it is clear that one can exploit this fact to compute the corresponding H-join decomposition of a given H-join decomposable graph G.

3 Dynamic programming on H-join decomposable graphs

Our goal is to 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. The main idea is to use a technique called vertex splitting, introduced by Paige and Tarjan as the act of splitting parts according to the neighborhood of a vertex [25].

Definition 3.1 *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.*

Given as input a graph G and a vertex subset $S \subseteq V(G)$, by using partition refinement techniques we can compute a maximum external module partition of S, which is well-defined by Lemma 3.2 (below). Indeed, 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 (refer to [17] for more details in efficient implementations of partition refinement). The correctness of the computation is stated in the following lemma.

Lemma 3.2 *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, and let Q be an external module partition of S that is not comparable (coarse-wise) to P. If no parts among P and Q overlap (where X and Y overlap if they have a non-empty intersection and two non-empty differences), Then, replacing the parts of P that are included in some part Y of Q by Y will lead to an external module partition of S which is coarser than P. Contradiction. Hence, we deduce 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. Finally, the fact the computed partition is maximum can be proved by a straightforward argument by contradiction. □

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

Proposition 3.3 *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 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. Note that F is not necessarily twin-free. Clearly G is an F-join across $\{S_u, S_v\}$

using partitions Q_u and Q_v. It is straightforward to check that Q_u and Q_v are external module partitions. From Lemma 3.2, P_u (resp. P_v) is coarser than Q_u (resp. Q_v). We deduce that H' is an induced subgraph of F which is obtained by some successive twin contractions of F. Therefore, H' is an induced subgraph of H. Finally, 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. □

3.1 Equivalence classes used for independent set, dominating set and q-coloring

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 a of T_r we associate a data structure *table*, that will store optimal solutions to subproblems restricted to the graph $G[V_a]$, where V_a are the vertices of G mapped to leaves of the subtree of T_r rooted at a. Each index of the table will be associated with an equivalence class of subproblems. For the problems studied in this paper, Maximum Independent Set, Minimum Dominating Set and Vertex q-Coloring, these classes of subproblems will be related to the following equivalence classes of vertex subsets.

Definition 3.4 *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 the node a of T_r we will be interested in the equivalence relation $\equiv_{V_a}$ on the above defined subset V_a. Note that we have explained how to compute P_a and Q_a such 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 an arbitrary ordering $P_a(1), P_a(2), ..., P_a(h1)$ of the classes of P_a. For all $1 \leq i \leq h1$, let v_i be an arbitrary element of $P_a(i)$.

Definition 3.5 (Representative) *Given an arbitrary ordering $P_a(1), P_a(2), ..., P_a(h1)$ of the classes of P_a, and an arbitrary element v_i of $P_a(i)$, for $1 \leq i \leq h1$, we define the* canonical representative *$can_{V_a}(X)$ of a vertex set $X \subseteq V_a \subseteq V(G)$ as the lexicographically smallest subset taken over every $R \subseteq \{v_1, v_2, \ldots, v_{h1}\}$ satisfying:*

- $R \equiv_{V_a} X$
- *For any $v_i \in R$ we have $N(v_i) \setminus \left(\bigcup_{j<i,\ v_j \in R} N(v_j) \cup V_a\right) \neq \emptyset$.*

The definition is well-defined, in the sense that $R_0 = \{v_i,\ X \cap P_a(i) \neq \emptyset\}$ is a subset which satisfy the first item in the definition. Then, a subset satisfying both items can be defined from R_0 by a greedy scan on the v_i's (this is basically the computation that will be formally proved in Lemma 3.7). Besides, the above definition also leads to a canonical representative for every equivalence class of $\equiv_{V_a}$, namely we have that

$$X \equiv_{V_a} X' \quad \Rightarrow \quad can_{V_a}(X) = can_{V_a}(X').$$

Finally, the following bounds are crucial for an efficient complexity analysis of all algorithms presented in this paper:

Proposition 3.6 *For any vertex subset $X \subseteq V_a \subseteq V(G)$, we have that $|can_{V_a}(X)| \le \rho(H)$, the rank of the bipartite adjacency matrix of H. Moreover, for the number neq of equivalence classes of $\equiv_{V_a}$ we have $neq \le 2^{\frac{1}{4}\rho(H)^2+\frac{5}{4}\rho(H)}\rho(H)$.*

Proof Let $M(V_a)$ be the bipartite adjacency matrix associated to the cut $\{V_a, V(G)\setminus V_a\}$ of G. For convenience, let $R = can_{V_a}(X)$.

