Finding Branch-decompositions and Rank-decompositions

Petr Hliněný[1]⋆ and Sang-il Oum[2]⋆⋆

[1] Faculty of Informatics, Masaryk University,
Botanická 68a, 602 00 Brno, Czech Republic,
hlineny@fi.muni.cz
[2] Department of Combinatorics and Optimization,
University of Waterloo, Waterloo, Ontario N2L 3G1 Canada,
sangil@math.uwaterloo.ca

Abstract. We present a new algorithm that can output the rank-decomposition of width at most k of a graph if such exists. For that we use an algorithm that, for an input matroid represented over a fixed finite field, outputs its branch-decomposition of width at most k if such exists. This algorithm works also for partitioned matroids. Both these algorithms are fixed-parameter tractable, that is, they run in time $O(n^3)$ for each fixed value of k where n is the number of vertices / elements of the input. (The previous best algorithm for construction of a branch-decomposition or a rank-decomposition of optimal width due to Oum and Seymour [Testing branch-width. J. Combin. Theory Ser. B, **97**(3) (2007) 385–393] is not fixed-parameter tractable.)

Keywords: rank-width, clique-width, branch-width, fixed parameter tractable algorithm, graph, matroid.

1 Introduction

Many graph problems are known to be NP-hard in general. But for practical application we still need to solve them. One method to solve them is to restrict the input graph to have a certain structure. Clique-width, defined by Courcelle and Olariu [1], is very useful for that purpose. Many hard graph problems (in particular all those expressible in MSO logic of adjacency graphs) are solvable in polynomial time as long as the input graph has bounded clique-width and is given in the form of the decomposition for clique-width, called a *k-expression* [2–6]. A k-expression is an algebraic expression with the following four operations on vertex-labeled graph with k labels: create a new vertex with label i; take the disjoint union of two labeled graphs; add all edges between vertices of label i and label j; and relabel all vertices with label i to have label j. However, for fixed $k > 3$, it is not known how to find a k-expression of an input graph having clique-width at most k. (If $k \leq 3$, then it has been shown in [7, 8].)

⋆ Supported in part by the Institute of Theoretical Computer Science, project 1M0545.
⋆⋆ Partially supported by NSF grant 0354742.

Rank-width is another graph complexity measure introduced by Oum and Seymour [9], aiming at construction of an $f(k)$-expression of the input graph having clique-width k for some fixed function f in polynomial time. Rank-width is defined (Section 6) as the branch-width (see Section 2) of the *cut-rank* function of graphs. Rank-width turns out to be very useful for algorithms on graphs of bounded clique-width, since a class of graphs has bounded rank-width if and only if it has bounded clique-width. In fact, if rank-width of a graph is k, then its clique-width lies between k and $2^{k+1}-1$ [9] and an expression can be constructed from a rank-decomposition of width k.

In this paper, we are mainly interested in the following problem:

- Find a fixed-parameter tractable algorithm that outputs a rank-decomposition of width at most k if the rank-width of an input graph (with more than one vertex) is at most k.

The first rank-width algorithm by Oum and Seymour [9] only finds a rank-decomposition of width at most $3k+1$ for n-vertex graphs of rank-width at most k in time $O(n^9 \log n)$. This algorithm has been improved by Oum [10] to output a rank-decomposition of width at most $3k$ in time $O(n^3)$. Using this approximation algorithm and finiteness of excluded vertex-minors [11], Courcelle and Oum [12] have constructed an $O(n^3)$-time algorithm to decide whether a graph has rank-width at most k. However, this is only a decision algorithm; if the rank-width is at most k, then this algorithm verifies that the input graph contains none of the excluded graphs for rank-width at most k as a vertex-minor. It does *not* output a rank-decomposition showing that the graph indeed has rank-with at most k.

In another paper, Oum and Seymour [13] have constructed a polynomial-time algorithm that can output a rank-decomposition of width at most k for graphs of rank-width at most k. However, it is not fixed-parameter tractable; its running time is $O(n^{8k+12} \log n)$. Obviously, it is very desirable to have a fixed-parameter tractable algorithm to output such an "optimal" rank-decomposition, because most algorithms on graphs of bounded clique-width require a k-expression on their input. So far probably the only known efficient way of constructing an expression with bounded number of labels for a given graph of bounded clique-width uses rank-decompositions.

In this paper, we present an affirmative answer to the above problem. An amusing aspect of our solution is that we deeply use submodular functions and matroids to solve the rank-decomposition problem, which shows (somehow unexpectedly) a "truly geometrical" nature of this graph-theoretical problem. In fact we solve the following related problem on matroids, too.

- Find a fixed-parameter tractable algorithm that, given a matroid represented by a matrix over a fixed finite field, outputs a branch-decomposition of width at most k if the branch-width of the input matroid is at most k.

Here we actually bring together two separate lines of research; Oum and Seymour's above sketched work on rank-width and on branch-width of submodular functions, with Hliněný's work [14, 15] on parametrized algorithms for matroids over finite fields, to give the final solution of our first problem — Theorem 6.3.

We lastly remark that the following (indeed widely expected) hardness result has been given only recently by Fellows, Rosamond, Rotics, and Szeider [16]; it is NP-hard to find graph clique-width. To argue that it is NP-hard to find rank-width, we combine some known results: Hicks and McMurray Jr. [17] (independently Mazoit and Thomassé [18]) recently proved that the branch-width of the cycle matroid of a graph is equal to the branch-width of the graph if it is 2-connected. Hence we can reduce (Section 6) the problem of finding branch-width of a graph to finding rank-width of a certain bipartite graph, and finding graph branch-width is NP-hard as shown by Seymour and Thomas [19].

Our paper is structured as follows: The next section briefly introduces definitions of branch-width, partitions, matroids and the amalgam operation on matroids. After that (Section 3) we explain the notion of so-called titanic partitions, which we further use to "model" partitioned matroids in ordinary matroids. (At this point it is worth to note that partitioned matroids present the key tool that allows us to shift from a branch-width-testing algorithm [14] to a construction of an "optimal" branch-decomposition, see Theorem 4.4, and of a rank-decomposition.) In Section 4, we will discuss a simple but slow algorithm for matroid branch-decompositions. In Section 5, we will present a faster algorithm. As the main application we then use our result to give an algorithm for constructing a rank-decomposition of optimal width of a graph in Section 6.

2 Definitions

Branch-width. Let $\mathbb{Z}$ be the set of integers. For a finite set V, a function $f : 2^V \to \mathbb{Z}$ is called *symmetric* if $f(X) = f(V \setminus X)$ for all $X \subseteq V$, and is called *submodular* if $f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y)$ for all subsets X, Y of V. A tree is *subcubic* if all vertices have degree 1 or 3. For a symmetric submodular function $f : 2^V \to \mathbb{Z}$ on a finite set V, the branch-width is defined as follows.

