Matrices of Optimal Tree-Depth and Row-Invariant Parameterized Algorithm for Integer Programming

Timothy F. N. Chan
School of Mathematical Sciences, Monash University, Melbourne, Australia
Mathematics Institute and DIMAP, University of Warwick, Coventry, UK
timothy.chan@monash.edu

Jacob W. Cooper
Faculty of Informatics, Masaryk University, Brno, Czech Republic
jcooper@mail.muni.cz

Martin Koutecký
Computer Science Institute, Charles University, Prague, Czech Republic
koutecky@iuuk.mff.cuni.cz

Daniel Král'
Faculty of Informatics, Masaryk University, Brno, Czech Republic
Mathematics Institute, DIMAP and Department of Computer Science,
University of Warwick, Coventry, UK
dkral@fi.muni.cz

Kristýna Pekárková
Faculty of Informatics, Masaryk University, Brno, Czech Republic
kristyna.pekarkova@mail.muni.cz

Abstract

A long line of research on fixed parameter tractability of integer programming culminated with showing that integer programs with n variables and a constraint matrix with tree-depth d and largest entry Δ are solvable in time $g(d, \Delta)\text{poly}(n)$ for some function g, i.e., fixed parameter tractable when parameterized by tree-depth d and Δ. However, the tree-depth of a constraint matrix depends on the positions of its non-zero entries and thus does not reflect its geometric structure. In particular, tree-depth of a constraint matrix is not preserved by row operations, i.e., a given integer program can be equivalent to another with a smaller dual tree-depth.

We prove that the *branch-depth* of the matroid defined by the columns of the constraint matrix is equal to the minimum tree-depth of a row-equivalent matrix. We also design a fixed parameter algorithm parameterized by an integer d and the entry complexity of an input matrix that either outputs a matrix with the smallest dual tree-depth that is row-equivalent to the input matrix or outputs that there is no matrix with dual tree-depth at most d that is row-equivalent to the input matrix. Finally, we use these results to obtain a fixed parameter algorithm for integer programming parameterized by the branch-depth of the input constraint matrix and the entry complexity. The parameterization by branch-depth cannot be replaced by the more permissive notion of branch-width.

2012 ACM Subject Classification Mathematics of computing → Combinatorial optimization; Mathematics of computing → Matroids and greedoids; Mathematics of computing → Combinatorial algorithms

Keywords and phrases Matroid algorithms, width parameters, integer programming, fixed parameter tractability, branch-width, branch-depth

Digital Object Identifier 10.4230/LIPIcs.ICALP.2020.26

Category Track A: Algorithms, Complexity and Games

Related Version A full version is available at `https://arxiv.org/abs/1907.06688`.

47th International Colloquium on Automata, Languages, and Programming (ICALP 2020).
Editors: Artur Czumaj, Anuj Dawar, and Emanuela Merelli; Article No. 26; pp. 26:1–26:19

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Funding The first, second and fourth authors were supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 648509). The second, fourth and fifth authors were supported by the MUNI Award in Science and Humanities of the Grant Agency of Masaryk university. The third author was supported by Charles University project UNCE/SCI/004, and by the project 19-27871X of GA ČR. This publication reflects only its authors' view; the European Research Council Executive Agency is not responsible for any use that may be made of the information it contains.

1 Introduction

Integer programming is a fundamental problem of importance in both theory and practice. It is well-known that integer programming in fixed dimension, i.e., with a bounded number of variables, is polynomially solvable since the work of Lenstra and Kannan [21, 26] from the 1980's. Much subsequent research has focused on studying extensions and speed-ups of the results of Kannan and Lenstra. However, on the side of integer programs with many variables, research has been sparser. Until relatively recently, the most prominent tractable case is that of totally unimodular constraint matrices, i.e., matrices with all subdeterminants equal to 0 and ± 1; in this case, all vertices of the feasible region are integral and algorithms for linear programming can be applied.

Besides total unimodularity, many recent results [1, 2, 5, 6, 9, 10, 14, 15] on algorithms for integer programming exploited various structural properties of the constraint matrix yielding efficient algorithms for n-fold IPs, tree-fold IPs, multi-stage stochastic IPs, and IPs with bounded fracture number and bounded tree-width. This research culminated with an algorithm by Levin, Onn and the third author [25] who constructed a fixed parameter algorithm for integer programs with bounded (primal or dual) tree-depth and bounded coefficients. We remark that it is possible to show that the problem is $\mathsf{W}[1]$-hard when parameterized by tree-depth only [10, 24] and NP-hard even for instances with coefficients and tree-width (even path-width) bounded by two [7, Lemma 102] (also cf. [10, 25]).

The tree-depth of a constraint matrix depends on the position of its non-zero entries and thus does not properly reflect the true geometric structure of the integer program. In particular, a matrix with a large (dual) tree-depth may be row-equivalent to another matrix with small (dual) tree-depth that is susceptible to efficient algorithms. We will overcome this drawback with tools from matroid theory. To do so, we consider the branch-depth of the matroid defined by the columns of the constraint matrix and refer to this parameter as to the *branch-depth* of the matrix. Since this matroid is invariant under row operations, the branch-depth of a matrix is *row-invariant*, i.e., preserved by row operations, and captures the true simplicity of the geometric structure of the problem, which can be obfuscated in the case of tree-depth by the choice of the basis.

Our main results can be summarized as follows (we state the results exactly in the next subsection).

- The branch-depth of a matrix A is equal to the minimum dual tree-depth of a matrix row-equivalent to A (Theorem 1).
- There exists a fixed parameter algorithm for computing a matrix with minimum tree-depth that is row-equivalent to an input matrix (Theorem 32).

Our second result is based on a fixed parameter algorithm for computing the branch-depth of a vector matroid represented over a finite field (Theorem 28). Computing decompositions of such matroids is of interest in relation to model checking, in particular monadic second order model checking is fixed parameter tractable for matroids with bounded branch-width

that are representable over finite fields [16–18], also see [11]. The existing fixed parameter algorithm [19, 20] for computing the branch-width of such matroids relies on the upper bound on the size of excluded minors for branch-width by Geelen et al. [12] and needs to precompute the (likely very large) list of excluded minors. We could follow a similar path in the setting of branch-depth, however, we decided to design a completely explicit algorithm. While this turned out surprisingly challenging, the hidden constants are significantly better, and we hope that our techniques can be extended to the even more challenging setting of branch-width.

We remark that our first result in conjunction with existing results on approximating branch-depth of a matroid (Theorem 6) yields a fixed parameter algorithm for integer programs with bounded branch-depth (Corollary 4). Since the branch-depth of the constraint matrix is always at most its dual tree-depth (Proposition 12), our algorithm extends the algorithm presented in [25] for integer programs with small dual tree-depth. Since the algorithm from our second result (Theorem 32) preserves that the entry complexity is bounded (cf. Theorem 3), it can also be used a preprocessing step for the algorithm presented in [25], which gives another proof of Corollary 4. Our results on fixed parameter tractability of integer programming cannot be extended to constraint matrices with bounded branch-width (see the discussion at the end of this section); however, Cunningham and Geelen [3] (also cf. [27] for detailed proofs and implementation) provided a slicewise pseudopolynomial algorithm for IPs with non-negative matrices with bounded branch-width, i.e., the problem belongs to the complexity class XP for unary encoding of input.

1.1 Exact statement of our results

To state our results precisely, we need to fix some notation. We consider the general integer programming (IP) problem in the standard form:

$$\min\{f(\mathbf{x}) \mid A\mathbf{x} = \mathbf{b}\,,\ \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\,,\ \mathbf{x} \in \mathbb{Z}^n\}\,, \tag{1}$$

where $A \in \mathbb{Z}^{m\times n}$ is an integer $m \times n$ matrix, $\mathbf{b} \in \mathbb{Z}^m$, $\mathbf{l}, \mathbf{u} \in (\mathbb{Z} \cup \{\pm\infty\})^n$, and $f : \mathbb{Z}^n \to \mathbb{Z}$ is a separable convex function, i.e., $f(\mathbf{x}) = \sum_{i=1}^{n} f_i(x_i)$ where $f_i : \mathbb{Z} \to \mathbb{Z}$ are convex functions. In particular, each $f_i(x_i)$ can be a linear function of x_i. We remark that integer programming is well-known to be NP-hard even when $f(\mathbf{x}) \equiv 0$, or when the largest coefficient $\Delta := \|A\|_\infty$ is 1 (by a reduction from the Vertex Cover problem), or when $m = 1$ (by a reduction from the Subset Sum problem). We refer the reader to Section 2 for the definitions of the primal and dual tree-depth of a matrix A and the branch-depth of A.