For the first claim of the proposition, we only need to prove that the rows in $M(V_a)$ which correspond to the vertices of R are $GF(2)$-independent (hence form a subbasis of $M(V_a)$). This can be proven by induction on $|R|$. Indeed, if $|R| = 1$ then it is clear how to conclude. Suppose that the property is true for every canonical representative with cardinality upto $p-1 \ge 1$, and let us consider a canonical representative R with cardinality p. Let $x \in R$ be the highest element belonging to R (w.r.t. the order $v_1, v_2, \ldots, v_{h1}$). Notice from the definition of R that $R \setminus \{x\}$ is not in the same equivalence class as the one R and X belong to. Moreover, it also follows directly from definition that $R \setminus \{x\}$ is a canonical representative (of some other set than X): otherwise R would not be a canonical representative of X. Applying the inductive hypothesis, we obtain that the rows corresponding to $R \setminus \{x\}$ in $M(V_a)$ are $GF(2)$-independent. Finally, from the maximality of x and the definition of R we have that $N(x) \setminus \left(\bigcup_{v\in R\setminus\{x\}} N(v) \cup V_a\right) \neq \emptyset$. Combining the previous facts, we obtain the desired property on R.

Now, what we just proved also implies that every equivalence class of $\equiv_{V_a}$ can be associated with (at least) one space that is $GF(2)$-spanned by some rows of $M(V_a)$. In other words, the number of spaces spanned by a subset of rows of $M(V_a)$ is larger than the value of neq. This will be used to prove the following bound on neq:

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

Indeed, this bound is just a combination of the previous fact and 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).

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 $m \le n$, 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 $0 \le m \le n$, is when $m = \lceil \frac{n}{2} \rceil$. Therefore, $a(n) \le n \times b(n)$ with $b(n) = \binom{n}{\lceil \frac{n}{2} \rceil}_q$. Finally, still using the q-analog of Pascal triangles, one can check that $b(n) \le \left(2^{\lceil \frac{n}{2} \rceil} + 1\right) \times b(n-1) \le 2^{\frac{1}{4}n^2+\frac{5}{4}n}$. □

We now show a straightforward computation of the canonical representative $can_{V_a}(X)$ of a subset $X \subseteq V_a$. Recall that v_i was an arbitrary element of $P_a(i)$, for $1 \le i \le h1$. Let $Q_a(1), Q_a(2), \ldots, Q_a(h2)$ be an arbitrary ordering of the classes of Q_a, and let u_i be an arbitrary element of $Q_a(i)$, for $1 \le i \le h2$. Let H' be the bipartite graph induced by edges (of G) between the two vertex sets $\{v_1, v_2, ..., v_{h1}\}$ and $\{u_1, u_2, ..., u_{h2}\}$. Note from Proposition 3.3 that we have H' isomorphic to an induced subgraph of H, and in particular $|E(H')| \le |E(H)|$ and $\rho(H') \le \rho(H)$. For more clarity, we denote the neighbors of v_i in H' by $N_{H'}(v_i)$, while we denote the neighbors of v_i in G simply by $N(v_i)$. Besides, we will also denote the neighborhood of a vertex subset by $N(X) = \bigcup_{x \in X} N(x) \setminus X$. The algorithm to compute the canonical representative of X can be:

compute the set $X_{H'} = \{v_i : X \cap P_a(i) \neq \emptyset\}$.
initialize $can_{V_a}(X)$ and W to the emptyset.
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 in the above we could have broken out of the for loop as soon as $W = N_{H'}(X_{H'})$.