A *branch-decomposition* of the symmetric submodular function f is a pair (T, μ) of a subcubic tree T and a bijective function $\mu : V \to \{t : t \text{ is a leaf of } T\}$. (If $|V| \leq 1$ then f admits no branch-decomposition.) For an edge e of T, the connected components of $T \setminus e$ induce a partition (X, Y) of the set of leaves of T. (In such a case, we say that $\mu^{-1}(X)$ (or $\mu^{-1}(Y)$) is *displayed* by e in the branch-decomposition (T, μ). We also say that V and $\emptyset$ are displayed by the branch-decomposition.) The *width* of an edge e of a branch-decomposition (T, μ) is $f(\mu^{-1}(X))$. The *width* of (T, μ) is the maximum width of all edges of T. The *branch-width* of f, denoted by $\mathrm{bw}(f)$, is the minimum of the width of all branch-decompositions of f. (If $|V| \leq 1$, we define $\mathrm{bw}(f) = f(\emptyset)$.)

A natural application of this definition is the branch-width of a graph, as introduced by Robertson and Seymour [20] along with better known tree-width, and its direct matroidal counterpart below in this section. We also refer to further formal definition of rank-width in Section 6.

Partitions. A *partition* $\mathcal{P}$ of V is a collection of nonempty pairwise disjoint subsets of V whose union is equal to V. Each element of $\mathcal{P}$ is called a *part*. For

a symmetric submodular function f on 2^V and a partition $\mathcal{P}$ of V, let $f^{\mathcal{P}}$ be a function on $2^{\mathcal{P}}$ (also symmetric and submodular) such that $f^{\mathcal{P}}(X) = f(\cup_{Y \in X} Y)$. The *width* of a partition $\mathcal{P}$ is $f(\mathcal{P}) = \max\{f(Y) : Y \in \mathcal{P}\}$.

Matroids. We refer to Oxley [21] in our matroid terminology. A *matroid* is a pair $M = (E, \mathcal{B})$ where $E = E(M)$ is the ground set of M (elements of M), and $\mathcal{B} \subseteq 2^E$ is a nonempty collection of *bases* of M, no two of which are in an inclusion. Moreover, matroid bases satisfy the "exchange axiom": if $B_1, B_2 \in \mathcal{B}$ and $x \in B_1 \setminus B_2$, then there is $y \in B_2 \setminus B_1$ such that $(B_1 \setminus \{x\}) \cup \{y\} \in \mathcal{B}$. A typical example of a matroid is given by a set of vectors (forming the columns of a matrix $\boldsymbol{A}$) with usual linear independence. The matrix $\boldsymbol{A}$ is then called a *representation* of the matroid.

All matroid bases have the same cardinality called the *rank* $r(M)$ of the matroid. Subsets of bases are called *independent sets*, and the remaining sets are *dependent*. A matroid M is *uniform* if all subsets of $E(M)$ of size $r(M)$ are the bases, and M is *free* if $E(M)$ is a basis. The *rank function* $r_M(X)$ in M is the maximum cardinality of an independent subset of a set $X \subseteq E(M)$. For a subset X of E, the *deletion* $M \setminus X$ of X from M or the *restriction* $M \restriction (E \setminus X)$ of M to $E \setminus X$, is the matroid on $E \setminus X$ in which $Y \subseteq E \setminus X$ is independent in $M \setminus X$ if and only if Y is an independent set of M.

To define the branch-width of a matroid, we consider its (symmetric and submodular) connectivity function $\lambda_M(X) = r_M(X) + r_M(E \setminus X) - r_M(E) + 1$ defined for all subsets $X \subseteq E = E(M)$. A "*geometric*" meaning is that the subspaces spanned by X and $E \setminus X$ intersect in a subspace of rank $\lambda_M(X) - 1$. *Branch-width* $\mathrm{bw}(M)$ and *branch-decompositions* of a matroid M are defined as the branch-width and branch-decompositions of λ_M. A pair $(M, \mathcal{P})$ is called a *partitioned matroid* if M is a matroid and $\mathcal{P}$ is a partition of $E(M)$. A *connectivity function* of a partitioned matroid $(M, \mathcal{P})$ is defined as $\lambda^{\mathcal{P}}_M$. *Branch-width* $\mathrm{bw}(M, \mathcal{P})$ and *branch-decompositions* of a partitioned matroid $(M, \mathcal{P})$ are defined as branch-width, branch-decompositions of $\lambda^{\mathcal{P}}_M$.

Amalgams of matroids. Let M_1, M_2 be matroids on E_1, E_2 respectively and $T = E_1 \cap E_2$. Moreover let us assume that $M_1 \restriction T = M_2 \restriction T$. If M is a matroid on $E_1 \cup E_2$ such that $M \restriction E_1 = M_1$ and $M \restriction E_2 = M_2$, then M is called an *amalgam* of M_1 and M_2 (see Fig. 1). It is known that an amalgam of two matroids need not exist (and need not be unique). However, in a special case we shall use here (see also Proposition 4.2), its existence easily follows from [21, 12.4.2];

Lemma 2.1. *If $M_1 \restriction T$ is free, then an amalgam of M_1 and M_2 exists.*

3 Titanic Partitions and Gadgets

Let V be a finite set and f be a symmetric submodular function on 2^V. A subset X of V is called *titanic* with respect to f if whenever A_1, A_2, A_3 are pairwise disjoint subsets of X such that $A_1 \cup A_2 \cup A_3 = X$, there is $i \in \{1, 2, 3\}$ such that

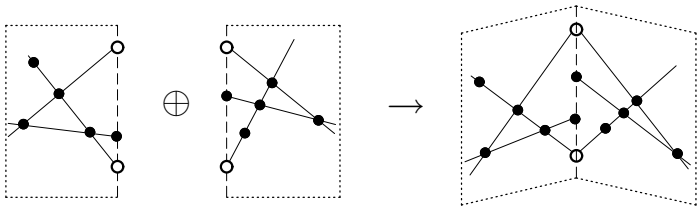

Fig. 1. A "geometrical" illustration of an amalgam of two matroids, in which hollow points are the shared elements T.

$f(A_i) \geq f(X)$. A partition $\mathcal{P}$ of V is called *titanic* with respect to f if every part of $\mathcal{P}$ is titanic with respect to f. The following lemma is equivalent to a lemma by Geelen, Gerards, and Whittle [22, 4.4], which generalizes a result of Robertson and Seymour [20, (8.3)].

Lemma 3.1. *Let V be a finite set and f be a symmetric submodular function on 2^V of branch-width at most k. If $\mathcal{P}$ is a titanic partition of width at most k with respect to f, then the branch-width of $f^{\mathcal{P}}$ is at most k.*

The purpose of this section is to show how a partitioned matroid may be "modeled" by an ordinary matroid having the same branch-width. We aim to transform a partitioned matroid $(M, \mathcal{P})$ to another partitioned matroid $(M^{\#}, \mathcal{P}^{\#})$, such that they have the same branch-width and $\mathcal{P}^{\#}$ is a titanic partition with respect to $\lambda_{M^{\#}}$.