We now demonstrate the drawback of the parameterization of integer programs by tree-depth that we have mentioned earlier. Consider the following matrices A and A'.

$$A = \begin{pmatrix} 1 & 1 & \cdots & 1 & 1 \\ 2 & 1 & \cdots & 1 & 1 \\ 1 & 2 & \ddots & 1 & 1 \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 1 & 1 & \ddots & 2 & 1 \\ 1 & 1 & \cdots & 1 & 2 \end{pmatrix} \quad \text{and} \quad A' = \begin{pmatrix} 1 & 1 & \cdots & 1 & 1 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \ddots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix}.$$

The dual tree-depth of the matrix A is equal to the number of its rows while the dual tree-depth of A' is two (its dual graph is a star); we remark that the branch-depth of both matrices A and A' is also equal to two. Since the matrices A and A' are row-equivalent, the integer programs determined by them ought to be of the same computational difficulty. More precisely, consider the following matrix B:

$$B = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ -1 & 1 & 0 & \cdots & 0 & 0 \\ -1 & 0 & 1 & \ddots & 0 & 0 \\ -1 & \vdots & \ddots & \ddots & \ddots & 0 \\ -1 & 0 & 0 & \ddots & 1 & 0 \\ -1 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix}.$$

Since $A' = BA$, it is possible to replace an integer program of the form (1) with an integer program with a constraint matrix $A' = BA$, right hand side $\mathbf{b}' = B\mathbf{b}$, and bounds $\mathbf{l}' = \mathbf{l}$ and $\mathbf{u}' = \mathbf{u}$, and attempt to solve this new instance of IP which has dual tree-depth two.

In Section 4, we first observe that the branch-depth of a matrix A is at most its dual tree-depth, and prove that the branch-depth of a matrix A is actually equal to the minimum dual tree-depth of a matrix A that is row-equivalent to A:

▶ **Theorem 1.** *Let A be a matrix over a field $\mathbb{F}$. The branch-depth of A is equal to the minimum dual tree-depth of a matrix A' that is row-equivalent to A, i.e., that can be obtained from A by row operations.*

The tools developed to prove Theorem 1 together with existing results on matroid branch-depth yield an algorithm that given a matrix A of small branch-depth yields a matrix B that transforms the matrix A to a row-equivalent matrix with small dual tree-depth. Recall that the *entry complexity of a matrix A*, denoted by $\mathrm{ec}(A)$, is the maximum length of the binary encoding of an entry $A_{i,j}$ (the length of binary encoding a rational number $r = p/q$ with p and q being coprime is $\lceil \log_2 p \rceil + \lceil \log_2 q \rceil + 1$).

▶ **Theorem 2.** *There exist a computable function $g : \mathbb{N} \to \mathbb{N}$ and an FPT-parameter algorithm parameterized by d with running time polynomial in $\mathrm{ec}(A)$, n and m that for an input $m \times n$ integer matrix A and an integer d*

1. *outputs that the branch-depth of A is larger than d, or*
2. *outputs an invertible rational matrix $B \in \mathbb{Q}^{m \times m}$ such that the dual tree-depth of BA is at most 4^d and the entry complexity of BA is $O(g(d)\,\mathrm{ec}(A))$.*

However, we go further and design a fixed parameter algorithm for computing the branch-depth of a vector matroid (Theorem 32) and use this algorithm to prove the following strengthening of Theorem 2.

▶ **Theorem 3.** *There exist a computable function $g' : \mathbb{N}^2 \to \mathbb{N}$ and an FPT-parameter algorithm parameterized by d with running time polynomial in $\mathrm{ec}(A)$, n and m that for an input $m \times n$ integer matrix A and an integer d*

1. *outputs that the branch-depth of A is larger than d, or*
2. *outputs an invertible rational matrix $B \in \mathbb{Q}^{m \times m}$ such that the dual tree-depth of BA is equal to the branch-depth of A and the entry complexity of BA is at most $g'(d, \mathrm{ec}(A))$.*

As explained above, Theorems 2 and 3 allow us to perform row operations to obtain an equivalent integer program with small dual tree-depth from an integer program with small branch-depth. In particular, if the instance of an integer program described as in (1) has bounded branch-depth, then Theorem 3 yields a matrix B such that the instance with $A' = BA$, $\mathbf{b}' = B\mathbf{b}$, $\mathbf{l}' = \mathbf{l}$ and $\mathbf{u}' = \mathbf{u}$ has dual tree-depth equal to branch-depth. To apply the algorithm from [25], we need to transform the matrix A' into an integer matrix. We

do so by multiplying each row by the least common multiple of the denominators of the fractions in this row; note that the value of this least common multiple is at most $2^{2^{\mathrm{ec}(A')}}$ since there can be at most $2^{\mathrm{ec}(A')}$ different denominators appearing in the row. In particular, the entry complexity of the resulting integer matrix is bounded by a function of the entry complexity of A'. Also note that since the parameter dependence in the algorithm of [7] is roughly $\mathrm{ec}(A)^{\mathrm{td}_D(A)^2 \cdot 2^{\mathrm{td}_D(A)}}$, improving the exponent by replacing A with a row-equivalent matrix with smaller dual tree-depth likely outweighs the increase in the coefficients, which enters as the base of the exponent. Hence, we obtain the following corollary of the theorem.

▶ **Corollary 4.** *There exists a computable function $g'' : \mathbb{N}^2 \to \mathbb{N}$ such that integer programs with n variables and a constraint matrix A can be solved in time polynomial in $g''(\mathrm{bd}(A), \mathrm{ec}(A))$ and n, where $\mathrm{bd}(A)$ and $\mathrm{ec}(A)$ are the branch-depth and the entry complexity of the matrix A, i.e., integer programming is fixed parameter tractable when parameterized by branch-depth and entry complexity.*

We note that the results of [7,25] give a strongly fixed-parameter algorithm (i.e., an algorithm whose number of arithmetic operations does not depend on the size of the numbers involved) for integer programming in the regimes discussed above if the objective function f is a linear function (i.e., $f(\mathbf{x}) = \mathbf{w}\mathbf{x}$ for some $\mathbf{w} \in \mathbb{Z}^n$). Hence, the corollary above also gives a strongly-polynomial algorithm when f is a linear function.

We also remark that existing hardness results imply that the parameterization both by branch-depth and entry complexity in Corollary 4 is necessary unless FPT =W[1], i.e., it is not sufficient to parameterize instances only by one of the two parameters. Likewise, it is not possible to replace the branch-depth parameter by the more permissive notion of branch-width [3]. In fact, even solving integer programs with constant dual tree-width and constant entry complexity is NP-hard [25] (the dual tree-width of A is an upper bound on the branch-width of the vector matroid formed by columns of A). Let us also mention that Fomin et al. [8] proved lower bounds on the complexity of integer programming parameterized by branch-width under the exponential-time hypothesis.

The algorithm given in Corollary 4 is parameterized by the branch-depth of the vector matroid formed by the columns of the matrix A, i.e., it corresponds to the dual tree-depth of A. It is natural to ask whether the tractability also holds in the setting dual to this one, i.e., when the branch-depth of the vector matroid formed by the rows of the matrix A is bounded. This hope is dismissed in Proposition 14.

2 Notation

In this section, we fix the notation used throughout the paper and present the notions of tree-depth of a graph and of branch-depth of a matroid, including the results concerning them that we will need further. To avoid our presentation becoming cumbersome through adding or subtracting one at various places, we define the *depth* of a rooted tree to be the maximum number of edges on a path from the root to a leaf, and define the *height* of a rooted tree to be the maximum number of vertices on a path from the root to a leaf, i.e., the height of a rooted tree is always equal to its depth increased by one. The *depth of a vertex* in a rooted tree is the number of edges on the path from the root to that particular vertex. The height of a rooted forest F is the maximum height of a rooted tree in F. The *closure* $\mathrm{cl}(F)$ of a rooted forest is the graph obtained by adding edges from each vertex to all its descendants. Finally, the *tree-depth* $\mathrm{td}(G)$ of a graph G is the minimum height of a rooted forest F such that the closure $\mathrm{cl}(F)$ of the rooted forest F contains G as a subgraph. It can be shown that the path-width of a graph G is at most its tree-depth $\mathrm{td}(G)$ decreased by

one, and in particular, the tree-width of G is at most its tree-depth decreased by one (in this extended abstract, we do not give the definitions of path-width and tree-width here due to space limitations). We would like to note that the tree-depth as used in [23] is equal to the minimum depth of a rooted tree F such that $G \subseteq \mathrm{cl}(F)$, however, we here follow the definition of tree-depth that is standard; still, we wish to highlight this subtle difference since [23] is one of our main references.

