REPORTS
IN
INFORMATICS

ISSN 0333-3590

Algorithms for vertex subset and vertex partitioning problems with runtime single exponential in rankwidth

B.-M.Bui-Xuan, M.Vatshelle, J.A.Telle

REPORT NO 379 **November 2008**

Department of Informatics

UNIVERSITY OF BERGEN

Bergen, Norway

This report has URL http://www.ii.uib.no/publikasjoner/texrap/pdf/2008-379.pdf

Reports in Informatics from Department of Informatics, University of Bergen, Norway, is available at http://www.ii.uib.no/publikasjoner/texrap/.

Requests for paper copies of this report can be sent to:

Department of Informatics, University of Bergen, Høyteknologisenteret,
P.O. Box 7800, N-5020 Bergen, Norway

Algorithms for vertex subset and vertex partitioning problems with runtime single exponential in rankwidth

Binh-Minh BUI-XUAN Jan Arne TELLE Martin VATSHELLE

Department of Informatics, University of Bergen, Norway.

Abstract

Let σ and ρ be finite or co-finite subsets of natural numbers. A subset of vertices S of a graph $G = (V, E)$ is a (σ, ρ)-set of G if

$$\forall v \in V : |N(v) \cap S| \in \begin{cases} \sigma & \text{if } v \in S \\ \rho & \text{if } v \in V \setminus S \end{cases}$$

A *degree constraint* matrix D_q is a q by q matrix with entries being finite or co-finite subsets of natural numbers. A D_q *-partition* in a graph $G = (V, E)$ is a partition $V_1, V_2, ..., V_q$ of V such that for $1 \leq i, j \leq q$ we have

$$\forall v \in V_i : |N_G(v) \cap V_j| \in D_q[i, j]$$

For graphs of rankwidth k, given with a rank-decomposition of width k, we show how to solve any minimization or maximization problem over (σ, ρ)-sets in time $O(n * (m + 2^{3k^2 d+2k} dk))$, with d being the maximum member of σ and ρ (or of the complement set if it is co-finite). We also show how to decide if the graph contains a D_q-partition in time $O(n * (m + 2^{3k^2 dq+k} kd^2))$, now with d taken over all $D_q[i, j]$.

1 Introduction

This technical report is a continuation of the work reported in our paper [3]. For an introduction to rankwidth and discussion of related work please see that paper. In that paper we introduce H-join decompositions and give, for each integer k, a graph R_k such that a rank decomposition of width k is identical to an R_k-join decomposition. In this technical report we apply this view of graphs of rankwidth k to solve a class of vertex subset and a class of vertex partitioning problems in time single exponential in rankwidth, as described in the abstract. Section 2 gives all the necessary definitions and results that we need from paper [3]. This is followed by two Sections, the first giving algorithms for the vertex subset problems and the second giving the algorithms for the vertex partitioning problems. For a discussion of common problems that can be formulated as these vertex subset and vertex partioning problems, see for example [2, 1]. The extension in [1] that allows, for each vertex, membership in only certain partition classes, is easily accommodated into our algorithms. A later version of this paper will contain a better introduction with discussion of related work. Let us remark that our results can be expressed more generally for H-join decompositions simply by replacing rankwidth k with the rank of the adjacency matrix of the graph H over GF(2).

2 H-join decomposable graphs and rankwidth

Definition 2.1 *Let H be a bipartite graph with colour 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 colour 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 colour class. However, note that we do allow one isolated vertex in each colour 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.*

The link between modular decomposition and H-join decomposition is reflected 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.*

We now 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 any 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 colour 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.*

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 $\rho(H)$.*

For algorithmic use of H-join decompositions, we must be able to compute 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 $\{S_u, S_v\}$. For this we use the following simple result.

Lemma 2.8 *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 2.9 *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 2.9 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.

All results in this section were taken from [3], where all proofs can be found.

3 Dynamic Programming for σ, ρ problems

Definition 3.1 *Let σ and ρ be finite or co-finite subsets of natural numbers. A subset of vertices S of a graph G is a (σ, ρ)-set of G if*

$$\forall v \in V : |N(v) \cap S| \in \begin{cases} \sigma & \text{if } v \in S \\ \rho & \text{if } v \in V \setminus S \end{cases}$$

We consider the problem of computing the size of a minimum or maximum σ, ρ-set in a graph G given with its rank decomposition (T, δ) having rank-width k.

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 certain equivalence classes of vertex subsets that we need to define. To do this, we notice that for σ and ρ being finite sets, if a node has more neighbours than the highest element in the sets, then it does not matter how many. Similar property holds for co-finite.