It is easy to see that, in order to measure the branch-width of a partitioned matroid $(M, \mathcal{P})$, we may assume each part T of $\mathcal{P}$ satisfies $\lambda_M(T) = |T| + 1$ if $|T| > 1$. This means that $M \restriction T$ is a free matroid. For each part T of $(M, \mathcal{P})$, if $|T| > 1$, then we define a matroid U_T as a rank-$|T|$ uniform matroid on the ground set $E_T = E(U_T)$ such that $|E_T| = 3|T| - 2$, $E(M) \cap E_T = T$, and $E_T \cap E_{T'} = \emptyset$ if $T' \neq T$ is a part of $\mathcal{P}$ and $|T'| > 1$. Since $M \restriction T = U_Q \restriction T$ is a free matroid, an amalgam of M and U_T exists by Lemma 2.1. Moreover, it can be shown that the set $E(U_T)$ is always titanic in this amalgam.

Theorem 3.2. *Let $(M_0, \mathcal{P}_0)$ be a partitioned matroid and let $T_1, T_2, \ldots, T_m$ be the parts of $\mathcal{P}_0$ having at least two elements. Assume that $\lambda_{M_0}(T_i) = |T_i| + 1$ for every $i \in \{1, 2, \ldots, m\}$. For all $i = 1, 2, \ldots, m$, let M_i be an amalgam of M_{i-1} and U_{T_i}. Then the branch-width of M_m is equal to the branch-width of the partitioned matroid $(M_0, \mathcal{P}_0)$.*

We call resulting $M^{\#} = M_m$ the *normalized matroid* of $(M_0, \mathcal{P}_0)$.

4 Branch-decompositions of Represented Partitioned Matroids

We now specialize the above ideas to the case of representable matroids. We aim to provide an efficient algorithm for testing small branch-width on such

matroids. For the rest of our paper, a *represented matroid* is the vector matroid of a (given) matrix over a *fixed* finite field. We also write $\mathbb{F}$-*represented* matroid to explicitly refer to the field $\mathbb{F}$. In other words, an $\mathbb{F}$-represented matroid is a set of points (a *point configuration*) in a (finite) projective geometry over $\mathbb{F}$.

Not all matroids are representable over $\mathbb{F}$. Particularly, in the construction of the normalized matroid (Theorem 3.2) we apply amalgams with (uniform) matroids which need not be $\mathbb{F}$-representable. To achieve their representability, we extend the field $\mathbb{F}$ to an *extension field* $\mathbb{F}' = \mathbb{F}(\alpha)$ with $|\mathbb{F}|^d$ elements in the standard algebraic way with a polynomial root α of degree d.

Lemma 4.1. *The n-element rank-r uniform matroid $U_{r,n}$ is representable over any (finite) field $\mathbb{F}$ such that $|\mathbb{F}| \geq n - 1$.*

Proposition 4.2 (cf. Lemma 2.1). *Let M_1, M_2 be two matroids such that $E(M_1) \cap E(M_2) = T$ and $M_1 \restriction T = M_2 \restriction T$. If both M_1, M_2 are $\mathbb{F}$-represented, and the matroid $M_1 \restriction T$ is free, then there exists an amalgam of M_1 and M_2 which is also $\mathbb{F}$-represented.*

Let k be a fixed integer now. We outline a simple fixed-parameter-tractable algorithm for testing branch-width $\leq k$ on $\mathbb{F}$-represented partitioned matroids:

- First we extend $\mathbb{F}$ to a (nearest) field $\mathbb{F}'$ such that $|\mathbb{F}'| \geq 3k - 6$.
- If, for a given partitioned matroid $(M, \mathcal{P})$, the width of $\mathcal{P}$ is more than k, then the immediate answer is NO.
- Otherwise, we construct the normalized matroid $M^{\#}$ (Theorem 3.2), together with its vector representation over $\mathbb{F}'$ (Lemma 4.1 and Proposition 4.2).
- Finally, we use the algorithm of Hliněný [14] to test the branch-width $\leq k$ of $M^{\#}$.

Hence we conclude:

Theorem 4.3. *Let $k > 1$ be fixed and $\mathbb{F}$ be a finite field. For a partitioned matroid $(M, \mathcal{P})$ represented over $\mathbb{F}$, one can test in time $O(|E(M)|^3)$ (with fixed $k, \mathbb{F}$) whether the branch-width of $(M, \mathcal{P})$ is at most k.*

Now that we are able to test branch-width of partitioned matroids, we show how this result can be extended to finding an appropriate branch-decomposition, which was not known before.

Theorem 4.4. *Let $\mathcal{K}$ be a class of matroids and let k be an integer. If there is an $f(|E(M)|, k)$-time algorithm to decide whether a partitioned matroid $(M, \mathcal{P})$ has branch-width at most k for every pair of a matroid $M \in \mathcal{K}$ and a partition $\mathcal{P}$ of $E(M)$, then a branch-decomposition of the partitioned matroid $(M, \mathcal{P})$ of width at most k, if it exists, can be found in time $O\big(|\mathcal{P}|^3 \cdot f(|E(M)|, k)\big)$.*

The idea of the proof is due to Jim Geelen, published by Oum and Seymour in [13]. We briefly outline the algorithm since it is a base for our improved algorithm in the next section.

– If $|\mathcal{P}| \leq 2$, then it is trivial to output a branch-decomposition.
– We find a pair X, Y of disjoint parts of $\mathcal{P}$ such that a partitioned matroid $(M, (\mathcal{P} \setminus \{X, Y\}) \cup \{X \cup Y\})$ has branch-width at most k. Let $\mathcal{P}' = (\mathcal{P} \setminus \{X, Y\}) \cup \{X \cup Y\}$.
– Let (T', μ') be the branch-decomposition of $(M, \mathcal{P}')$ of width at most k obtained by calling this algorithm recursively.
– Let T be a tree obtained from T' by splitting the leaf $\mu'(X \cup Y)$ into two leaves which we denote by $\mu(X)$ and $\mu(Y)$. Let $\mu(Z) = \mu'(Z)$ for all $Z \in \mathcal{P} \setminus \{X, Y\}$. We output (T, μ) as a branch-decomposition of (M, P) of width at most k.

Corollary 4.5. *For fixed k and finite field $\mathbb{F}$, we can find a branch-decomposition of a given $\mathbb{F}$-represented matroid M of branch-width at most k, if it exists, in time $O(|E(M)|^6)$.*

Remark 4.6. One can actually improve the bound in Theorem 4.4 to $O\big(|\mathcal{P}|^2 \cdot f(|E(M)|, k)\big)$ time. The basic idea is the following: At the first level of recursion we find not only one pair of parts, but a maximal set of disjoint pairs of parts from $\mathcal{P}$ that can be joined (pairwise) while keeping the branch-width at most k. At the deeper levels of recursion we then use the same approach but process only such pairs of parts that contain one joined at the previous level. The details of this approach can be found further in Theorem 5.2.