The *primal graph* of an $m \times n$ matrix A is the graph $G_P(A)$ with vertices $\{1, \ldots, n\}$, i.e., its vertices correspond to the columns of A, where vertices i and j are connected if A contains a row whose i-th and j-th entries are non-zero. The *primal tree-depth* $\mathrm{td}_P(A)$ of a matrix A is the tree-depth of its primal graph. Analogously, the *dual graph* of A is the graph $G_D(A)$ with vertices $\{1, \ldots, m\}$, i.e., its vertices correspond to the rows of A, where vertices i and j are connected if A contains a column whose i-th and j-th entries are non-zero, i.e., the dual graph $G_D(A)$ is isomorphic to the primal graph of the matrix A^T. Finally, the *dual tree-depth* of A, which is denoted by $\mathrm{td}_D(A)$, is the tree-depth of the dual graph $t_D(A)$.

We next introduce the notion of branch-depth of a matroid. To keep our presentation self-contained, we start by recalling the definition of a matroid. A *matroid* M is a pair $(X, \mathcal{I})$, where $\mathcal{I}$ is a non-empty hereditary collection of subsets of X that satisfies the *augmentation axiom*. The collection $\mathcal{I}$ is hereditary if for every $X' \in \mathcal{I}$, $\mathcal{I}$ contains all subsets of X'. The augmentation axiom asserts that for all $X' \in \mathcal{I}$ and $X'' \in \mathcal{I}$ with $|X'| < |X''|$, there exists an element $x \in X'' \setminus X'$ such that $X' \cup \{x\} \in \mathcal{I}$. The sets contained in $\mathcal{I}$ are referred to as *independent*. The *rank* $r(X')$ of a set $X' \subseteq X$ is the size of the maximum independent subset of X'; the rank $r(M)$ of a matroid $M = (X, \mathcal{I})$ is the rank of X and an independent set of size $r(M)$ is a *basis* of M. A *circuit* is a set $X' \subseteq X$ such that X' is not independent but every proper subset of X' is. Two elements of x and x' are said to be *parallel* if $r(\{x\}) = r(\{x'\}) = r(\{x, x'\}) = 1$, and an element x of M is a *loop* if $r(\{x\}) = 0$.

Two particular examples of matroids are graphic matroids and vector matroids. If G is a graph, then the pair $(E(G), \mathcal{I})$ where $\mathcal{I}$ contains all acyclic subsets of edges of G is a matroid and is denoted by $M(G)$; matroids of this kind are called *graphic matroids*. If X is a set of vectors of a vector space and $\mathcal{I}$ contains all subsets of X that are linearly independent, then the pair $(X, \mathcal{I})$ is a matroid; matroids of this kind are *vector matroids*. In the setting of vector matroids, the rank of $X' \subseteq X$ is the dimension of the linear hull of X'. If $(X, \mathcal{I})$ is a vector matroid, we write $\mathcal{L}(X')$ for the linear hull of the vectors contained in $X' \subseteq X$ and abuse the notation by writing $\dim X'$ for $\dim \mathcal{L}(X')$.

In what follows, we will need a notion of a quotient of a vector space, which we now recall. If A is a vector space and K a subspace of A, the *quotient space* A/K is a vector space of dimension $\dim A - \dim K$ obtained from A by considering cosets of A given by K and inheriting addition and scalar multiplication from A; see e.g. [13] for further details if needed. One can show show for every subspace K of A, there exists a subspace B of A with dimension $\dim A - \dim K$ such that each coset contains a single vector from B, i.e., every vector w of A can be uniquely expressed as the sum of a vector w_B of B and a vector w_K of K. We call the vector w_B to be the *quotient* of w by K. Note that the quotient of a vector is not uniquely defined by K, however, it becomes uniquely defined when the subspace B is fixed.

A *depth-decomposition* of a matroid $M = (X, \mathcal{I})$ is a pair (T, f), where T is a rooted tree and f is a mapping from X to the leaves of T such that the number of edges of T is the rank of M and the following holds for every subset $X' \subseteq X$: the rank of X' is at most the number of edges contained in the paths from the root to the vertices $f(x)$, $x \in X'$. If the matroid M contains two parallel elements x and x', we will always assume that $f(x) = f(x')$. The *branch-depth* $\mathrm{bd}(M)$ of a matroid M is the smallest depth of a tree T that forms a

depth-decomposition of M. For example, if $M = (X, \mathcal{I})$ is a matroid of rank r, T is a path with r edges rooted at one of its end vertices, and f is a mapping such that $f(x)$ is equal to the non-root end vertex of T for all $x \in X$, then the pair (T, f) is a depth-decomposition of M. In particular, the branch-depth of any matroid M is well-defined and is at most the rank of M. We remark that the notion of branch-depth of a matroid given here is the one defined in [22, 23]; another matroid parameter, which is also called branch-depth but is different from the one that we use here, is defined in [4]. Finally, the *branch-depth* $\mathrm{bd}(A)$ of a matrix A is the branch-depth of the vector matroid formed by the columns of A. Since the vector matroid formed by the columns of A and the vector matroid formed by the columns of any matrix row-equivalent to A are the same, the branch-depth of A is invariant under row operations.

An *extended depth-decomposition* of a vector matroid $M = (X, \mathcal{I})$ is a triple (T, f, g) such that (T, f) is a depth-decomposition of M and g is a bijective mapping from the non-root vertices of T to a basis of the linear hull of X that satisfies that every element $x \in X$ is contained in the linear hull of the g-image of the non-root vertices on the path from $f(x)$ to the root of T. We next state and prove a simple proposition on the way that the vectors forming M are expressed as linear combinations of the vectors of the base formed by the g-image.

▶ **Proposition 5.** *Let (T, f, g) be an extended depth-decomposition of a vector matroid M. Let X' be a subset of elements of M and let X'' be an independent subset of X' with $r(X')$ elements. Then for every vector $x' \in X'$, there is a vector $x'' \in X''$ such that x' is contained in the linear hull of the g-images of the vertices on the path from $f(x'')$ to the root.*

If (T, f, g) is an extended depth-decomposition of a matroid M and all g-images are elements of the matroid M, then we say that the extended depth-decomposition is *principal*. Kardoš et al. [23, Corollary 3.17] designed an algorithm that outputs an approximation of an optimal depth-decomposition; we state the result here for the case of vector matroids.

▶ **Theorem 6.** *There exists a polynomial-time algorithm that given a vector matroid M and an integer d, either outputs that the branch-depth of M is larger than d or outputs a principal extended depth-decomposition of M of depth at most 4^d.*

If (T, f, g) is an extended depth-decomposition and u is a vertex of T, then K_u is the linear hull of the g-images of the vertices on the path from u to the root of T; in particular, if u is the root, then K_u contains the zero vector only. It will always be clear from the context for which extended depth-decomposition of M the spaces K_u are defined since the vertex u determines which rooted tree T is considered.

A *branch* of a rooted tree T is a subtree S rooted at a vertex u of T, with u having at least two children, such that S contains exactly u, one child u' of u, and all descendants of u'. In particular, a rooted tree has a branch if and only if it has a vertex with at least two children. A branch S is *primary* if every ancestor of the root of S has exactly one child. Every rooted tree T that is not a rooted path has at least two primary branches and all primary branches are rooted at the same vertex. We write $\widehat{S}$ for the set of elements of the matroid M mapped by f to the leaves of S and $\|S\|$ for the number of edges of S. Let S be a branch of T and $S_1, \ldots, S_k$ be the other branches with the same root. The branch S is *at capacity* if

$$r\left(X \setminus \left(\widehat{S}_1 \cup \cdots \cup \widehat{S}_k\right)\right) = r(M) - \|S_1\| - \|S_2\| - \cdots - \|S_k\|,$$

where X is the set of all elements of the matroid M. Note that if S is primary, then the left side of the equality is $r(\widehat{S})$ and the right side is $h + \|S\|$ in this case, where h is the depth of the root of S In particular, a primary branch S is *at capacity* if and only if the rank of

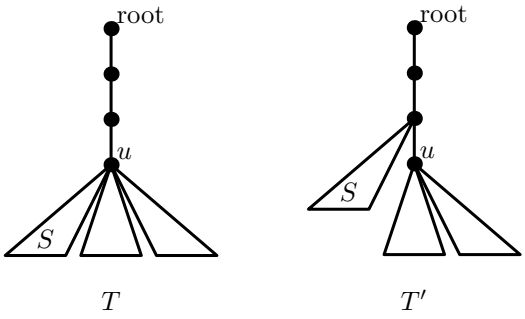

Figure 1 The trees T and T' from the statement of Lemma 8.