Lemma 3.7 *For $X \subseteq V_a \subseteq V(G)$ the above algorithm computes correctly the canonical representative $can_{V_a}(X)$ of X and runs in $O(|X| + |E(H)|)$ time.*

Proof Let R be the output of the algorithm. If $R \equiv_{V_a} X$ then it is straightforward to check that R fulfills the other requirements in Definition 3.5. Therefore, we only give the proof of $R \equiv_{V_a} X$. Firstly, it is clear that $X \equiv_{V_a} X_{H'}$. Now, the only trick in the algorithm is to restrict the visitted edges to those of H'. It means in particular that $N_{H'}(X_{H'}) = N_{H'}(R)$ at the end of the algorithm (and *apriori* we cannot guarantee anything about the neighborhood of R in G). However, Q_a is an external module partition of $V(G) \setminus V_a$. This can then be exploited and leads to $X_{H'} \equiv_{V_a} R$. Hence, $R \equiv_{V_a} X$.

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. Note that we here require the adjacency list of H'. This task can be included in the pre-computation explained in Lemma 3.2 by some straightforward modifications. Finally, $|E(H')| \leq |E(H)|$ from Proposition 3.3. □

We present a last tool that will be used afterwards. Basically, we do not want in our algorithms to parse the subsets of V_a in order to look for some equivalence class of $\equiv_{V_a}$. Fortunately, the previous definition of canonical representatives comes in handy since there is a fast and simple manner to output the list C_a containing all canonical representatives:

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 3.8 *R belongs to C_a if and only if R is a canonical representative of $\equiv_{V_a}$. Moreover, C_a can be output in $O(h1 * neq * |E(H)|)$ time.*

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 3.7 since the calls to $can_{V_a}(X)$ always satisfy $|X| = O(|E(H)|)$. $\square$

3.2 Maximum Independent Set and Maximum Clique

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 3.6 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 3.9 *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. $\square$

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 $O(neq^2|E(H)|)$. This gives a total runtime of $O(n(m + neq^2|E(H)|))$. Using the bound on neq from Proposition 3.6 we thus get:

Theorem 3.10 *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 $\overline{G}$. Therefore, the Maximum Clique problem can be solved using the "Independent Set" algorithm with a runtime bounded by $O(n(m + 2^{\frac{1}{2}\rho(H)^2+\frac{7}{2}\rho(H)}\rho(H)^2|E(\overline{H})|))$, since $\rho(\overline{H}) \leq \rho(H) + 1$.

3.3 Vertex q-Coloring

For q-Coloring a straightforward generalization of the ideas used for Maximum Independent Set works. We now ask if there exists a partition of the vertex set into q color classes each forming an independent set. The table at a node w will be indexed by q representatives, one for each color class, and the contents of $Tab_w[R_1][R_2]...[R_q]$ should be True if there exists a partition $(S_1, ..., S_q)$ of V_w with each S_i inducing an indpendent set and $S_i \equiv_{V_w} R_i$. There will thus be neq^q indices in each table. For the combining of two tables we loop over all pairs of indices having the value True, check for each of the q color classes whether the union of the two representatives are an independent set, and if so update the table at the parent, in time $O(|E(H)| \times q)$. Applying Proposition 3.6 we get

Theorem 3.11 *Given an H-join decomposition of an n-vertex m-edge graph G we can solve the q-Coloring problem in $O(n(m + 2^{\frac{q}{2}\rho(H)^2+\frac{5q}{2}\rho(H)}\rho(H)^{2q}q|E(H)|))$ time.*

3.4 Minimum Dominating Set

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 need to include 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'. The representatives of these equivalence classes are computed as described in subsection 3.1, for both $\equiv_{V_w}$ and for $\equiv_{\overline{V_w}}$. 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 loop over all possibilities of sets that can dominate 'from the outside'.

Definition 3.12 *Let $G = (V, E)$ be a graph, for $V_w, X \subseteq V$ we say that X dominates V_w if V_w is a subset of $X \cup N(X)$.*

Note that if X dominates V_w then $X \cap V_w$ are the 'inside' dominators and $X \setminus V_w$ are the 'outside' dominators. 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:

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

Since we are dealing with a minimization problem we first set all entries of all tables to ∞. At a leaf w of T_r corresponding to a vertex x of G, there are at most four entries in Tab_w. Let $R = can_{\overline{V_w}}(\overline{V_w})$. We then set $Tab_w[\{x\}][R] = 1$, we set $Tab_w[\{x\}][\emptyset] = 1$, and if $x \in N(R)$ then we set $Tab_w[\emptyset][R] = 0$. The rest of the entries stay equal to ∞.