5 Faster Algorithm for Branch-decompositions

Even with Remark 4.6 in account, the approach of Section 4 results in an $O(n^5)$ (at best) parametrized algorithm for constructing a branch-decomposition of an n-element matroid represented over a finite field. That is still far from the running time $O(n^3)$ (note fixed k and $\mathbb{F}$) of the decision algorithm in [14]. Although not straightforwardly, we are able to improve the running time of our constructive algorithm to asymptotically match $O(n^3)$ of [14] and [12].

It is the purpose of this section to present a detailed analysis of such a faster implementation of the algorithmic idea of Theorem 4.4 in Algorithm 5.1. For that we have to dive into fine details of the algorithms in [14], and recall few necessary technical definitions here.

Briefly speaking, a *parse tree* [15] of an $\mathbb{F}$-represented matroid M is a rooted tree $\mathcal{T}$, with at most two children per node, such that: The leaves of $\mathcal{T}$ hold non-loop elements of M represented by points of a projective geometry over $\mathbb{F}$ (or loops of M represented by the empty set). The internal nodes of $\mathcal{T}$, on the other hand, hold *composition operators* over $\mathbb{F}$. A composition operator $\odot$ is a configuration in the projective geometry over $\mathbb{F}$ such that $\odot$ has three subspaces (possibly empty) distinguished as its *boundaries*; two of which are used to "glue" the matroid elements represented in the left and right subtrees, respectively, together. The third one, *upper boundary*, is then used to "glue" this node further up in the parse tree $\mathcal{T}$. (Our "glue" operation, precisely the boundary sum by [15], is analogous to the amalgam of matroids in Proposition 4.2.) The ranks of adjacent boundaries of two composition operators in $\mathcal{T}$ must be equal for

"gluing". A parse tree $\mathcal{T}$ is $\leq t$-*boundaried* if all composition operators in $\mathcal{T}$ have boundaries of rank at most t. (Such a parse tree actually gives a branch-decomposition of width at most $t+1$ and vice versa.)

Algorithm 5.1. Computing a branch-decomposition of a represented partitioned matroid:

Parameters: A finite field $\mathbb{F}$, and a positive integer k.
Input: A rank-r matrix $\boldsymbol{A} \in \mathbb{F}^{r\times n}$ and a partition $\mathcal{P}$ of the columns of $\boldsymbol{A}$. (Assume $n \geq 2$.)
Output: For the vector matroid $M = M(\boldsymbol{A})$ on the columns of $\boldsymbol{A}$, either a branch-decomposition of the partitioned matroid $(M, \mathcal{P})$ of width at most k, or the answer NO if $\mathrm{bw}(M, \mathcal{P}) > k$.

1. Using brute force, we extend the field $\mathbb{F}$ to a (nearest) finite field $\mathbb{F}'$ such that $|\mathbb{F}'| \geq 3k-6$.
2. We check whether $\mathrm{bw}(M, \mathcal{P}) \leq k$ (Theorem 4.3, in cubic time). If not, then we answer NO. Otherwise we keep the normalized matroid $M^{\#}$ and its $\mathbb{F}'$-representation $\boldsymbol{A}^{\#}$ obtained at this step. We denote by $\mathcal{P}_1$ the (titanic) partition of $E(M^{\#})$ corresponding with $\mathcal{P}$, and by $\tau(T) \in \mathcal{P}$ for $T \in \mathcal{P}_1$ the corresponding parts.
3. We compute a $\leq 3(k-1)$-boundaried parse tree $\mathcal{T}$ for the matroid $M^{\#}$ which is $\mathbb{F}'$-represented by $\boldsymbol{A}^{\#}$ (regardless of $\mathcal{P}_1$). This is done by [14, Algorithm 4.1] in cubic time.
4. We initially set $\mathcal{T}_1 := \mathcal{T}$, $\mathcal{Q}_1 := \emptyset$, $\mathcal{Q}_2 := \big\{\{T_1, T_2\} : T_1 \neq T_2,\ T_1, T_2 \in \mathcal{P}_1\big\}$, and create a new rooted forest D consisting so far of the set of disconnected nodes $\mathcal{P}_1$. Then we repeat the following steps (a),(b), until $\mathcal{P}_1$ contains at most two parts:
   (a) While there is $\{T_1, T_2\} \in \mathcal{Q}_2$ such that $T_1, T_2 \in \mathcal{P}_1$, we do:
      i. Let $\mathcal{Q}_2 := \mathcal{Q}_2 \setminus \big\{\{T_1, T_2\}\big\}$. Calling [14, Algorithm 4.9] in linear time, we compute connectivity $\ell = \lambda_{M_1}(T_1 \cup T_2)$ over the parse tree $\mathcal{T}_1$ which now represents a matroid M_1. If $\ell > k$, then we continue this cycle again from (a).
      ii. We call Algorithm 5.3 on $\mathcal{T}_1$ and $W = T_1 \cup T_2$ to compute a $\leq(3k+\ell-2)$-boundaried parse tree $\mathcal{T}_2$. The matroid M_2 of $\mathcal{T}_2$ is actually a represented amalgam (Proposition 4.2) of our *titanic gadget* — a uniform matroid $U_W \simeq U_{\ell-1,3\ell-5}$, with the matroid M_1 (formally replacing W with an $(\ell-1)$-element free sub-matroid of U_W).
      iii. We can immediately check whether branch-width $\mathrm{bw}(M_2) \leq k$ by applying [14, Corollary 5.4] on $\mathcal{T}_2$, that is by linear-time testing of the (finitely many by [23]) excluded minors for branch-width at most k. If $\mathrm{bw}(M_2) > k$, then we continue this cycle again from (a).
      iv. So (Lemma 3.1) we have $\mathrm{bw}\big(M_1, \mathcal{P}_1 \cup \{W\} \setminus \{T_1, T_2\}\big) = \mathrm{bw}(M_2) \leq k$, and we add a *new node* $E(U_W)$ adjacent to T_1 and T_2 in our constructed decomposition D and make the new node the root for its connected component. We update $\mathcal{P}_1 := \mathcal{P}_2 = \mathcal{P}_1 \cup \{E(U_W)\} \setminus \{T_1, T_2\}$, and $\mathcal{Q}_1 := \mathcal{Q}_1 \cup \{E(U_W)\}$.
      v. Lastly, by calling [14, Algorithm 4.1.3] on $\mathcal{T}_2$, we compute in quadratic time a *new* $\leq 3(k-1)$-*boundaried parse tree* $\mathcal{T}_3$ for the matroid M_2, and set $\mathcal{T}_1 := \mathcal{T}_3$.
   (b) When the "while" cycle (4.a) is finished, we set $\mathcal{Q}_2 := \big\{\{T_1, T_2\} : T_1 \neq T_2,\ T_1 \in \mathcal{P}_1,\ T_2 \in \mathcal{Q}_1\big\}$ and $\mathcal{Q}_1 := \emptyset$, and continue with (4.).

5. Finally, if $|\mathcal{P}_1| = 2$, then we connect by an edge in D the two nodes $T_1, T_2 \in \mathcal{P}_1$. We output (D, τ) as the branch-decomposition of $(M, \mathcal{P})$.