$\widehat{S}$ is equal to the sum of $\|S\|$, i.e., if and only if the rank inequality from the definition of a depth-decomposition holds with equality for the set $\widehat{S}$. Finally, a branch S rooted at a vertex u is *solid* if the matroid $(M/K_u)\left[\widehat{S}\right]$ is connected.

3 Optimal extended depth-decompositions

The goal of this section is to show that every vector matroid has an extended depth-decomposition with depth equal to its branch-depth. To do so, we start with noting that branches rooted at the root of the decomposition tree are always at capacity; the proof is left due to space limitations.

▶ **Lemma 7.** *Let (T, f) be a depth-decomposition of a vector matroid M. If T has a branch S rooted at the root of T, then S is at capacity.*

The following lemma is a core of an argument that every matroid has a depth-decomposition of optimal depth such that each primary branch is at capacity. See Figure 1 for the illustration of the operation described in the statement of the lemma.

▶ **Lemma 8.** *Let (T, f) be a depth-decomposition of a vector matroid M. Assume that T contains a primary branch S that is not at capacity. Let u be the root of S, and let T' be the rooted tree obtained from T by changing the root of S to be the parent of u. Then, (T', f) is a depth-decomposition of M.*

Proof. Let X be all elements of M and fix a subset X' of X. We need to show that $\dim X'$ is at most the number e_0 of edges on the paths in T' from the vertices in the f-image of X' to the root. If X' contains an element of $X \setminus \widehat{S}$, then the number of such edges is the same in the trees T and T' and the inequality follows from the fact that (T, f) is a depth-decomposition of M. Hence, we will assume that X' is a subset of $\widehat{S}$. Observe that collectively the primary branches of T different from S contain $r - h - \|S\|$ edges, where h is the depth of the root of S. We derive using the fact that (T, f) is a depth-decomposition the following:

$$\begin{aligned}
e_0 + 1 + (r - h - \|S\|) &\geq \dim X' \cup (X \setminus \widehat{S}) \\
&= \dim X' + \dim X \setminus \widehat{S} - \dim \mathcal{L}(X') \cap \mathcal{L}\left(X \setminus \widehat{S}\right) \\
&\geq \dim X' + \dim X \setminus \widehat{S} - \dim \mathcal{L}\left(\widehat{S}\right) \cap \mathcal{L}\left(X \setminus \widehat{S}\right) \\
&= \dim X' + \dim X \setminus \widehat{S} - (\dim \widehat{S} + \dim X \setminus \widehat{S} - \dim X) \\
&= \dim X' - \dim \widehat{S} + r.
\end{aligned}$$

This implies that $\dim X'$ is at most

$$e_0 + \dim \widehat{S} + 1 - h - \|S\| \leq e_0,$$

where the inequality follows using that S is not at capacity, i.e., $\dim \widehat{S} < h + \|S\|$. Hence, (T', f) is a depth-decomposition of M. ◀

We can obtain the following by iteratively applying Lemma 8.

▶ **Lemma 9.** *Let (T, f) be a depth-decomposition of a vector matroid $M = (X, \mathcal{I})$ of depth d. There exists a depth-decomposition of M of depth at most d such that every primary branch is at capacity.*

The following lemma describes the way how the structure of the vector matroids is captured by depth-decompositions such that each primary branch is at capacity.

▶ **Lemma 10.** *Let (T, f) be a depth-decomposition of a vector matroid $M = (X, \mathcal{I})$ such that T is not a rooted path and each primary branch of T is at capacity. Let $S_1, \ldots, S_k$ be the primary branches of T, and let $A_1, \ldots, A_k$ be the linear hulls of $\widehat{S_1}, \ldots, \widehat{S_k}$, respectively. Further, let h be the depth of the common root of $S_1, \ldots, S_k$ in T. There exists a subspace K of dimension h such that $A_i \cap A_j = K$ for all $1 \leq i < j \leq k$.*

We are now ready to prove the main theorem of this section.

▶ **Theorem 11.** *Let (T, f) be a depth-decomposition of a vector matroid $M = (X, \mathcal{I})$ of depth d. There exists an extended depth-decomposition of M of depth at most d.*

Proof. The proof proceeds by induction on the rank of M. By Lemma 9, we can assume that all primary branches of T are at capacity. If T is a rooted path, we assign elements of a basis of $\mathcal{L}(X)$ to the non-root vertices of T arbitrarily, i.e., we choose g to be any bijection to a basis of $\mathcal{L}(X)$, which yields an extended depth-decomposition (T, f, g) of M. Note that if the rank of M is one, then T is the one-edge rooted path, i.e., this case covers the base of the induction in particular.

We next assume that T is not a rooted path for the rest of the proof. Let $S_1, \ldots, S_k$ be the primary branches of T, and let h be the depth of the common root of $S_1, \ldots, S_k$. By Lemma 10, there exists a subspace K of dimension h such that the intersection of linear hulls of $\widehat{S_i}$ and $\widehat{S_j}$ is K for all $1 \leq i < j \leq k$; let $b_1, \ldots, b_h$ be an arbitrary basis of K.

We define M_i, $i = 1, \ldots, k$, to be the matroid such that the elements of M_i are $\widehat{S_i}$ and $X' \subseteq \widehat{S_i}$ is independent if and only if the elements $X' \cup \{b_1, \ldots, b_h\}$ are linearly independent. In particular, the rank of $X' \subseteq \widehat{S_i}$ in M_i is equal to $\dim X' \cup K - h$. The matroid M_i can be viewed as obtained by taking the vector matroid with the elements $\widehat{S_i} \cup \{b_1, \ldots, b_h\}$ and contracting the elements $b_1, \ldots, b_h$. In particular, M_i is a vector matroid, and the vector representation of M_i can be obtained from $\widehat{S_i}$ by taking quotients by K. Note that the rank of M_i is $\dim \widehat{S_i} \cup K - h$, i.e., its rank is smaller than the rank of M and we will be able to eventually apply induction to it.

Let f_i be the restriction of f to $\widehat{S_i}$. We claim that (S_i, f_i) is a depth-decomposition of M_i. Let X' be a subset of $\widehat{S_i}$, and let e_i be the number of edges contained in the union of paths from the elements $f(x)$, $x \in X'$, to the root of S_i. By the definition of M_i, the rank of X' in M_i is equal to $\dim X' \cup K - h$. Choose an arbitrary $j \neq i$, $1 \leq j \leq k$. Since (T, f) is a depth-decomposition of M, the intersection of linear hulls of $\widehat{S_i}$ and $\widehat{S_j}$ is K, and the branch S_j is at capacity, i.e., $\dim \widehat{S_j} = \|S_j\| + h$, we obtain that the rank of X' in M_i is equal to

$$\begin{aligned}\dim X' \cup K - h &= \dim X' \cup \widehat{S_j} - \dim \widehat{S_j} \\ &\leq e_i + \|S_j\| + h - \dim \widehat{S_j} = e_i.\end{aligned}$$

Hence, (S_i, f_i) is a depth-decomposition of M_i.

We now apply induction to each matroid M_i and its depth-decomposition (S_i, f_i), $i = 1, \ldots, k$, to obtain extended depth-decompositions (S'_i, f'_i, g_i) of M_i such that the depth of S'_i is at most the depth of S_i. Let T' be a rooted tree obtained from a rooted path of length h by identifying its non-root end with the roots of $S'_1, \ldots, S'_k$. Note that the depth of T' does not exceed the depth of T. Further, let f' be the unique function from X to the leaves of T such that the restriction of f' to the elements of M_i is f_i. Finally, let g be any function from the non-root vertices of T such that the h non-root vertices of the path from the root are mapped to the vectors $b_1, \ldots, b_h$ by g and $g(v) = g_i(v)$ for every non-root vertex v of S_i.

We claim that (T', f', g) is an extended depth-decomposition of M. We first verify that, for every $x \in X$, $f'(x)$ is contained in the linear hull of the g-image of the non-root vertices on the path from $f'(x)$ to the root. Fix $x \in X$ and let i be such that $x \in \widehat{S}_i$. Since (S'_i, f'_i, g_i) is an extended depth-decomposition of M_i, x is contained in the linear hull of K and the g_i-images of the non-root vertices on the path from $f'(x) = f_i(x)$ to the root of S'_i. Hence, x is contained in the linear hull of the g-image of the non-root vertices on the path from $f'(x)$ to the root of T'.