Formally we use σ and ρ to define $d(\sigma, \rho)$ as follows:
For any set $\mu \subset \mathbb{N}$, let $c(\mu) = min(max_{x \in \mathbb{N}} x : x \in \mu, max_{x \in \mathbb{N}} x : x \notin \mu)$.
We then define $d(\sigma, \rho) = max(c(\sigma) + 1, c(\rho) + 1)$.

If $\mu = \mathbb{N}$ then there is no restriction on the set, and hence it need not to be considered. We can therefore define $c(\mathbb{N}) = -1$. For simplicity we denote $d = d(\sigma, \rho)$.

Two vertex subsets $X, Y \subseteq V(G)$ are equivalent, with respect to d and vertex subset $A \subseteq V(G)$, if every vertex outside A either has the same number of neighbours in X as it has in Y, or at least d neighbours in both.

Definition 3.2 (d-neighbour equivalence) *For a fixed graph G and a vertex subset $A \subseteq V(G)$, consider two vertex subsets $X \subseteq A$ and $Y \subseteq A$ and define $X \equiv_A^d Y$ if and only if*

$$\forall v \in \overline{A} : (|N(v) \cap X| = |N(v) \cap Y|) \vee (|N(v) \cap X| \geq d \wedge |N(v) \cap Y| \geq d)$$

Let w be a node in T_r and let the cut associated with w be $(V_w, \overline{V_w})$. Note that V_w are the vertices of G mapped by δ to leaves of the subtree of T_r rooted at w. The following bound is crucial to get runtime FPT when parameterized by rank-width.

Lemma 3.3 *For a fixed graph G with rank-width k and the vertex sets $S \subseteq V_w \subseteq V(G)$, $\forall\, d$ there exists a vertex set $R \subseteq S$ such that $|R| \leq dk$ and $R \equiv_{V_w}^d S$*

Proof We give an algorithm to find such a set $R \subseteq S$.

$R = \emptyset$
for 1 **to** d **do:**
find a minimal $Y \subseteq (S \setminus R)$ such that $N(Y) \setminus V_w = N(S \setminus R) \setminus V_w$, and such that no two vertices of Y has the same neighbourhood in $\overline{V_w}$
$R = R \cup Y$
endfor

This algorithm uses time $O(|S|2^k d)$.

Since the graph has rank-width k we know that $|Y| \leq k$, since otherwise there is a subset of Y with same neighbourhood, and then Y is not minimal. Now R will be a subset of S, and $|R| \leq dk$. We pick Y such that $N(Y) = N(S \setminus R) \setminus V_w$ in each step to ensure that every node in $\overline{V_w}$ which have some neighbour in $S \setminus R$ is hit. This means that $\forall v \in \overline{V_w}$ we have that $|N(v) \cap S| \geq d \Leftrightarrow |N(v) \cap R| \geq d$, otherwise there is at most $d-1$ nodes in $S \cap N(v)$. At each step we will pick some node in $S \cap N(v)$ until we picked all the nodes, then we have $|S \cap N(v)| = |R \cap N(v)|$, hence we have $R \equiv^d_{V_w} S$. □

Theorem 3.4 *The number of equivalence classes of $\equiv^d_{V_w}$ is at most $2^{k^2 d}$.*

Proof In Lemma 3.3 we present an algorithm which for any set S finds a set R equivalent to S such that $|R| \leq dk$. There is at most $(2^k)^{kd} = 2^{k^2 d}$ such sets. □

Note that there could be two subsets $S_1, S_2 \subseteq V_w$ that when run on the algorithm above would give two sets R_1, R_2 with $R_1 \neq R_2$ but still we could have $R_1 \equiv^d_{V_w} R_2$. We therefore need to be careful in defining canonical representatives of these equivalence classes. The main idea is to assume a total ordering of the vertices of $V(G)$ and use the lexicographically smallest sub set of V_w as canonical representative.

Definition 3.5 *Let $X \subseteq V_w$. The canonical representative $can^d_{V_w}(X)$ is defined as the lexicographically smallest set $R \subseteq V_w$ such that: $|R|$ is minimized and $R \equiv^d_{V_w} X$.*

Given $X \subseteq V_w$ we discuss how to compute the canonical representative $can^d_{V_w}(X)$. In a preprocessing step we first compute a mapping from the list of all possible subsets of size at most kd of V_w to the canonical representatives. To do this we go through the list of $2^{k^2 d}$ possible subsets of size at most kd in lexicographic order, and check whether it is equivalent to some of the representatives already in the computed list. If so we add a pointer to that representative, otherwise we add the subset to the so far computed list of representatives, and make a pointer to itself. This takes $O(2^{2k^2 d + k} k d^2)$ time.