Theorem 5.2. *Let k be a fixed integer and $\mathbb{F}$ be a fixed finite field. We assume that a vector matroid $M = M(\boldsymbol{A})$ is given as an input together with a partition $\mathcal{P}$ of $E(M)$, where $n = |E(M)|$ and $|\mathcal{P}| \geq 2$. Algorithm 5.1 outputs in time $O(n^3)$ (parametrized by k and $\mathbb{F}$), a branch-decomposition of the partitioned matroid $(M, \mathcal{P})$ of width at most k, or confirms that* $\mathrm{bw}(M, \mathcal{P}) > k$.

Note that Algorithm 5.1 implements the general outline of Theorem 4.4. Our proof of this theorem constitutes the following four claims holding true if $\mathrm{bw}(M, \mathcal{P}) \leq k$.

(I) The computation of Algorithm 5.1 maintains invariants, with respect to the actual matroid M_2 of $\mathcal{T}_2$, the decomposition D and current value $\mathcal{P}_2$ of the partition variable $\mathcal{P}_1$ after each call to step (4.a.iv), that
   - $\mathcal{P}_2$ is the set of roots of D, and a titanic partition of M_2 such that $\mathrm{bw}(M_2, \mathcal{P}_2) = \mathrm{bw}(M_2) \leq k$,
   - $\lambda_M\big(\tau^D(\mathcal{S})\big) = \lambda_{M_2}^{\mathcal{P}_2}(\mathcal{S})$ for each $\mathcal{S} \subseteq \mathcal{P}_2$, where $\tau^D(\mathcal{S})$ is a shortcut for the union of $\tau(T)$ with T running over all leaves of the connected components of D whose root is in $\mathcal{S}$ (see Algorithm 5.1 step 2. for τ).

(II) Each iteration of the main cycle in Algorithm 5.1 (4.) succeeds to step (4.a.iv) at least once.

(III) The main cycle in Algorithm 5.1 (4.) is repeated $O(n)$ times. Moreover, the total number of calls to the steps in (4.a) is: $O(n^2)$ for steps i,ii,iii, and $O(n)$ for steps iv,v.

(IV) Further defined Algorithm 5.3 (cf. step (4.a.ii)) computes correctly in time $O(n)$.

Having all these facts at hand, it is now easy to finish the proof. It is immediate from (I) that resulting (D, τ) is a branch-decomposition of width at most k of $(M, \mathcal{P})$. Note that all parse trees involved in the algorithm have constant width less than $4k$ (see in steps (4.a.ii,v)). The starting steps (1.),(2.),(3.) of the algorithm are already known to run in time $O(n^3)$ (Hliněný [14] and Theorem 4.3), and the particular steps in (4.a) need time $O(n^2) \cdot O(n) + O(n) \cdot O(n^2) = O(n^3)$ by (III) and (IV).

We are left with (IV), an immediate extension of [14, Algorithm 4.9] computing $\lambda_{M_1}(W)$.

Algorithm 5.3. Computing an amalgam with a uniform matroid on the parse tree.

Input: A $\leq(3k-1)$-boundaried parse tree $\mathcal{T}_1$ of a matroid M_1, and a set $W \subseteq E(M_1)$ such that $\lambda_{M_1}(W) = \ell \leq k$.

Output: A $\leq(3k+\ell-2)$-boundaried parse tree $\mathcal{T}_2$ representing a matroid M_2 on the ground set $E(M_2) = E_1 \cup E_2$ where $E_1 = E(M_1) \setminus W$ and $E_2 \cap E(M_1) = \emptyset$; such that $M_2 \restriction E_1 = M_1 \restriction E_1$, $M_2 \restriction E_2 = U_W \simeq U_{\ell-1,3\ell-5}$, and the set $E_2 = E(U_W)$ is spanned both by E_1 and by W in the (combined) point configuration $M_1 \cup M_2$.

6 Finding a Rank-Decomposition of a Graph

In this last section, we present a fixed-parameter-tractable algorithm to find a rank-decomposition of width at most k or confirm that the input graph has rank-width larger than k. It is a direct translation of the algorithm of Theorem 5.2. Let us first review necessary definitions. We assume that all graphs in this section have no loops and no parallel edges.

We have seen in Section 2 that every symmetric submodular function can be used to define branch-width. We define a symmetric submodular function on a graph, called the *cut-rank* function of a graph. For an $X \times Y$ matrix $\boldsymbol{R}$ and $A \subseteq X$, $B \subseteq Y$, let $\boldsymbol{R}[A, B]$ be the $A \times B$ submatrix of $\boldsymbol{R}$. For a graph G, let $\boldsymbol{A}(G)$ be the adjacency matrix of G, that is a $V \times V$ matrix over the binary field GF(2) such that an entry is 1 if and only if vertices corresponding to the column and the row are adjacent in G. The cut-rank function $\rho_G(X)$ of a graph $G = (V, E)$ is defined as the rank of the matrix $\boldsymbol{A}(G)[X, V \setminus X]$ for each subset X of V. Then ρ_G is symmetric and submodular, see [9]. *Rank-decomposition* and *rank-width* and of a graph G is branch-decomposition and branch-width of the cut-rank function ρ_G of the graph G, respectively. So if the graph has at least two vertices, then the rank-width is at most k if and only if there is a rank-decomposition of width at most k.

Now let us recall why bipartite graphs are essentially binary matroids. Oum [11] showed that the connectivity function of a binary matroid is exactly one more than the cut-rank function of its *fundamental graph*. The fundamental graph of a binary matroid M on $E = E(M)$ with respect to a basis B is a bipartite graph on E such that two vertices in E are adjacent if and only if one vertex v is in B, another vertex w is not in B, and $(B \setminus \{v\}) \cup \{w\}$ is independent in M. Given a bipartite graph G, we can easily construct a binary matroid having G as a fundamental graph; if (C, D) is a bipartition of $V(G)$, then take the matrix

$$\left(\begin{array}{ccc|c} 1 & & 0 & \boldsymbol{A}(G)[C,D] \\ & \ddots & & \\ 0 & & 1 & \begin{array}{c} C \times D \text{ submatrix of} \\ \text{the adjacency matrix} \end{array} \end{array}\right)$$

as the representation of a binary matroid. (Thus the column indices are elements of the binary matroid and a set of columns is independent in the matroid if and only if its vectors are linearly independent.) After all, finding the rank-decomposition of a bipartite graph is equivalent to finding the branch-decomposition of the associated binary matroid, that is essentially Theorem 5.2.

To find a rank-decomposition of non-bipartite graphs, we transform the graph into a canonical bipartite graph. For a finite set V, let V^* be a disjoint copy of V, that is, formally speaking, $V^* = \{v^* : v \in V\}$ such that $v^* \neq w$ for all $w \in V$ and $v^* \neq w^*$ for all $w \in V \setminus \{v\}$. For a subset X of V, let $X^* = \{v^* : v \in X\}$. For a graph $G = (V, E)$, let $\mathrm{bip}(G)$ be the bipartite graph on $V \cup V^*$ such that vw^* are adjacent in $\mathrm{bip}(G)$ if and only if v and w are adjacent in G. Let $P_v = \{v, v^*\}$ for each $v \in V$. Then $\Pi(G) = \{P_v : v \in V\}$ is a canonical partition of $V(\mathrm{bip}(G))$.