Consider now an arbitrary subset $X' \subseteq X$. We have already established that all elements of X' are contained in the linear hull of the g-image of the non-root vertices on the paths from $f'(x)$, $x \in X'$, to the root of T'. Since the dimension of this linear hull is equal to the number of non-root vertices on such paths, which is equal to the number of edges of the paths, it follows that (T', f') is a depth-decomposition of M. ◀

4 Optimal tree-depth of a matrix

In this section, we relate the optimal dual tree-depth of a matrix A to its branch-depth. We start with observing that the branch-depth of a matrix A is at most its dual tree-depth; the proof is left due to space limitations.

▶ **Proposition 12.** *If A is an $m \times n$ matrix, then* $\mathrm{bd}(A) \leq \mathrm{td}_D(A)$.

We next establish the main theorem of this section.

▶ **Theorem 13.** *Let A be an $m \times n$ matrix of rank m, M the vector matroid formed by columns of A, and (T, f, g) an extended depth-decomposition of M. Further, let* $\mathrm{Im}(g) = \{w_1, \ldots, w_m\}$. *The dual tree-depth of the $m \times n$ matrix A' such that the j-th column of A is equal to*

$$\sum_{i=1}^{m} A'_{ij} w_i$$

is at most the depth of the tree T.

Proof. Let F be the rooted forest obtained from T by removing the root and associate the i-th row of A' with the vertex v of F such that $g(v) = w_i$. Note that the height of F is the depth of T. We will establish that the dual graph $G_D(A')$ is contained in the closure $\mathrm{cl}(F)$ of F. Let i and i', $1 \leq i, i' \leq m$, be such that the vertices of F associated with the i-th and i'-th rows of A' are adjacent in $G_D(A')$. This means that there exists j, $1 \leq j \leq n$, such that $A'_{ij} \neq 0$ and $A'_{i'j} \neq 0$. Let v be the leaf of T such that the j-th column of A is mapped by f

to v. The definition of an extended depth-decomposition yields that the j-th column is a linear combination of the g-image of the non-root vertices on the path from v to the root of T. In particular, the path contains the two vertices of T mapped by g to w_i and $w_{i'}$; these two vertices are associated with the i-th and i'-th rows of A'. Hence, the vertices associated with the i-th and i'-th rows are adjacent in $\mathrm{cl}(F)$. We conclude that $G_D(A')$ is a subgraph of $\mathrm{cl}(F)$. ◀

Proposition 12 and Theorem 11 combine to a proof of Theorem 1.

5 Parameterized algorithms for integer programming

The main purpose of this section is to combine Theorem 1 with the existing approximation algorithm for branch-depth (Theorem 6) to obtain an approximation algorithm for computing a row-equivalent matrix with small dual tree-depth if it exists.

Proof of Theorem 2. Let A be an $m \times n$ matrix. Without loss of generality, we can assume that the rows of A are linearly independent, i.e., the rank of A is m. This also implies that the rank of the column space of A is m, in particular, $n \geq m$.

We apply the approximation algorithm described in Theorem 6 to the vector matroid M formed by the columns of the matrix A, and we obtain an extended depth-decomposition (T, f, g) of M. If the depth of T is larger than 4^d, then the branch-depth of A is larger than d; we report this and stop. Let B_g be the matrix with the columns formed by the vectors in $\mathrm{Im}(g)$ and let $B = B_g^{-1}$. Note that the matrix A' from the statement of Theorem 13 is equal to BA. By Theorem 13, the dual tree-depth of A' is at most 4^d. The proof that the entry complexity of A' is at most $O(d \cdot 4^d \cdot \mathrm{ec}(A))$ is left due to space limitations. ◀

Theorem 2 yields Corollary 4, which asserts that integer programming is fixed parameter tractable when parameterized by the branch-depth and the entry complexity of the constraint matrix. We complement this corollary by showing that integer programming is not fixed parameter tractable when parameterized by the "primal" branch-depth.

▶ **Proposition 14.** *Integer programming is* NP*-hard for instances with constraint matrices A satisfying* $\mathrm{bd}(A^T) = 1$ *and* $\mathrm{ec}(A) = 1$*, i.e., for instances such that the vector matroid formed by rows of the constraint matrix has branch-depth one.*

6 Structure of extended depth-decompositions

In this section, we present structural results on extended depth-decompositions that we need to design a fixed parameter algorithm to compute a depth-decomposition of a vector matroid with an optimal depth. The proofs of the next two lemmas are left due to space limitations; we note that the first of the two lemmas can be viewed as a generalization of Lemma 10.

▶ **Lemma 15.** *Let (T, f) be a depth-decomposition of a vector matroid M and let U be a set of vertices of T such that every vertex contained in U has at least two children and every ancestor of a vertex in U with at least two children is contained in U. Assume that every branch of T rooted at a vertex from U is at capacity. Every vertex $u \in U$ can be associated with a subspace L_u of the linear hull of the elements of M such that the dimension of L_u is the depth of u and the following holds. Let u be a vertex of U, let $S_1, \ldots, S_k$ be all branches rooted at u, and let $A_1, \ldots, A_k$ be the linear hulls of $\widehat{S_1}, \ldots, \widehat{S_k}$, respectively. If each ancestor of u has a single child, let L_0 be the vector space containing the zero vector only; otherwise, let u' be the nearest ancestor of u with at least two children, and let L_0 be the space $L_{u'}$. It holds that $\mathcal{L}(A_i \cup L_0) \cap \mathcal{L}(A_j \cup L_0) = L_u$ for all $1 \leq i < j \leq k$.*

▶ **Lemma 16.** *Let (T, f) be a depth-decomposition of a vector matroid M, u_1 a vertex of T with at least 2 children, and $u_2, \ldots, u_k$ all ancestors of u_1 with at least two children (listed in the increasing distance from u_1). Assume that every branch rooted at one the vertices $u_1, \ldots, u_k$ is at capacity, and let L_1 be the space L_{u_1} from the statement of Lemma 15 applied with $U = \{u_1, \ldots, u_k\}$. Further, let S_1 be any branch rooted at u_1 and f_1 the restriction of f to S_1. The pair (S_1, f_1) is a depth-decomposition of the vector matroid $(M/L_1)\left[\widehat{S_1}\right]$ and a branch of (S_1, f_1) is at capacity if and only if it is at capacity in (T, f). In addition, if (S_1', f_1') is another depth-decomposition of the vector matroid $(M/L_1)\left[\widehat{S_1}\right]$, then (T', f') is a depth-decomposition of the vector matroid M, where T' is obtained from T by replacing S_1 with S_1', and the function f' is defined as $f'(x) = f_1'(x)$ for $x \in \widehat{S_1}$, and $f'(x) = f(x)$ otherwise.*

Lemmas 15 and 16 allow to extend Lemma 8 to all branches.

▶ **Lemma 17.** *Let (T, f) be a depth-decomposition of a vector matroid M, and S_0 a branch of T rooted at a vertex u_0 such that S_0 is not at capacity. Suppose that every branch rooted at an ancestor of u_0 is at capacity. Let T' be the rooted tree obtained from T by changing the root of S_0 to be the parent of u_0. Then, (T', f) is a depth-decomposition of M.*

Lemmas 15–17 yield an iterative algorithm described in the next theorem.

▶ **Theorem 18.** *There exists a polynomial time algorithm that given a vector matroid M and a depth-decomposition (T, f) of M outputs an extended depth-decomposition (T', f', g) of M such that the depth of T' is at most the depth of T and every branch of T' is at capacity.*

We obtain the following two statements as corollaries of Theorem 18.

▶ **Corollary 19.** *Every vector matroid M has a depth-decomposition (T, f) with depth $\mathrm{bd}(M)$ such that every branch of T is at capacity.*

▶ **Corollary 20.** *If (T, f) is a depth-decomposition of a vector matroid M, then there exists g such that (T, f, g) is an extended depth-decomposition of M.*

We conclude this section with a theorem that asserts that every vector matroid has a depth-decomposition of minimum depth that has a special structure. We need three auxiliary lemmas.