Now, to find $can^d_{V_w}(X)$, we first run the algorithm in Lemma 3.3 to get R_X, then partition R_X according to the external module partition. For each partition class, containing say c elements, we substitute the c elements of that class of R_X with the lexicographically first c elements of the corresponding partition class of V_w. Now we look up in the table of mappings computed in the preprocessing step and find the canonical representative. This can be done in $O(|X|2^k d)$ time. We have showed that

Proposition 3.6 *For all $X \subseteq V_w$, the canonical representative $R = can^d_{V_w}(X)$ is such that $|R| \leq kd$, this can be computed in $O(|X|2^k d)$ time.*

3.1 Define Tables

Let us discuss the tables used in dynamic programming. The table Tab_w associated with a node w of T_r will have index set $\{can^d_{V_w}(X) \times can^d_{\overline{V_w}}(Y) : X \subseteq V_w, Y \subseteq \overline{V_w}\}$. To define the contents of the table we first need the concept of σ, ρ-domination.

Definition 3.7 *Let μ be a finite or co-finite subset of natural numbers.*
A subset of vertices $S \subseteq A \subseteq V(G)$ μ-dominates A if $\forall v \in A : |N(v) \cap S| \in \mu$

Definition 3.8 *For a fixed graph G and vertex subset $A \subseteq V(G)$, consider two vertex subsets $S \subseteq A$ and $X \subseteq \overline{A}$. We say that (S, X) σ, ρ-dominates A if $(S \cup X)$ σ-dominates S and $(S \cup X)$ ρ-dominates $A \setminus S$*

The reason we define equivalence classes the way we do is illustrated by the following lemma:

Lemma 3.9 *For a fixed graph G and vertex subset $A \subseteq V(G)$, consider three vertex subsets $S \subseteq A$ and $X, Y \subseteq \overline{A}$ such that $X \equiv_{\overline{A}}^{d} Y$. Then $S \cup X$ μ-dominates $A \Leftrightarrow S \cup Y$ μ-dominates A*

Proof And from Definition 3.7 we have: $S \cup X$ μ-dominates $A \Leftrightarrow \forall v \in A : |N(v) \cap (S \cup X)| \in \mu$ and $S \cup Y$ μ-dominates $A \Leftrightarrow \forall v \in A : |N(v) \cap (S \cup Y)| \in \mu$.
Hence we need to show $\forall v \in A : |N(v) \cap (S \cup X)| \in \mu \Leftrightarrow |N(v) \cap (S \cup Y)| \in \mu$.
Since X and Y are disjoint from S this is equivalent to:
$\forall v \in A : |N(v) \cap S| + |N(v) \cap X| \in \mu \Leftrightarrow |N(v) \cap S| + |N(v) \cap Y| \in \mu$.
From Definition 3.2 we have:
$\forall v \in A : (|N(v) \cap X| = |N(v) \cap Y|) \vee (|N(v) \cap X| \geq d \wedge |N(v) \cap Y| \geq d)$
There are two cases that need to be considered

1. $|N(v) \cap X| = |N(v) \cap Y|$, then it is obviously true.
2. $|N(v) \cap X| \geq d \wedge |N(v) \cap Y| \geq d$, then since all values $\geq d$ are in μ by choice of d, we know that $|N(v) \cap S| + |N(v) \cap X| \in \mu$ and $|N(v) \cap S| + |N(v) \cap Y| \in \mu$

□

By combining Definition 3.8 and Lemma 3.9 we get the following corollary.

Corollary 3.10 *For a fixed graph G and vertex subset $A \subseteq V(G)$, consider three vertex subsets $S \subseteq A$, $X \subseteq \overline{A}$ and $Y \subseteq \overline{A}$ such that $X \equiv_{A}^{d} Y$. Then (S, X) σ, ρ-dominates $A \Leftrightarrow (S, Y)\sigma, \rho$-dominates A*

We define the contents of $Tab_w[R_X][R_Y]$ where $R_X = can_{V_w}^{d}(X)$ and $R_Y = can_{\overline{V_w}}^{d}(Y)$ as:

$$Tab_w[R_X][R_Y] \stackrel{\text{def}}{=} opt_{S \subseteq V_w}\{|S| : S \equiv_{V_w}^{d} R_X \wedge (S, R_Y) \ \sigma, \rho\text{-dominates } V_w\}.$$

In the above opt should be replaced by either $maximum$ or $minimum$ depending on wether we are looking for a largest or smallest (σ, ρ)-set respectively.