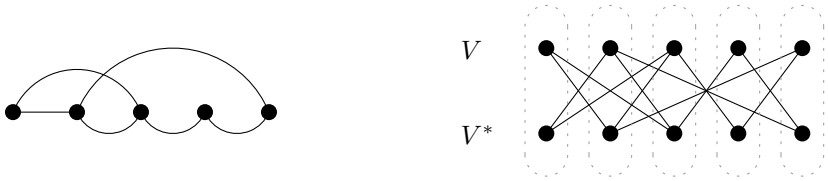

Fig. 2. Graph G and the associated bipartite graph $\mathrm{bip}(G)$ with its canonical partition.

Lemma 6.1. *For every subset X of $V(G)$, $2\rho_G(X) = \rho_{\mathrm{bip}(G)}(X \cup X^*)$.*

Corollary 6.2. *Let $p : V(G) \to \Pi(G)$ be the bijective function such that $p(x) = P_x$. If (T, μ) is a branch-decomposition of $\rho^{\Pi(G)}_{\mathrm{bip}(G)}$ of width k, then $(T, \mu \circ p)$ is a branch-decomposition of ρ_G of width $k/2$. Conversely, if (T, μ') is a branch-decomposition of ρ_G of width k, then $(T, \mu' \circ p^{-1})$ is a branch-decomposition of $\rho^{\Pi(G)}_{\mathrm{bip}(G)}$ of width $2k$. Therefore the branch-width of ρ_G is equal to the half of the branch-width of $\rho^{\Pi(G)}_{\mathrm{bip}(G)}$.*

Let $M = \mathrm{mat}(G)$ be the binary matroid on $V \cup V^*$ represented by the matrix:

$$\begin{array}{c} \\ V \end{array} \begin{array}{c} \begin{array}{cc} V & V^* \end{array} \\ \left(\begin{array}{c|c} \text{Identity matrix} & \boldsymbol{A}(G) \end{array}\right) \end{array}.$$

Since the bipartite graph $\mathrm{bip}(G)$ is a fundamental graph of M, we have $\lambda_M(X) = \rho_{\mathrm{bip}(G)}(X)+1$ for all $X \subseteq V \cup V^*$ (see Oum [11]) and therefore (T, μ) is a branch-decomposition of a partitioned matroid $(M, \Pi(G))$ of width $k+1$ if and only if it is a branch-decomposition of $\rho^{\Pi(G)}_{\mathrm{bip}(G)}$ of width k. Corollary 6.2 implies that a branch-decomposition of $\rho^{\Pi(G)}_{\mathrm{bip}(G)}$ of width k is equivalent to that of ρ_G of width $k/2$. So we can deduce the following theorem from Theorem 5.2.

Theorem 6.3. *Let k be a constant. Let $n \geq 2$. For an n-vertex graph G, we can output the rank-decomposition of width at most k or confirm that the rank-width of G is larger than k in time $O(n^3)$.*

Acknowledgments The authors are very grateful to Jim Geelen for his comments on the possible approach to the problem.

References

1. Courcelle, B., Olariu, S.: Upper bounds to the clique width of graphs. Discrete Appl. Math. **101**(1-3) (2000) 77–114
2. Courcelle, B., Makowsky, J.A., Rotics, U.: Linear time solvable optimization problems on graphs of bounded clique-width. Theory Comput. Syst. **33**(2) (2000) 125–150

3. Wanke, E.: k-NLC graphs and polynomial algorithms. Discrete Appl. Math. **54**(2-3) (1994) 251–266
4. Espelage, W., Gurski, F., Wanke, E.: How to solve NP-hard graph problems on clique-width bounded graphs in polynomial time. In: Graph-theoretic concepts in computer science (Boltenhagen, 2001). Volume 2204 of Lecture Notes in Comput. Sci., Berlin, Springer (2001) 117–128
5. Kobler, D., Rotics, U.: Edge dominating set and colorings on graphs with fixed clique-width. Discrete Appl. Math. **126**(2-3) (2003) 197–221
6. Gerber, M.U., Kobler, D.: Algorithms for vertex-partitioning problems on graphs with fixed clique-width. Theoret. Comput. Sci. **299**(1-3) (2003) 719–734
7. Corneil, D.G., Perl, Y., Stewart, L.K.: A linear recognition algorithm for cographs. SIAM J. Comput. **14**(4) (1985) 926–934
8. Corneil, D.G., Habib, M., Lanlignel, J.M., Reed, B., Rotics, U.: Polynomial time recognition of clique-width ≤ 3 graphs (extended abstract). In: Gonnet, Gastón H. (ed.) et al., LATIN 2000: Theoretical informatics. 4th Latin American symposium, Punta del Este, Uruguay, April 10-14, 2000. Volume 1776 of Lecture Notes in Comput. Sci. Springer, Berlin (2000) 126–134
9. Oum, S., Seymour, P.: Approximating clique-width and branch-width. J. Combin. Theory Ser. B **96**(4) (2006) 514–528
10. Oum, S.: Approximating rank-width and clique-width quickly. Submitted, an extended abstract appeared in [24] (2006)
11. Oum, S.: Rank-width and vertex-minors. J. Combin. Theory Ser. B **95**(1) (2005) 79–100
12. Courcelle, B., Oum, S.: Vertex-minors, monadic second-order logic, and a conjecture by Seese. J. Combin. Theory Ser. B **97**(1) (2007) 91–126
13. Oum, S., Seymour, P.: Testing branch-width. J. Combin. Theory Ser. B **97**(3) (2007) 385–393
14. Hliněný, P.: A parametrized algorithm for matroid branch-width. SIAM J. Comput. **35**(2) (2005) 259–277, loose erratum (electronic)
15. Hliněný, P.: Branch-width, parse trees, and monadic second-order logic for matroids. J. Combin. Theory Ser. B **96**(3) (2006) 325–351
16. Fellows, M.R., Rosamond, F.A., Rotics, U., Szeider, S.: Clique-width minimization is NP-hard. In: Proceedings of the 38th annual ACM Symposium on Theory of Computing, ACM Press New York, NY, USA (2006) 354–362
17. Hicks, I.V., McMurray Jr., N.B.: The branchwidth of graphs and their cycle matroids. J. Combin. Theory Ser. B (2007) doi:10.1016/j.jctb.2006.12.007.
18. Mazoit, F., Thomassé, S.: Branchwidth of graphic matroids. Manuscript (2005)
19. Seymour, P., Thomas, R.: Call routing and the ratcatcher. Combinatorica **14**(2) (1994) 217–241
20. Robertson, N., Seymour, P.: Graph minors. X. Obstructions to tree-decomposition. J. Combin. Theory Ser. B **52**(2) (1991) 153–190
21. Oxley, J.G.: Matroid theory. Oxford University Press, New York (1992)
22. Geelen, J.F., Gerards, A.M.H., Whittle, G.: Tangles, tree-decompositions, and grids in matroids. Research Report 04-5, School of Mathematical and Computing Sciences, Victoria University of Wellington (2004)
23. Geelen, J.F., Gerards, A.M.H., Robertson, N., Whittle, G.: On the excluded minors for the matroids of branch-width k. J. Combin. Theory Ser. B **88**(2) (2003) 261–265
24. Oum, S.: Approximating rank-width and clique-width quickly. In: Graph-theoretic concepts in computer science (Metz, 2005). Volume 3787 of Lecture Notes in Comput. Sci. Springer (2005) 49–58