▶ **Lemma 21.** *Let M be a vector matroid and $M_1, \ldots, M_k$ be its components. Further, let (T_i, f_i, g_i) be an extended depth-decomposition of M_i. Let T be the rooted tree obtained from the trees $T_1, \ldots, T_k$ by identifying the roots of the trees, let f be the mapping from the elements of M to the leaves of T such that $f(x) = f_i(x)$ if x belongs to M_i, and let g be the mapping such that $g(v) = g_i(v)$ if v is a non-root vertex of T_i. The triple (T, f, g) is an extended depth-decomposition of M.*

▶ **Lemma 22.** *Let (T, f, g) be an extended depth-decomposition of a vector matroid M, and let u be a vertex with at least two children. If S is a branch rooted at u, then $\widehat{S}$ is a union of components of M/K_u.*

▶ **Lemma 23.** *Let (T, f, g) be an extended depth-decomposition of a vector matroid M, and let u be a vertex with at least two children. Further, let S be a branch rooted at u and (T', f', g') be an extended depth-decomposition of the matroid $(M/K_u)\left[\widehat{S}\right]$. Let T'' be the rooted tree obtained by removing from T the branch S and identifying the root of T' with*

u, setting $f''(x) = f'(x)$ for elements $x \in \widehat{S}$ and $f''(x) = f(x)$ for other elements x of M, and setting $g''(v) = g(v)$ for vertices of T not contained in S and $g''(v) = g'(v) + K_u$ for non-root vertices of T'. The triple (T'', f'', g'') is an extended depth-decomposition of M.

We are now ready to prove the final theorem of this section.

▶ **Theorem 24.** *Every vector matroid M has an extended depth-decomposition (T, f, g) of depth $\mathrm{bd}(M)$ such that every branch of T is both at capacity and solid.*

Proof. We start with a depth-decomposition (T, f, g) of M with depth $\mathrm{td}(M)$ and modify it iteratively as follows. At each iteration, we first apply Theorem 18 to obtain a depth-decomposition such that every branch is at capacity. If every branch is solid, we stop. If there is a branch S that is not solid, we proceed as follows. Since S is not solid, the matroid $(M/K_u)\left[\widehat{S}\right]$ is not connected, where u is the root of S. Let $M_1, \ldots, M_k$ be the components of the matroid $(M/K_u)\left[\widehat{S}\right]$ and let X_u be the set containing all loops of the matroid $(M/K_u)\left[\widehat{S}\right]$. Let (S_i, f_i, g_i) be an extended depth-decomposition of M_i, $i = 1, \ldots, k$, with depth $\mathrm{bd}(M_i)$. Since the branch-depth $\mathrm{bd}(M_i)$ of M_i is at most the branch-depth of $(M/K_u)\left[\widehat{S}\right]$, the depth of each of the trees $S_1, \ldots, S_k$ is at most the depth of S. By Lemmas 21 and 23, it is possible to replace the branch S with the branches $S_1, \ldots, S_k$ rooted at the root of S and assigning the elements of X_u to arbitrary leaves of the branches $S_1, \ldots, S_k$. Note that the depth of the new rooted tree does not exceed the depth of the original rooted tree. In this way, we obtain a new extended depth-decomposition of M, and we proceed to the next iteration. The proof that the procedure described above terminates after at most $r^{\mathrm{bd}(M)+1}$ iterations is left due to space limitations. ◀

7 Algorithm for finite fields

In this section, we design a fixed parameter algorithm for computing a depth-decomposition of a vector matroid over a fixed finite field. To do so, we need to introduce additional notation. Let (T, f, g) be an extended depth-decomposition of a vector matroid M, and let r be the rank of M. Let $u_0, \ldots, u_{2r}$ be a depth-first-search transversal of the tree T (see Figure 2 for illustration). For $i \in \{0, \ldots, 2r\}$, we define A_i to be the linear hull of K_{u_i} and the f-preimage of the leaves among the vertices $u_0, \ldots, u_i$. Similarly, we define B_i to be the linear hull of K_{u_i} and the f-preimage of the leaves among the vertices $u_i, \ldots, u_{2r}$. The sequence $(u_i, A_i, B_i)_{i \in \{0,\ldots,2r\}}$ is called a *transversal sequence* for (T, f, g). Note that $A_i \cap B_i = K_{u_i}$ by the fact that $\mathrm{Im}(g)$ is a basis of the linear hull of elements of M. If (T, f, g) is principal and (T', f', g') is another extended depth-decomposition of M, we say that a branch S of T' is *i-crossed* if $\widehat{S}$ contains the g-image of a vertex on the path from u_i to the root of T.

▶ **Lemma 25.** *Let M be a vector matroid, (T, f, g) a principal extended depth-decomposition of M, and (T', f', g') an extended depth-decomposition of M such that every branch is solid. Further, let $(u_i, A_i, B_i)_{i \in \{0,\ldots,2r\}}$ be a transversal sequence for (T, f, g). If S is a branch of (T', f', g') that is not i-crossed, then $\widehat{S}$ is a subset of A_i or B_i.*

We will design a dynamic programming algorithm, which will be constructing an optimal depth-decomposition of a vector matroid M using the information on the structure of M captured by an extended depth-decomposition of M produced by an approximation algorithm

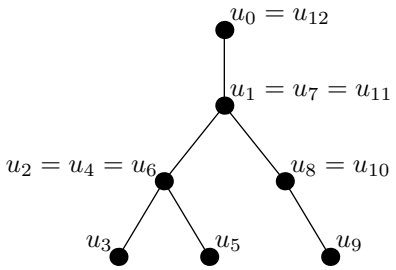

Figure 2 An example of a depth-first-search transversal of a rooted tree.

given in Theorem 6. The depth-decomposition will be constructed iteratively for elements of M in the order in that the leaves that they are assigned to appear in the transversal sequence of the depth-decomposition produced by the approximation algorithm. Since it would not be feasible to store all possible "partial" depth-decompositions, we need a more succinct way of representing an already constructed part of a depth-decomposition, which we now formally introduce.

Let (T, f, g) be a principal extended depth-decomposition of a vector matroid M with rank r over a field $\mathbb{F}$, $(u_i, A_i, B_i)_{i\in\{0,\ldots,2r\}}$ a transversal sequence for (T, f, g), and (T', f', g') another extended depth-decomposition of a matroid M. A *frontier* is a tuple (T_0, d, a, b, h) such that d, a, and b are non-negative integers, T_0 is a rooted tree of depth with at most d leaves, and h is a mapping from non-root vertices of T_0 to $\mathbb{F}^{d+a+b}$ such that $\mathrm{Im}(h)$ is a set of linearly independent vectors and for every $j = 1, \ldots, d$, there is a leaf of T_0 for which the j-th unit vector is contained in the linear hull of the h-image of the vertices on the path from the leaf to the root of T_0. We will refer the middle a coordinates of images of h as *A-coordinates* and to the last b coordinates as *B-coordinates*.

The *i-frontier* of (T', f', g') with respect to (T, f, g) and $(u_i, A_i, B_i)_{i\in\{0,\ldots,2r\}}$ is the frontier (T_0, d, a, b, h) such that

- T_0 is the rooted subtree of T' formed by the paths from the f'-images of U to the root, where U is the set of g-images of the vertices on the path from u_i to the root of T.
- The integer d is the depth of u_i in T.
- The integers a and b are the smallest integers for that there exists an a-dimensional subspace L_A of A_i and a b-dimensional subspace L_B of B_i such that the linear hull of the g'-images of the vertices of T_0 (note that T_0 is a subtree of T') is a subspace of the linear hull of $v_1^u, \ldots, v_d^u$, L_A and L_B, where $v_1^u, \ldots, v_d^u$ are g-images of the vertices on the path from the root of T to u_i (in this order).
- Finally, h is a mapping from the non-root vertices of T_0 to $\mathbb{F}^{d+a+b}$ that satisfies the following. Let $v_1^A, \ldots, v_a^A$ be vectors such that $v_1^u, \ldots, v_d^u, v_1^A, \ldots, v_a^A$ form a basis of L_A, and let $v_1^B, \ldots, v_b^B$ be vectors such that $v_1^u, \ldots, v_d^u, v_1^B, \ldots, v_b^B$ form a basis of L_B. The value $h(v)$ for a non-root vertex v of T_0 is equal to the coordinates of $f'(v)$ with respect to the (linearly independent) vectors $v_1^u, \ldots, v_d^u, v_1^A, \ldots, v_a^A, v_1^B, \ldots, v_a^B$.

The following lemma justifies the definition of an *i-frontier*. Informally speaking, the lemma says that two depth-decompositions of a vector matroid M can be combined along the same i-frontier, i.e., the i-frontier contains all information that needs to be stored when iteratively constructing a depth-decomposition of M in a dynamic way for the elements of contained in $A_0, A_1, \ldots, A_{2r}$.