Lemma 3.11 *For a fixed graph G and vertex subset $S \subseteq V(G)$, $\forall A, B \subseteq V(G)$ we have:*
S μ-dominates A and S μ-dominates $B \Leftrightarrow$ S μ-dominates $A \cup B$

Proof
$(\Rightarrow)$ S μ-dominates A and S μ-dominates B.
This means: $\forall v \in A : |N(v) \cap S| \in \mu$ and $\forall v \in B : |N(v) \cap S| \in \mu$
By unifying these we get: $\forall v \in (A \cup B) : |N(v) \cap S| \in \mu$.
Hence S μ-dominates $A \cup B$.
$(\Leftarrow)$ S μ-dominates $A \cup B$.
This means: $\forall v \in (A \cup B) : |N(v) \cap S| \in \mu$. Then it easily follows that $\forall v \in A : |N(v) \cap S| \in \mu$ and $\forall v \in B : |N(v) \cap S| \in \mu$
Hence S μ-dominates A and S μ-dominates B. □

By combining Definition 3.8 and Lemma 3.11 we get the following corollary.

Corollary 3.12 *For a fixed graph G and vertex subset $S \subseteq V(G)$, $\forall A, B \subseteq V(G)$ we have:*
$(S \cap A, S \cap \overline{A})$ σ, ρ-dominates A and $(S \cap B, S \cap \overline{B})$ σ, ρ-dominates $B \Leftrightarrow$ $(S \cap (A \cup B), S \cap \overline{A \cup B})$ σ, ρ-dominates $A \cup B$

3.2 Table initialization

All values of all tables should first be set to $\pm\infty$ depending on whether it is a minimization or maximization problem.

At a leaf u of T_r there are only two possible subsets, either $\emptyset$ or u. For vertices in $V \setminus u$ there are 2 types, those who are neighbours to u and those who are not. We therefore have at most $2 * d^2$ entries of Tab_u. Let $X \subseteq V \setminus u$, then since (u, X) σ, ρ-dominates $\{u\} \Leftrightarrow |N(u) \cap X| \in \sigma$ and $(\emptyset, X)$ σ, ρ-dominates $\{u\} \Leftrightarrow |N(u) \cap X| \in \rho$.
$\forall R_X$ such that R_X is a canonical representative of $\equiv^d_{V_u}$ $Tab_u[u][R_X] = |N(u) \cap X| \in \sigma$ and $Tab_u[\emptyset][R_X] = |N(u) \cap X| \in \rho$.

3.3 Table update

Let w be a vertex of T_r with children a and b.
$\forall R_{\overline{w}}$ such that $R_{\overline{w}}$ is a canonical representative of $\equiv^d_{V_w}$
$\forall R_a$ such that R_a is a canonical representative of $\equiv^d_{V_a}$
$\forall R_b$ such that R_b is a canonical representative of $\equiv^d_{V_b}$
Let $R_w = can^d_{V_w}(R_a \cup R_b)$, $R_{\overline{a}} = can^d_{V_a}(R_b \cup R_{\overline{w}})$ and $R_{\overline{b}} = can^d_{V_b}(R_a \cup R_{\overline{w}})$
$Tab_w[R_w][R_{\overline{w}}] = opt(Tab_w[R_w][R_{\overline{w}}], Tab_a[R_a][R_{\overline{a}}] + Tab_b[R_b][R_{\overline{b}}])$

Theorem 3.13 *The table at node w is updated correctly, namely for any $R_w, R_{\overline{w}}$*

$$Tab_w[R_w][R_{\overline{w}}] \leq s \Leftrightarrow \exists S_w \subseteq V_w : |S_w| \leq s \wedge R_w \equiv^d_{V_w} S_w \wedge (S_w, R_{\overline{w}})\sigma, \rho - dominates\ G[V_w]$$

Proof Let a, b be the children of w in T_r, assume Tab_a and Tab_b are correct.
$(\Rightarrow)$ $Tab_w[R_w][R_{\overline{w}}] \leq s$ means that such an update happened in the algorithm, hence there exists R_a and R_b such that: $R_w = can^d_{V_w}(R_a \cup R_b)$ and $Tab_a[R_a][R_{\overline{a}}] + Tab_b[R_b][R_{\overline{b}}] \leq s$. Where $R_{\overline{a}} = can^d_{V_a}(R_b \cup R_{\overline{w}})$ and $R_{\overline{b}} = can^d_{V_b}(R_a \cup R_{\overline{w}})$ Then we know that there exists S_a and S_b such that $(S_a, R_{\overline{a}})$ σ, ρ-dominates V_a and $(S_b, R_{\overline{b}})$ σ, ρ-dominates V_b and that $|S_a \cup S_b| \leq s$. Let $S_w = S_a \cup S_b$, then S_w fulfil the two conditions $|S| \leq s$ and $R_w \equiv^d_{V_w} S_w$, now we need to show that $(S_w, R_{\overline{w}})$ σ, ρ-dominates V_w.