Appendix: some proofs

Theorem 3.2, proof:

Lemma A.4. *Let M be a matroid and T be a subset of $E(M)$ such that $\lambda_M(T) = |T| + 1$. Let M' be an amalgam of M and U_T. Then the following are true:*

(1) If $T \subseteq X \subseteq E(M')$, then $\mathrm{r}_M(X \cap E(M)) = \mathrm{r}_{M'}(X)$.
(2) $\lambda_M(X) = \lambda_{M'}(X)$ for all $X \subseteq E(M) \setminus T$.
(3) The set $E(U_T)$ is titanic in the matroid M'.

Proof (Lemma A.4). (1) Because $M' \restriction E(M) = M$, we have $\mathrm{r}_M(X \cap E(M)) = \mathrm{r}_{M'}(X \cap E(M)) \leq \mathrm{r}_{M'}(X)$.

Since T is independent in U_T, we can pick a maximally independent subset I of X in M' such that $T \subseteq I$. Since $M' \restriction E(U_T) = U_T$, the set $I \cap E(U_T)$ is independent in U_T and therefore $I \cap E(U_T) = T$. So $I \subseteq E(M)$. Therefore $\mathrm{r}_M(X \cap E(M)) \geq |I| = \mathrm{r}_{M'}(X)$.

(2) Let $Y = E(M') \setminus X$. We note that $E(U_T)$ is a subset of Y. By definition,

$$\lambda_M(X) = \mathrm{r}_M(X) + \mathrm{r}_M(Y \cap E(M)) - \mathrm{r}(M) + 1,$$
$$\lambda_{M'}(X) = \mathrm{r}_{M'}(X) + \mathrm{r}_{M'}(Y) - \mathrm{r}(M') + 1.$$

Since $M' \restriction E(M) = M$, we have $\mathrm{r}_M(X) = \mathrm{r}_{M'}(X)$. By (1), $\mathrm{r}_M(Y \cap E(M)) = \mathrm{r}_{M'}(Y)$ and $\mathrm{r}(M') = \mathrm{r}(M)$. Thus $\lambda_M(X) = \lambda_{M'}(X)$.

(3) We claim that if X is a subset of $E(U_T)$ and $|X| \geq |T|$, then $\lambda_{M'}(X) \geq \lambda_{M'}(E(U_T))$. Since U_T is a uniform matroid of rank $|T|$, we have

$$\mathrm{r}_{M'}(X) = |T| = \mathrm{r}_{M'}(E(U_T)),$$
$$\mathrm{r}_{M'}(E(M') \setminus X) \geq \mathrm{r}_{M'}(E(M') \setminus E(U_T)).$$

Therefore, $\lambda_{M'}(X) \geq \lambda_{M'}(E(U_T))$.

Now suppose that X_1, X_2, X_3 are pairwise disjoint subsets of $E(U_T)$. Then there is $i \in \{1, 2, 3\}$ such that $|X_i| \geq \lceil |E(U_T)|/3 \rceil = |T|$ and therefore $\lambda_{M'}(X_i) \geq \lambda_{M'}(E(U_T))$. So $E(U_T)$ is titanic in M'. □

Now let us prove Theorem 3.2. Let $\mathcal{P}_i = (\mathcal{P}_0 \setminus \{T_i\}) \cup \{E(U_{T_i})\}$. By Lemma A.4 (2), the branch-width of $(M_i, \mathcal{P}_i)$ is equal to that of $(M_{i-1}, \mathcal{P}_{i-1})$ and therefore the branch-width of $(M_m, \mathcal{P}_m)$ is equal to the branch-width of $(M_0, \mathcal{P}_0)$. By Lemma A.4 (3), $\mathcal{P}_m$ is a titanic partition. Let k be the branch-width of M_m. It is easy to see that the branch-width of the uniform matroid U_{T_i} is $|T_i|+1 = \lambda_{M_m}(E(U_{T_i}))$. Since U_{T_i} is a minor of M_m, the branch-width of M_m is at least $|T_i|+1$ for all i and therefore the width of $\mathcal{P}_m$ is at most k. We conclude that the branch-width of $(M_m, \mathcal{P}_m)$ is at most k by Lemma 3.1.

To finish the proof, we need to show that the branch-width of $(M_m, \mathcal{P}_m)$ is at least k. Let (T, μ) be the branch-decomposition of $(M_m, \mathcal{P}_m)$ of width at most k. From (T, μ), we would like to obtain a branch-decomposition (T', μ') of M_m

whose width is at most k as follows. We first prepare branch-decompositions (T_i, μ_i) of width at most k for each i. For each leaf of T corresponding to $E(U_i)$, we attach T_i by subdividing one edge of T_i and join the new vertex to the leaf of T. Let T' be the new tree obtained by the above process for all i. The bijection μ' from V to leaves of T' is easily obtained from μ_i. Since $\lambda_{M_m}(X) = |X| + 1$ for all $X \subseteq E(U_{T_i})$, the width of (T', μ') is at most k. □

Theorem 4.4, proof:

At each level of recursion, we call the decision algorithm at most $\binom{|\mathcal{P}|}{2} = O(|\mathcal{P}|^2)$ times. The depth of recursion is $|\mathcal{P}| - 1$, and therefore the number of calls to the decision algorithm is at most $O(|\mathcal{P}|^3)$. Thus, the running time of the algorithm is $O\big(|\mathcal{P}|^3 \cdot f(|E(M)|, k)\big)$.

Now it remains to show that the algorithm is indeed correct. It is obvious that in every subcubic tree with at least three leaves, there are two leaves that have a common neighbor. Suppose that (T, μ) is a branch-decomposition of $(M, \mathcal{P})$ of width at most k. Then there are two leaves $\mu(X)$ and $\mu(Y)$ having a common neighbor z in T. It is easy to see that if we remove $\mu(X)$ and $\mu(Y)$ from T and map $X \cup Y$ to z by μ, then (T, μ) is a branch-decomposition of $(M, \mathcal{P}')$ of width at most k. Therefore the branch-width of $(M, \mathcal{P}')$ is at most k.