▶ **Lemma 26.** *Let (T, f, g) be a principal extended depth-decomposition of a vector matroid M, $(u_i, A_i, B_i)_{i\in\{0,\dots,2r\}}$ a transversal sequence for (T, f, g), and (T', f', g') and (T'', f'', g'') two solid extended depth-decompositions of M. Suppose that the i-frontiers of (T', f', g') and (T'', f'', g'') with respect to (T, f, g) and $(u_i, A_i, B_i)_{i\in\{0,\dots,2r\}}$ are the same, and let T_0 be the rooted tree of the i-frontier. Obtain T'_A from T' by removing all branches S with $\widehat{S} \subseteq B_i$ that are not i-crossed, T''_B from T'' by removing all branches S with $\widehat{S} \subseteq A_i$ that are not i-crossed, and T''' by gluing T'_A and T''_B together on the vertices that correspond to each vertex of T_0. Finally, let f''' be a function from the elements of M to the leaves of T''' defined as follows. If $x \in A_i \setminus B_i$, then $f'''(x) = f'(x)$. If $x \in B_i \setminus A_i$, then $f'''(x) = f''(x)$. Lastly, if $x \in A_i \cap B_i = K_{u_i}$, then $f'''(x)$ is any leaf u of T_0 such that $x \in \mathcal{L}(g'(P_u))$. Then (T''', f''') is a depth-decomposition of M.*

Before stating the main result of this section, we need to observe that the number of frontiers for any fixed d is bounded.

▶ **Lemma 27.** *For every integer d and any finite field $\mathbb{F}$, there exist at most $d^{2d+4}|\mathbb{F}|^{2d^4}$ choices of a rooted tree T of depth at most d, integers $d' \leq d$, a and b and a mapping h to $\mathbb{F}^{d'+a+b}$ such that (T, d', a, b, h) is a frontier.*

We can now design a dynamic programming algorithm for computing the branch-depth of a matroid represented over a fixed finite field. While there are many technical details that needs to be taken care of, the basic idea of the algorithm is simple: we first obtain an approximate depth-decomposition using Theorem 6 and then proceed computing along its depth-first-search transversal possible frontiers; Lemma 26 guarantees that frontiers capture all information that needs to be carried through dynamic programming, and their number of frontiers is bounded by Lemma 27.

▶ **Theorem 28.** *For the parameterization by a positive integer d and a prime power q, there exists a fixed parameter algorithm that for a vector matroid M over the q-element field either outputs that $\mathrm{bd}(M)$ is larger than d, or outputs a depth-decomposition of M with depth d.*

Proof. We first apply the algorithm from Theorem 6. The algorithm either outputs that the branch-depth of M is larger than d or outputs a principal extended depth-decomposition (T, f, g) of a vector matroid M with depth at most 4^d. For the purpose of the analysis of the algorithm that we present, fix a solid extended depth-decomposition (T_s, f_s, g_s) of M with depth $\mathrm{bd}(M)$, which exists by Theorem 24.

Let r be the rank of the matroid M, and let $(u_i, A_i, B_i)_{i\in\{0,\dots,2r\}}$ be a transversal sequence for (T, f, g). The algorithm then iteratively for $j = 0, \dots, 2r$ computes a list of all frontiers (T_0, d', a, b, h), $d' \leq d$ for which there there exists a vector matroid M' with rank $\dim A_i + b$ such that the restrictions of M and M' to the elements contained in the subspace A_i are the same, and a solid extended depth-decomposition (T', f', g') of M' of depth at most d such that (T, d', a, b, h) is the i-frontier of (T', f', g') with respect to (T, f, g) and its transversal sequence.

Note that the number of such frontiers is bounded by a function of d and q only by Lemma 27; the number of edges of T can be shown to be $d' + a + b$. If the branch-depth of M is at most d, then the set of such frontiers is non-empty for every $j = 0, \dots, 2r$: the matroid M' in the i-th iteration can be chosen to be the the union of the restriction of the matroid M to the elements of A_i and the elements of K_u, where u ranges over all the vertices on the path from u_i to the root of T. A solid extended depth-decomposition (T', f', g') can be obtained from (T_s, f_s, g_s) by removing all branches S with $\widehat{S} \subseteq B_i$ that are not i-crossed. So, if the set of the frontiers becomes empty at any of the iterations, the algorithm can stop and output that the branch-depth of M is larger than d.

We now describe the iterations of the algorithm in detail. For $j = 0$, the list of frontiers contains a single element $(R, 0, 0, 0, h)$, where R is the rooted tree that contains the root only and h is the null mapping. We now describe how the algorithm computes the list for $j > 0$ assuming that the list is already available for $j - 1$. The iteration of the algorithm differs according to whether u_j is a parent of u_{j-1} and u_j is a child of u_{j-1}.

We start with describing the case that u_j is a parent of u_{j-1}. Let $d' < d$ be the depth of u_j. Then the depth of u_{j-1} is $d' + 1$. The following is performed for every frontier of the form $(T_0, d' + 1, a, b, h)$ in the list from the previous iteration, and every leaf u of T_0 such that the linear hull of the h-image of the vertices from u to the root contains the $(d' + 1)$-th unit vector. If the leaves of T_0 can be assigned distinct indices $i' = 1, \ldots, d'$ such that the linear hull of the h-image of the vertices from each leaf to the root contains contains the i'-th unit vector, where i' is the index assigned to the leaf, then we include $(T_0, d', a + 1, b, h')$ to the list from the j-th iteration, where h' is a mapping from the non-root vertices of T_0 obtained from h by turning the $(d' + 1)$-th coordinate to be one of the A-coordinates and applying any invertible linear transformation to the $a + 1$ A-coordinates, i.e., we fix such a linear transformation L and set $h'(v) = L(h(v))$ for all vertices v of T_0. If there exists $i' = 1, \ldots, d'$ such that u is the only leaf for which the linear hull of the h-image of the vertices from it to the root contains the i'-th unit vector, we continue with the next choice of (T_0, d', a, b, h') and u. Otherwise, for every $i' = 1, \ldots, d'$ there exists another leaf for which the linear hull of the h-image of the vertices from it to the root contains the i'-th unit vector, which means that we may be able to remove u from T_0. So, let T_0' be the tree obtained from T_0 by removing the path from u to the nearest ancestor with at least two children and turn the $(d' + 1)$-th coordinate to an A-coordinate. If the linear hull of the h-images of the vertices of T_0' restricted to their B-coordinates does not have dimension b, then it is not possible to modify this frontier by removing u and we continue with the next choice of (T_0, d', a, b, h') and u. Otherwise, let a' be the dimension of the linear hull of the h-images of the vertices of T_0' restricted to their A-coordinates and include (T_0, d', a', b, h') to the list from the j-th iteration, where h' is a mapping obtained from h by mapping its A-coordinates by a linear transformation L such that L maps a-dimensional vector space to a'-dimensional vector space and its image has dimension a'.

We next describe the case that u_j is a child of u_{j-1}. Again, let $d' \leq d$ be the depth of u_j. Then the depth of u_{j-1} is $d' - 1$. The following is performed for every frontier of the form $(T_0, d' - 1, a, b, h)$ in the list from the previous iteration. For every leaf u and every $i' = 1, \ldots, b$ such that the unit vector for the i'-th B-coordinate is contained in the linear hull of the h-image of the vertices from u to the root, we turn the i'-th B-coordinate to the d'-th coordinate to obtain h' and include $(T_0, d', a, b - 1, h')$ to the list from the j-th iteration. In addition, we perform the following. For every $\ell = 1, \ldots, d$, we consider a rooted path of length ℓ and identify its root with a vertex T_0 in all possible ways that the depth of the resulting tree T' does not exceed d. Let u be the new leaf of T_0' and let h' be a mapping obtained from h by assigning each of the ℓ new vertices one of the unit vectors for the i'-th B-coordinates for $i' = b + 1, \ldots, b + \ell$. For every invertible linear transformation L to the $b + \ell$ B-coordinates that yields h'' such that the linear hull of the h''-images of the vertices on the path from u to the root of T_0' contains the unit vector for the $(b + \ell)$-th B-coordinate, we turn the $(b + \ell)$-th B-coordinate to the d'-th coordinate to obtain h''' and include $(T_0, d', a', b + \ell - 1, h''')$ to the list from the j-th iteration.