Since $(S_b \cup R_{\overline{w}}) \equiv^d_{V_a} R_{\overline{a}}$ and $(S_a, R_{\overline{a}})\sigma, \rho$-dominates V_a we have from Corollary 3.10 that $(S_a, S_b \cup R_{\overline{w}})\sigma, \rho$-dominates V_a. Similarly we conclude that $(S_b, S_a \cup R_{\overline{w}})\sigma, \rho$-dominates V_b Let $S = S_w \cup R_{\overline{w}} = S_a \cup S_b \cup R_{\overline{w}}$ then we get $(S \cap V_a, S \setminus V_a)\sigma, \rho$-dominates V_a and $(S \cap V_b, S \setminus V_b)\sigma, \rho$-dominates V_b.

From Corollary 3.12 we get that $(S \cap V_w, S \setminus V_w)\sigma, \rho$-dominates V_w. Or equivalently $(S_w, R_{\overline{w}})\sigma, \rho$-dominates V_w.
$(\Leftarrow)\exists S_w \subseteq V_w : |S_w| \leq s \wedge S_w \equiv^d_{V_w} R_w \wedge (S_w, R_{\overline{w}})\sigma, \rho$-dominates V_w. Let $S_a = S_w \cap V_a$ and $S_b = S_w \cap V_b$ Since the algorithm goes through all triples, it will at some point go through $(R_a, R_b, R_{\overline{w}})$, where $R_a = can^d_{V_a}(S_a)$ and $R_b = can^d_{V_b}(S_b)$.

Let $S = S_a \cup S_b \cup R_{\overline{w}}$ Then Corollary 3.12 tells us that $(S_a, S_b \cup R_{\overline{w}})\sigma, \rho$-dominates V_a and $(S_b, S_a \cup R_{\overline{w}})\sigma, \rho$-dominates V_b. Since in the algorithm $R_{overlinea} = can^d_{V_a}(S_b \cup R_{\overline{w}})$, Corollary 3.10 give us that $(S_a, R_{\overline{a}})$ σ, ρ-dominates V_a. Similarly we get that $(S_b, R_{\overline{b}})$ σ, ρ-dominates V_b. This means that $Tab_a[R_a][R_{\overline{a}}] + Tab_b[R_b][R_{\overline{b}}] \leq |S_a \cup S_b| = |S_w| \leq s$, hence $Tab_w[R_w][R_{\overline{w}}] \leq s$. By induction all tables will be correct. □

The solution to the problem is found by optimizing over all entries in the table at the root.

Theorem 3.14 *For a graph G of rank-width k, given with its rank-decomposition, any σ, ρ-problem can be solved in*
$O(n * (m + 2^{3k^2d+k}kd^2))$ *time.*

Proof Calculating external module partitions take $O(m)$ time. Computing the list of representatives takes $O(2^{2k^2d+k}kd^2)$ time. The intialization of tables takes $O(2^{k^2d})$ time. The update loops over all possibilities of 3 canonical representatives. From Theorem 3.4 we know that there is at most 2^{k^2d} canonical

representatives. The only work we do in the inner loop is finding canonical representatives, this takes $O(2^k kd^2)$ time since the set has at most kd vertices. Hence the total runtime of any σ, ρ-problem is $O(n * (m + 2^{3k^2 d+k} kd^2))$ □

4 Dynamic Programming for Vertex Partitioning problems

We extend to problems asking for a partition of the vertex set into q classes with each class satisfying a certain σ, ρ-property, as follows.