At an internal node w we only proceed when both children already have been processed. Let a and b be two nodes of T_r with w their common parent. As described by Theorem 3.8 we have computed all lists of representatives, and in particular we have $C_a = \{can_{V_a}(X) : X \subseteq V_a\}$, $C_b = \{can_{V_b}(X) : X \subseteq V_b\}$, and $C_{\overline{w}} = \{can_{\overline{V_w}}(X) : X \subseteq \overline{V_w}\}$. Given the two tables Tab_a, Tab_b we compute Tab_w as follows:

initialize all values of Tab_w to ∞
for all indices $R_a \in C_a, R_b \in C_b$ and $R_{\overline{w}} \in C_{\overline{w}}$ do:
    $R_w := can_{V_w}(R_a \cup R_b)$
    $R_{\overline{a}} := can_{\overline{V_a}}(R_b \cup R_{\overline{w}})$
    $R_{\overline{b}} := can_{\overline{V_b}}(R_a \cup R_{\overline{w}})$
    $Tab_w[R_w][R_{\overline{w}}] := \min(Tab_w[R_w][R_{\overline{w}}], Tab_a[R_a][R_{\overline{a}}] + Tab_b[R_b][R_{\overline{b}}])$

Theorem 3.13 *The table at node w is updated correctly, namely*

$$Tab_w[R_w][R_{\overline{w}}] \leq s \Leftrightarrow \exists S_w : |S_w| \leq s \wedge R_w \equiv_{V_w} S_w \textit{ and } S_w \cup R_{\overline{w}} \textit{ dominates } V_w$$

Proof We prove this inductively bottom-up in the tree of the H-join decomposition, namely we assume that Tab_a and Tab_b are correct.

($\Rightarrow$) We have that $Tab_w[R_w][R_{\overline{w}}] \leq s$. This means that an update happened in the algorithm, hence there must exist R_a and R_b such that: $can_{V_w}(R_a \cup R_b) = R_w$, $R_{\overline{a}} := can_{V_{\overline{a}}}(R_b \cup R_{\overline{w}})$, $R_{\overline{b}} := can_{V_{\overline{b}}}(R_a \cup R_{\overline{w}})$ and $Tab_a[R_a][R_{\overline{a}}] + Tab_b[R_b][R_{\overline{b}}] \leq s$. Then by induction there exist $S_a \equiv_{V_a} R_a$ and $S_b \equiv_{V_b} R_b$ and $S_a \cup S_b = S_w \equiv_{V_w} R_w$, and also $|S_w| = |S_a| + |S_b| \leq s$. It remains to show that $S_w \cup R_{\overline{w}}$ dominates V_w or equivalently that $S_a \cup S_b \cup R_{\overline{w}}$ dominates V_w. We do this in two steps, first we show that V_a is dominated, then we show that V_b is dominated. We know that $S_a \cup R_{\overline{a}}$ dominates V_a, now since $R_{\overline{a}} \equiv_{\overline{V_a}} S_b \cup R_{\overline{w}}$ we have that $S_a \cup S_b \cup R_{\overline{w}}$ dominates V_a. Similary we know that $S_b \cup R_{\overline{b}}$ dominates V_b, now since $R_{\overline{b}} \equiv_{\overline{V_b}} S_a \cup R_{\overline{w}}$ we have that $S_b \cup S_a \cup R_{\overline{w}}$ dominates V_b and we are done with this direction of the proof.

($\Leftarrow$) In this case we know $\exists S_w : |S_w| \leq s \wedge R_w \equiv_{V_w} S_w$ and $S_w \cup R_{\overline{w}}$ dominates V_w. Let $S_a = S_w \setminus V_b$, $S_b = S_w \setminus V_a$, $R_a = can_{V_a}(S_a)$ and $R_b = can_{V_b}(S_b)$. Since the algorithm goes through all triples it will go through $R_a, R_b, R_{\overline{w}}$. Let $R_{\overline{a}} = can_{V_a}(R_{\overline{w}} \cup R_b)$ and $R_{\overline{b}} = can_{V_b}(R_{\overline{w}} \cup R_a)$. Since $S_w = S_a \cup S_b$ we get $S_a \cup R_{\overline{a}}$ dominates V_a and $S_b \cup R_{\overline{b}}$ dominates V_b. By induction $Tab_a[R_a][R_{\overline{a}}] \leq |S_a|$ and $Tab_b[R_b][R_{\overline{b}}] \leq |S_b|$. Hence $Tab_w[R_w][R_{\overline{w}}] \leq |S_a| + |S_b| = |S_w|$. $\square$