Assume that all the iterations of the algorithm have been performed. If the list of frontiers is empty after any iteration, the branch-depth of M exceeds d and the algorithm reports this. Otherwise, the final list (for $j = 2r$) contains a single element $(R, 0, 0, 0, h)$ where R is the

rooted tree that contains the root only. Tracing back to the list for $j = 0$, we obtain a series of frontiers $(T_i, d_i, a_i, b_i, h_i)$, $i = 0, \ldots, 2r$, such that the i-th one can be obtained from the $(i-1)$-th by the operations explained earlier. Note that d_i is the depth of u_i in T. In the frontier $(T_i, d_i, a_i, b_i, h_i)$, the first d_i coordinate of the h-image correspond to the g-image of the vertices on the path from the root of T to u_i (in this order). The way the frontiers were constructed guarantees consistency between the shape of the frontiers and the basis elements displayed by frontiers; in particular, the mappings h_i are projections of the linear hull of elements of M to the subspace corresponding to the i-frontier and the linear transformations used in the steps of the algorithm provides a consistent way of relating these projections with each other. Hence, there exists an extended depth-decomposition (T', f', g') such that $(T_i, d_i, a_i, b_i, h_i)$ is the i-frontier of (T', f', g') with respect to (T, f, g) and $(u_i, A_i, B_i)_{i \in \{0,\ldots,j\}}$. The algorithm computes this extended depth-decomposition (T', f', g') and outputs it. ◀

Theorems 18 and 28 yield the following corollary.

▶ **Corollary 29.** *For the parameterization by a positive integer d and a prime power q, there exists a fixed parameter algorithm that for a vector matroid M over the q-element field either outputs that $\mathrm{bd}(M)$ is larger than d, or computes $\mathrm{bd}(M)$ and outputs an extended depth-decomposition of M with depth $\mathrm{bd}(M)$ such that every branch of the depth-decomposition is at capacity.*

8 Algorithm for rational matrices

In this section, we adopt the algorithm presented in Section 7 to matroids over rationals; the proofs are left due to space limitations. We start with two auxiliary lemmas. We remark that the bound of 2^{2d-1} in Lemma 30 can be replaced with $d \cdot 2^{d-1}$ using a more careful analysis.

▶ **Lemma 30.** *Let M be a vector matroid and (T, f) a depth-decomposition of M with depth d such that every branch is at capacity. There exists a mapping g such that (T, f, g) is an extended depth-decomposition of M and every element of $\mathrm{Im}(g)$ is a linear combination of at most 2^{2d-1} elements of M.*

▶ **Lemma 31.** *Let A be an integer matrix of branch-depth (over $\mathbb{Q}$) at most d such that all its entries are between $-K$ and $+K$. Further, let q be a prime larger than $(K2^{2d})^{2^{2d}}$. The following holds for any subset X of the columns of A: the vectors contained in X are linearly independent over $\mathbb{Q}$ if and only they are independent over the q-element field.*

We derive the following using Theorem 28, Corollary 29, and Lemmas 30 and 31.

▶ **Theorem 32.** *For the parameterization by positive integers d and K, there exists a fixed parameter algorithm that for a vector matroid M over $\mathbb{Q}$ such that the entries of all vectors in M have complexity at most K either outputs that $\mathrm{bd}(M)$ is larger than d, or computes $\mathrm{bd}(M)$ and outputs an extended depth-decomposition (T, f, g) of M with depth $\mathrm{bd}(M)$. Moreover, the entry complexity of the vectors in $\mathrm{Im}(g)$ is bounded by a function of d and K.*

Theorem 32 yields Theorem 3.

References

1 Matthias Aschenbrenner and Raymond Hemmecke. Finiteness theorems in stochastic integer programming. *Foundations of Computational Mathematics*, 7(2):183–227, 2007.

2 Lin Chen and Dániel Marx. Covering a tree with rooted subtrees–parameterized and approximation algorithms. In *29th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 2801–2820. SIAM, 2018.

3 William H. Cunningham and Jim Geelen. On integer programming and the branch-width of the constraint matrix. In *Integer Programming and Combinatorial Optimization, 12th International IPCO Conference*, pages 158–166, 2007.

4 Matt DeVos, O-joung Kwon, and Sang-il Oum. Branch-depth: Generalizing tree-depth of graphs. *preprint*, 2019. `arXiv:1903.11988`.

5 Pavel Dvořák, Eduard Eiben, Robert Ganian, Dušan Knop, and Sebastian Ordyniak. Solving integer linear programs with a small number of global variables and constraints. In *26th International Joint Conference on Artificial Intelligence*, pages 607–613. AAAI Press, 2017.

6 Friedrich Eisenbrand, Christoph Hunkenschröder, and Kim-Manuel Klein. Faster Algorithms for Integer Programs with Block Structure. In *45th International Colloquium on Automata, Languages, and Programming, ICALP*, pages 49:1–49:13, 2018.

7 Friedrich Eisenbrand, Christoph Hunkenschröder, Kim-Manuel Klein, Martin Koutecký, Asaf Levin, and Shmuel Onn. An algorithmic theory of integer programming. *preprint*, 2019. `arXiv:1904.01361`.

8 Fedor V. Fomin, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. On the optimality of pseudo-polynomial algorithms for integer programming. In *26th Annual European Symposium on Algorithms, ESA*, pages 31:1–31:13, 2018.

9 Robert Ganian and Sebastian Ordyniak. The complexity landscape of decompositional parameters for ILP. *Artificial Intelligence*, 2018.

10 Robert Ganian, Sebastian Ordyniak, and M. S. Ramanujan. Going beyond primal treewidth for (M)ILP. In *31st AAAI Conference on Artificial Intelligence*, pages 815–821, 2017.

11 Tomas Gavenčiak, Daniel Král', and Sang-il Oum. Deciding first order properties of matroids. In *39th International Colloquium Automata, Languages, and Programming, ICALP*, volume 7392 of *Lecture Notes in Computer Science*, pages 239–250, 2012.

12 Jim Geelen, Albertus Gerards, Neil Robertson, and Geoff Whittle. On the excluded minors for the matroids of branch-width k. *Journal of Combinatorial Theory, Series B*, 88:261–265, 2003.

13 Paul Halmos. *Finite-Dimensional Vector Spaces*. Undergraduate Texts in Mathematics. Springer, 1993.

14 Raymond Hemmecke, Matthias Köppe, and Robert Weismantel. Graver basis and proximity techniques for block-structured separable convex integer minimization problems. *Mathematical Programming*, 145:1–18, 2014.

15 Raymond Hemmecke, Shmuel Onn, and Lyubov Romanchuk. N-fold integer programming in cubic time. *Mathematical Programming*, 137:325–341, 2013.

16 Petr Hliněný. Branch-width, parse trees, and monadic second-order logic for matroids. In Helmut Alt and Michel Habib, editors, *20th Annual Symposium on Theoretical Aspects of Computer Science, STACS*, volume 2607 of *LNCS*, pages 319–330, 2003.

17 Petr Hliněný. On matroid properties definable in the MSO logic. In Branislav Rovan and Peter Vojtáš, editors, *27th International Symposium on Mathematical Foundations of Computer Science, MFCS*, volume 2747 of *LNCS*, pages 470–479, 2003.

18 Petr Hliněný. Branch-width, parse trees, and monadic second-order logic for matroids. *Journal of Combinatorial Theory, Series B*, 96(3):325–351, 2006.

19 Petr Hliněný and Sang-il Oum. Finding branch-decompositions and rank-decompositions. In Lars Arge, Michael Hoffmann, and Emo Welzl, editors, *15th Annual European Symposium, ESA*, volume 4698 of *LNCS*, pages 163–174, 2007.

20 Petr Hliněný and Sang-il Oum. Finding branch-decompositions and rank-decompositions. *SIAM Journal on Computing*, 38(3):1012–1032, 2008.

21 Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, 12(3):415–440, August 1987.

22 František Kardoš, Daniel Král', Anita Liebenau, and Lukáš Mach. First order convergence of matroids. *preprint*, 2015. `arXiv:1501.06518`.

23 František Kardoš, Daniel Král', Anita Liebenau, and Lukáš Mach. First order convergence of matroids. *Eur. J. Comb*, 59:150–168, 2017.

24 Dušan Knop, Martin Koutecký, and Matthias Mnich. Voting and bribing in single-exponential time. In *34th Symposium on Theoretical Aspects of Computer Science, STACS*, volume 66, pages 46:1–46:14, 2017.

25 Martin Koutecký, Asaf Levin, and Shmuel Onn. A parameterized strongly polynomial algorithm for block structured integer programs. In *45th International Colloquium on Automata, Languages, and Programming, ICALP*, volume 107, pages 85:1–85:14, 2018.

26 Hendrik W. Lenstra, Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983.

27 Susan Margulies, Jing Ma, and Illya V. Hicks. The Cunningham-Geelen method in practice: Branch-decompositions and integer programming. *INFORMS Journal on Computing*, 25(4):599–610, 2013.