Definition 4.1 *A degree constraint matrix D_q is a q by q matrix with entries being finite or co-finite subsets of natural numbers $\{0, 1, ...\}$. A D_q-partition in a graph G is a partition $V_1, V_2, ..., V_q$ of $V(G)$ such that for $1 \leq i, j \leq q$ we have $\forall v \in V_i : |N_G(v) \cap V_j| \in D_q[i, j]$.*

For technical reasons, we will allow a partition $V_1, ..., V_q$ of $V(G)$ to possibly have some empty partition classes, *i.e.*, if the degree constraints on a partition class V_i are satisfied by $V_i = \emptyset$ then we allow this possibility. Given a degree constraint matrix D_q, it is natural to ask about the existence of a D_q-partition in an input graph. We call this the $\exists D_q$ problem. We might also ask for an extremal value of the cardinality of a vertex partition class over all D_q-partitions. Additionally, given a sequence of degree constraint matrices, $D_1, D_2, ...$, we might want to find an extremal value of q for which a D_q-partition exists in the input graph. We call these partition minimization and partition maximization problems.
We concentrate on the $\exists D_q$ problem where all entries in the D_q matrix are either finite or co-finite. Extending to algorithms that handle also the cases mentioned above is quite straightforward. Let d be one larger than the largest integer that either belongs to a finite element of D_q or does not belong to a co-finite element of D_q. The equivalence relation we need is the following:

Definition 4.2 (q-partition, d-neighbourhood equivalence) *For a fixed graph G and vertex subset $A \subseteq V(G)$, consider two families of sets $(X_1, X_2, ..., X_q)$ and $(Y_1, Y_2, ..., Y_q)$ such that $\forall i : X_i \subseteq A \wedge Y_i \subseteq A$ and define*

$$(X_1, X_2, ..., X_q) \equiv_A^{q,d} (Y_1, Y_2, ..., Y_q)$$

if and only if

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

From Definitions 3.8 and 4.2 we get the following

Claim 4.3 *$(X_1, X_2, ..., X_q) \equiv_A^{q,d} (Y_1, Y_2, ..., Y_q)$ if and only if $\forall i X_i \equiv_A^d Y_i$.*

Corollary 4.4 *The number of equivalence classes of $\equiv_A^{q,d}$ is at most 2^{qdk^2}.*

Proof According to Claim 4.3 each of the q sets are picked independently. And from Lemma 3.4 we know that there is at most 2^{dk^2} equivalence classes of $\equiv_A^d$. Since we have q such sets there will be at most $(2^{dk^2})^q = 2^{qdk^2}$ equivalence classes of $\equiv_A^{q,d}$. □

For a node w of T_r, let $\mathcal{X} = (X_1, ..., X_q) : X_i \subseteq V_w$. Computing the canonical representative of this q-tuple is done by repeating q times the procedure for canonical representatives for d-neighbour equivalence, in other words $can_{V_w}^d(\mathcal{X}) = \{can_{V_w}^d(X_1), ..., can_{V_w}^d(X_q)\}$. Likewise, we compute a list of all canonical representatives by combining q times the result of the method used for d-neighbour equivalence.

The table Tab_w associated with w will have index set

$$\{can_{V_w}^d(X_1) \times ... \times can_{V_w}^d(X_q)\} \times \{can_{V_w}^d(Y_1) \times ... \times can_{V_w}^d(Y_q)\} : X_i \subseteq V_w \wedge Y_i \subseteq \overline{V_w}$$

When the node w is clear we refer to an entry in this set simply as $\mathcal{R}_X = \{R_{X_1}, ..., R_{X_q}\}$ and $\mathcal{R}_Y = \{R_{Y_1}, ..., R_{Y_q}\}$. To define the contents of the table we first need the concept of D_q-dominating a subgraph.

Definition 4.5 *For a fixed graph G and vertex subset $A \subseteq V(G)$, consider $\mathcal{X} = (X_1, ..., X_q) \in A^q$ and $\mathcal{Y} = (Y_1, Y_2, ..., Y_q) \in \overline{A}^q$. We say that $(\mathcal{X}, \mathcal{Y})$ D_q-dominates A if $\forall i, j : (X_j \cup Y_j)$ $D_q[i, j]$-dominates X_i.*

We define the contents of $Tab_w[\mathcal{R}_X][\mathcal{R}_Y]$ as

$$Tab_w[\mathcal{R}_X][\mathcal{R}_Y] \stackrel{\text{def}}{=} \left\{ \begin{array}{ll} 1 & \text{if } \exists \text{ partition } \mathcal{S} = (S_1, ...S_q) \text{ of } V_w \text{ such that:} \\ & \mathcal{S} \equiv^{q,d}_{V_w} \mathcal{R}_X \text{ and } (\mathcal{S}, \mathcal{R}_Y) \ D_q\text{-dominates } V_w \\ 0 & otherwise \end{array} \right\}$$