At the end we have a table Tab_r at the root of T_r where $V_r = V$. We thus find the size of the minimum dominating set of G stored in $Tab_r[\emptyset][\emptyset]$. For accessing the tables we use the same technique as for Max Independent Set. By applying Proposition 3.6 we get the following runtime:

Theorem 3.14 *Given an H-join decomposition of an n-vertex m-edge graph G we can solve the Minimum Dominating set problem in $O(n(m + 2^{\frac{3}{4}\rho(H)^2+\frac{15}{4}\rho(H)}\rho(H)^3|E(H)|))$ time.*

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 for each of the three subsets in time $O(|V(H)|2^{\frac{1}{4}\rho(H)^2+\frac{5}{4}\rho(H)}\rho(H)|E(H)|)$. Then, we go through all triples of representatives, namely $O(2^{\frac{3}{4}\rho(H)^2+\frac{15}{4}\rho(H)}\rho(H)^3)$ triples, and for each of them, we find R_w, $R_{\overline{a}}$, and $N_{\overline{b}}$ in $O(|E(H)|)$ time. The subsequent update takes constant time. In summary, the bottom-up process takes $O(n(m + 2^{\frac{3}{4}\rho(H)^2+\frac{15}{4}\rho(H)}\rho(H)^3|E(H)|))$ time. $\square$

4 Rankwidth

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 4.1 *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 4.2 *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 4.1 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 $\sigma_G(X) \le k$. 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 $\rho_G(X) \le \sigma_G(X)$. 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 3.3. Hence, the cut-rank value $\rho_G(X)$ can not exceed that of R_l, which is equal to l. □

Theorem 4.3 *(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.*

Theorem 4.3 follows directly from Lemma 4.2. The following straightforward observation shows how, on the other hand, H-join decompositions can be embedded in a rank decomposition of reasonable width.

Theorem 4.4 *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 $\rho(H)$.*

Note that Theorem 1.1 follows from Theorems 4.4, 3.10, 3.11 and 3.14 since $|E(R_k)| \leq 2^{2k}$. The above observation, though simple, implies that H-join decompositions inherit algorithmic results of rank decompositions. For instance, we immediately get the following.

Theorem 4.5 *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 4.4. 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 [11, 15, 21, 26], however, note that some of the solutions given therein do not have FPT runtime.

Applying the algorithm of [18] 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 $\rho(H') \leq \rho(H)$.

5 Conclusion

The alternative definition of rankwidth given in this paper, using R_k-join decompositions, should prove useful both for visualizing graphs of rankwidth k and for developing fast dynamic programming algorithms that could be practical for low values of k. 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 a paper by Charbit, Thomassé and Yeo [2].

We are working on algorithms for a general class of vertex subset and vertex partitioning problems for H-join decomposable graphs, see [28], that will also have runtime single exponential in $\rho(H)$. We believe that the very general notion of H-join decompositions deserves further study of its own. A major result would be to find another graph class $H_1, H_2, H_3, \ldots$, different from $R_1, R_2, R_3, \ldots$, satisfying the three properties on the desiderata list of the introduction.

References

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

[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.

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

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

[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

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

[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)

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

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

[10] R.Downey and M.Fellows. Parameterized Complexity, Springer-Verlag (1999)

[11] 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

[12] F.Fomin, P. Golovach, D.Lokshtanov and S.Saurabh Clique-width: on the price of generality. To appear SODA'09

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

[14] R.Ganian, P. Hliněný Automata approach to graphs of bounded rank-width. Proceedings *IWOCA 2008*, LNCS

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

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

[17] 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

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

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

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

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

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

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

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

[25] R. Paige, R. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing* 16(6):973989, 1987.

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

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

[28] J.A.Telle, A.Proskurowski. Algorithms for vertex partitionign problems on partial k-trees, *SIAM Journal Discrete Math.* 10: 529-550, 1997.