Conversely, suppose that (T', μ') is the branch-decomposition of $(M, \mathcal{P}')$. Since $(M, \mathcal{P})$ has branch-width at most k, we know that $\lambda_M(X) \leq k$ and $\lambda_M(Y) \leq k$. Thus (T, μ) is a branch-decomposition of $(M, \mathcal{P})$ of width at most k. □

Theorem 5.2, proof:

The proof of (I) essentially extends arguments of Theorem 4.4. Initially, with M_1 and $\mathcal{P}_1$ in place of M_2, $\mathcal{P}_2$, all the claims of (I) obviously hold true, analogously to Theorem 4.3. Each call to step (4.a.iv) then adds a new titanic (see Section 3) set $E(U_W)$ to $\mathcal{P}_2$, and hence the partition $\mathcal{P}_2$ remains titanic for M_2 and, subsequently, $\mathrm{bw}(M_2, \mathcal{P}_2) = \mathrm{bw}\big(M_1,\ \mathcal{P}_1 \cup \{W\} \setminus \{T_1, T_2\}\big) = \mathrm{bw}(M_2) \leq k$ follows from Lemma 3.1. The most complex claim of (I) is the last assertion, that $\lambda_M\big(\tau^D(\mathcal{S})\big) = \lambda^{\mathcal{P}_2}_{M_2}(\mathcal{S})$ for each $\mathcal{S} \subseteq \mathcal{P}_2$. By induction, we may assume that $\lambda_M\big(\tau^D(\mathcal{S}_1)\big) = \lambda^{\mathcal{P}_1}_{M_1}(\mathcal{S}_1)$ holds for all $\mathcal{S}_1 \subseteq \mathcal{P}_1$ just before this call to (4.a.iv). Now, by Algorithm 5.3, the titanic gadget $E(U_W)$ in the representation spans exactly the same subspace as it is the guts of the separation given by $W = T_1 \cup T_2$ in M_1. Therefore, for all $\mathcal{S}_1 \subseteq \mathcal{P}_1$ such that $|\mathcal{S}_1 \cap \{T_1, T_2\}| \neq 1$, the corresponding $\mathcal{S} \subseteq \mathcal{P}_2$ satisfies $\lambda^{\mathcal{P}_2}_{M_2}(\mathcal{S}) = \lambda^{\mathcal{P}_1}_{M_1}(\mathcal{S}_1)$. This proves the assertion.

To prove (II), we use that $\mathrm{bw}(M_1, \mathcal{P}_1) \leq k$ at each iteration of the main cycle (4.), which directly follows from above $\mathrm{bw}(M_2, \mathcal{P}_2) \leq k$. Then, by the same arguments as in Theorem 4.4, there is a pair $\{T_1, T_2\} \subset \mathcal{P}_1$ for which (4.a) would succeed up to step (4.a.iv) (which happens if $\mathrm{bw}\big(M_1, \mathcal{P}_1 \cup \{T_1 \cup T_2\} \setminus \{T_1, T_2\}\big) \leq k$). We call such a pair T_1, T_2 *admissible*. It remains to argue that all admissible pairs $\{T_1, T_2\} \subset \mathcal{P}_1$ belong also to $\mathcal{Q}_2$, which is trivial only during the first round of (4.). For a contradiction, assume that $\{T_1, T_2\} \notin \mathcal{Q}_2$ at the least round $i > 1$. Consider now the values of our variables $\mathcal{P}_1, \mathcal{Q}_1, \mathcal{Q}_2$ at the previous round $i - 1$:

It was $\{T_1, T_2\} \cap \mathcal{Q}_1 = \emptyset$ by (4.b), and so $\{T_1, T_2\} \subset \mathcal{P}_1$ already at round $i-1$. That means the pair T_1, T_2 has been admissible all time since round $i-1$, but it has not been processed only due to $\{T_1, T_2\} \notin \mathcal{Q}_2$ at round $i-1$. That contradicts our least choice of i.

Concerning (III), each iteration of (4.) adds at least one new node to the decomposition D by (II), and hence no more than $O(n)$ iterations occur. The precisely same argument also bounds the total number of calls to the crucial steps (4.a.iv–v). The situation with steps i,ii,iii is more versatile, and we bound the total number of calls to them from above by the total number of iterations of the cycle in (4.a): During the initial round of the main cycle (4.), there are clearly at most $|\mathcal{Q}_2| = O(n^2)$ iterations of (4.a). For each subsequent round $i > 1$, the number of iteration is at most $|\mathcal{Q}_2| \leq q_i \cdot |\mathcal{P}_1|$ where $q_i = |\mathcal{Q}_1|$ at the end of the previous run $i-1$. Hence the total number of iterations of the cycle in (4.a) is at most $O(n^2) + O(n) \cdot \sum_{i=2}^{r} q_i$ since always $|\mathcal{P}_1| = O(n)$. It remains to argue that $\sum_{i=2}^{r} q_i = O(n)$, which follows from the fact that each element ever assigned to $\mathcal{Q}_1$ in step (4.a.iv) appears as an internal node of the decomposition D, and $|V(D)| = O(n)$. □

Algorithm 5.3, body:

This algorithm is an immediate extension of [14, Algorithm 4.9] computing $\lambda_{M_1}(W)$. We describe it in terms of projective geometry and the point configuration representing M_1 by $\mathcal{T}_1$.

At the beginning we make $\mathcal{T}_2$ a copy of $\mathcal{T}_1$. The idea is to "enlarge" all the composition operators in $\mathcal{T}_2$ to fully contain the guts Γ (a projective subspace of rank $\ell - 1$) of the separation of W, and then to "glue" a (constant-size) decomposition of the uniform matroid $U_W \simeq U_{\ell-1,3\ell-5}$ to the root of $\mathcal{T}_2$ so that it spans Γ. We apply leaves-to-root dynamic programming on $\mathcal{T}_2$ with constant-time operations at each node, hence computing in time $O(n)$.

At node $x \in V(\mathcal{T}_2)$, we compute the subspace $\Sigma_x \subseteq \Gamma$ which is spanned by the elements of W held in the leaves below x. Knowing $\Sigma_{x'}$ and $\Sigma_{x''}$ for the children x', x'' of x in $\mathcal{T}_2$, it is a constant-time manipulation to determine Σ_x using the composition operator $\odot$ at x. (Notice that, as our algorithm is set up, Σ_x is also a subspace of $\odot$.) If the upper boundary of $\odot$ does not fully contain Σ_x, we enlarge it accordingly, and we also freely enlarge the matching boundary at the parent node of x. Note that $\Sigma_r = \Gamma$ will become the upper boundary of the root node r.

After finishing that, we take an arbitrary parse tree $\mathcal{T}_3$ of the uniform matroid $U_W \simeq U_{\ell-1,3\ell-5}$, and add to $\mathcal{T}_2$ a new root node r' adjacent to the former root of $\mathcal{T}_2$ and the root of $\mathcal{T}_3$. The composition operator at r' "glues" U_W directly to Σ_r. Finally, we strip from $\mathcal{T}_2$ all leaves holding the points of W. (This is trivial since our definition of a parse tree allows nodes with only one descendant.) □