4.1 Table initialization

All values of all tables should first be set to $FALSE$. Let us discuss how many entries there are in the table Tab_u, at a leaf of T_r corresponding to a vertex u of G. Firstly, there are q possible classes u could be in. Secondly, for vertices in $V \setminus u$ there are 2 types, those who are neighbours to u and those who are not, we have the choice of at most d of each of these 2 types in each of the q partition classes. We therefore have at most qd^{2q} choices for all the entries of Tab_u.
For $i = 1$ *to* q let $\mathcal{X}$ be the set family where class $X_i = \{u\}$ and all other classes are empty.
$\forall \mathcal{R}_Y \in \mathcal{C}^q_{V_{\overline{u}}}$, we then have that $(\mathcal{X}, \mathcal{R}_Y)$ D_q-dominates $\{u\} \Leftrightarrow \forall j | N(u) \cap R_{Y_j}| \in D_q[i, j]$.
So we set $Tab_u[\mathcal{R}_X][\mathcal{R}_Y]$ to be TRUE if and only if $\forall j \ |N(u) \cap R_{Y_j}| \in D_q[i, j]$

4.2 Table update

The notation $\bigcup_q$ means componentwise union of two q-tuples. Let w be a vertex of T_r with children a and b. We update the table at w as follows:
$\forall \mathcal{R}_{\overline{w}}$, canonical representative of $\equiv^{d,q}_{V_w}$
$\forall \mathcal{R}_a$, canonical represetantive of $\equiv^{d,q}_{V_a}$
$\forall \mathcal{R}_b$, canonical represetantive of $\equiv^{d,q}_{V_b}$
Let $\mathcal{R}_w = can^d_{V_w}(\mathcal{R}_a \bigcup_q \mathcal{R}_b)$, $\mathcal{R}_{\overline{a}} = can^d_{V_a}(\mathcal{R}_b \bigcup_q \mathcal{R}_{\overline{w}})$ and $\mathcal{R}_{\overline{b}} = can^d_{V_b}(\mathcal{R}_a \bigcup_q \mathcal{R}_{\overline{w}})$
If $Tab_w[\mathcal{R}_w][\mathcal{R}_{\overline{w}}] = FALSE$ then $Tab_w[\mathcal{R}_w][\mathcal{R}_{\overline{w}}] = Tab_a[\mathcal{R}_a][\mathcal{R}_{\overline{a}}] \wedge Tab_b[\mathcal{R}_b][\mathcal{R}_{\overline{b}}]$

Theorem 4.6 *The table at node w is updated correctly. In other words, for any $\mathcal{R}_w, \mathcal{R}_{\overline{w}}$, we have that $Tab_w[\mathcal{R}_w][\mathcal{R}_{\overline{w}}] = TRUE$ if and only if $\exists$ partition $\mathcal{S} = (S_1, ...S_q)$ of V_w such that:*
$(\mathcal{S}, \mathcal{R}_{\overline{w}})$ D_q-dominates V_w and $\mathcal{S} \equiv^{d,q}_{V_w} \mathcal{R}_w$

Proof Let a, b be the children of w in T_r, assume Tab_a and Tab_b are correct.
$(\Rightarrow)$ For this direction of the proof we have that $Tab_w[\mathcal{R}_w][\mathcal{R}_{\overline{w}}] = TRUE$.
Then there must exist some $\mathcal{R}_a, \mathcal{R}_b$ such that $Tab_a[\mathcal{R}_a][\mathcal{R}_{\overline{a}}] = TRUE$ and $Tab_b[\mathcal{R}_b][\mathcal{R}_{\overline{b}}] = TRUE$, where $\mathcal{R}_{\overline{a}} = can^d_{V_a}(\mathcal{R}_b \bigcup_q \mathcal{R}_{\overline{w}})$ and $\mathcal{R}_{\overline{b}} = can^d_{V_b}(\mathcal{R}_a \bigcup_q \mathcal{R}_{\overline{w}})$.
Hence there exists $\mathcal{S}_a$ partition of V_a and $\mathcal{S}_b$ partition of V_b such that
$(\mathcal{S}_a, \mathcal{R}_{\overline{a}})$ D_q-dominates V_a $(\mathcal{S}_b, \mathcal{R}_{\overline{b}})$ D_q-dominates V_b.
This means that $\forall i, j : (S_{a_j} \cup R_{\overline{a}_j})$ $D_q[i, j]$-dominates S_{a_i} and $\forall i, j : (S_{b_j} \cup R_{\overline{b}_j})$ $D_q[i, j]$-dominates S_{b_i}.
By Lemma 3.9 we have: $\forall i, j : (S_{a_j} \cup_q S_{b_j} \cup R_{\overline{w}_j})$ $D_q[i, j]$-dominates S_{a_i}
hence $\forall i, j : (S_{w_j} \cup_q R_{\overline{w}_j})$ $D_q[i, j]$-dominates S_{a_i} and similarly we have $\forall i, j : (S_{w_j} \cup_q R_{\overline{w}_j})$ $D_q[i, j]$-dominates S_{b_i}.
By Lemma 3.11 we have: $\forall i, j : (S_{w_j} \cup R_{\overline{w}_j})$ $D_q[i, j]$-dominates S_{w_i}.
Which means $(\mathcal{S}, \mathcal{R}_{\overline{w}})$ D_q-dominates V_w
$(\Leftarrow)$ For this direction of the proof we have that there exists a partition $\mathcal{S} = (S_1, ...S_q)$ of V_w such that:
$(\mathcal{S}, \mathcal{R}_{\overline{w}})$ D_q-dominates V_w
This means that $\forall i, j : (S_{w_j} \cup R_{\overline{w}_j})$ $D_q[i, j]$-dominates S_{w_i}.
Let $\mathcal{S}_a, \mathcal{S}_b$ be the componentwise intersection of $\mathcal{S}_w$ with V_a, V_b respectively.

By Lemma 3.11 we have: $\forall i,j : (S_{w_j} \cup R_{\overline{w}_j})\ D_q[i,j]$-dominates S_{a_i} and $\forall i,j : (S_{w_j} \cup R_{\overline{w}_j})\ D_q[i,j]$-dominates S_{b_i}.
Hence $\forall i,j : (S_{a_j} \cup S_{b_j} \cup R_{\overline{w}_j})\ D_q[i,j]$-dominates S_{a_i} and $\forall i,j : (S_{a_j} \cup S_{b_j} \cup R_{\overline{w}_j})\ D_q[i,j]$-dominates S_{a_i}.
By Lemma 3.9 we have:
$\forall i,j : (S_{a_j} \cup R_{\overline{a}_j})\ D_q[i,j]$-dominates S_{a_i} and $\forall i,j : (S_{b_j} \cup R_{\overline{b}_j})\ D_q[i,j]$-dominates S_{b_i} where $\mathcal{R}\overline{a} = can^d_{\overline{V_a}}(\mathcal{S}_b \bigcup_q \mathcal{R}_{\overline{w}})$ and $\mathcal{R}_{\overline{b}} = can^d_{\overline{V_b}}(\mathcal{S}_a \bigcup_q \mathcal{R}_{\overline{w}})$ Let $\mathcal{R}_a = can^d_{V_a}(\mathcal{S}_a)$ and $\mathcal{R}_b = can^d_{V_b}(\mathcal{S}_b)$ then $Tab_a[\mathcal{R}_a][\mathcal{R}_{\overline{a}}] = TRUE$ and $Tab_b[\mathcal{R}_b][\mathcal{R}_{\overline{b}}] = TRUE$.
Since the algorithm goes through all triples, it will at some point go through $(\mathcal{R}_a, \mathcal{R}_b, \mathcal{R}_{\overline{w}})$ And it will set $Tab_w[\mathcal{R}_w][\mathcal{R}_{\overline{w}}]$ to true, once it is true it will newer change.
By induction all tables will be correct. □

The solution to the problem is given by checking if some entry in the table at the root has value TRUE.

Theorem 4.7 *For a graph G with rank-width k, any D_q-problem can be solved in $O(n * (m + 2^{3k^2dq+k}dk))$ time.*

Proof Calculating external module partitions take $O(m)$ time. Computing the list of representatives takes $O(2^{qk^2d+k}kd^2)$ time. The intialization of tables takes $O(2^{qk^2d})$ time. The update loops over all possibilities of 3 canonical representatives. From Theorem 3.4 we know that there is at most 2^{k^2dq} canonical representatives. The only work we do in the inner loop is finding canonical representatives, this takes $O(2^k kqd^2)$ time since each of the q sets have at most $2kd$ vertices. Hence the total runningtime of any σ,ρ-problem is $O(n * (m + 2^{3k^2dq+k}kd^2))$ □

References

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

[2] J.A.Telle, A.Proskurowski, Algorithms for vertex-partitioning problems on partial k-trees. *SIAM J. Discrete Math* 10 (1997) 529-550 1

[3] B.-M.Bui-Xuan, M.Vatshelle, J.A.Telle, H-join and algorithms for graphs of bounded rankwidth, submitted SODA'09, Technical Report 378, Department of Informatics, University of Bergen, `http://www.ii.uib.no/publikasjoner/texrap/pdf/2008-378.pdf` 1, 2