Research Report

COMPLEXITY ASPECTS OF MATROID BRANCH-WIDTH

RNDr. Petr Hliněný, Ph.D.

(http://iti.mff.cuni.cz/~hlineny)

January 24, 2003

Preface

We shortly introduce this work: Our research deals mainly with computational-complexity questions on matroids representable over finite fields, that are related to matroid branch-width. The notion of a matroid is seen as a wide combinatorial generalization of graphs. Necessary concepts of matroid theory are introduced later.

Considering graph theory, computer scientists know that a tree-like structure of the input graph (captured by the concepts of tree-width or branch-width) has strong computational consequences — many otherwise difficult problems have fast algorithms over graphs of bounded tree-width. We briefly mention Courcelle's work on the graph problems expressible in the MSO logic, Bodlaender's parametrized algorithm for finding an optimal tree decomposition of a graph, or computation of the (notoriously difficult) Tutte polynomial for graphs of bounder tree-width by Andrzejak and Noble.

The (less known) notion of branch-width is equivalent to tree-width, and branch-width has an immediate generalization to all matroids. Inspired by the above results, we consider matroids of bounded branch-width which are represented over finite fields: We provide a polynomial algorithm for constructing a branch decomposition, and we then use this algorithm to compute the matroid Tutte polynomial, and to decide matroid problems expressible in the MSO logic. We also show why these results cannot be extended further to other matroids of bounded branch-width.

Part I of the work surveys some basic definitions of graph theory, parametrized complexity, and mainly matroid theory. Moreover, our original formalism of boundaried parse trees for represented matroids is introduced there.

Part II of the work presents our major complexity results for represented matroids of bounded branch-width. The chapters in this part are based on our original research papers [26, 27, 28, 29], and on an extended abstract [25].

Acknowledgement

To Prof. Geoff Whittle, and to the Victoria University of Wellington...

Everybody who ever visited New Zealand would surely agree that it is one of the most beautiful places in the world. That is why, in 1999, when I got an offer of a postdoc position at the Victoria University of Wellington from Prof. Geoff Whittle, I immediately accepted it. No matter that I had to change my research area from graph theory to a difficult field of structural matroid theory.

Now I should say that this was the most important decision of my scientific career yet. Not only that I have spent two years in an extraordinarily beautiful place — I have also found a new exciting research area in between structural matroid theory and computational complexity. (Which, indeed, is the main topic of this work.)

So, I would like to thank Prof. Geoff Whittle for inviting me to Wellington, and for his great patience and encouragement when introducing me to matroid theory. I also thank to Victoria University and the NZ Marsden Fund for supporting my stay in New Zealand. Lastly, I want to mention Prof. Rod Downey who pointed my research towards parametrized complexity of matroid problems.

January 24, 2003,

Petr Hliněný.

Contents

Glossary of Symbols

Algebra

$[m,n]$	the set of integers $\{m, m+1, \dots, n\}$
$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{D}$	matrices
$\boldsymbol{I}_n$	unit $n \times n$ matrix
$\boldsymbol{J}_n$	$n \times n$ matrix with all entries 1
$[\boldsymbol{A} \mid \boldsymbol{B}]$	matrix obtained by concatenation of the matrices $\boldsymbol{A}, \boldsymbol{B}$
$\mathbb{F}$	(algebraic) commutative field
$\mathbb{N}, \mathbb{Z}, \mathbb{Q}$	natural, integer and rational numbers
$GF(q)$	finite field with q (a power of prime) elements
$PG(r, \mathbb{F})$	projective geometry of rank $r+1$ over the field $\mathbb{F}$
$PG(r, q)$	finite projective geometry $PG(r, GF(q))$
$\mathrm{r}(X)$	rank of the set X in a projective geometry
$\langle X \rangle$	span (affine closure) of the set X in a projective geometry

Combinatorics

F, G, H and T	graphs and trees
$V(G)$, $E(G)$	vertex, edge set of the graph G
$G - v$, $G - e$	deleting a vertex, an edge from the graph G
K_n	complete graph (clique) on n vertices
M, N	matroids ($M = (E, \mathcal{B})$)
$E(M)$	ground set of the matroid M
r_M	rank function of the matroid M
λ_M	connectivity function of the matroid M
$M(\boldsymbol{A})$	matroid represented by the matrix $\boldsymbol{A}$
$M(G)$	cycle matroid of the graph G
M^*	dual matroid of the matroid M
$M \setminus e$, M/e	deletion, contraction of the matroid element e
$U_{r,n}$	uniform matroid of rank r on n elements
$T(M; x, y)$	Tutte polynomial of the matroid M

Computer Science

P	class of problems computable in polynomial time
NP	nondeterministic polynomial-time class
$\#P$	class of counting polynomially-recognizable structures
$O(f(n))$	function bounded from above by a scalar multiple of $f(n)$
$\Omega(f(n))$	function bounded from below by a scalar multiple of $f(n)$
$\mathcal{A}$	finite (tree-) automaton
Σ^*	set of all words over an alphabet Σ
Σ^{**}	set of all Σ-labelled rooted sub-binary trees

Logic

ϕ, ψ	logical formulas
$\wedge, \vee, \neg$	logical connectives
$\forall, \exists$	logical quantifiers
$\equiv$	logical identity
$\models$	logical truth in a model

Our Symbols

$\mathcal{B}_t$	family of matroids of branch-width at most t
$\bar{M}, \bar{N}$	boundaried matroids (see Section 2.3)
$J(\bar{M})$	internal elements of a boundaried matroid
$\partial(\bar{M})$	boundary subspace of a boundaried matroid
$\bar{M} \bar{\oplus} \bar{N}$	boundaried sum of two boundaried matroids
$\bar{M} \odot \bar{N}$	boundaried composition operator $\odot$ (see Section 3.3)
$\bar{\Omega}_t$	empty t-boundaried matroid (see Section 3.3)
$\bar{\Upsilon}$	single-element boundaried matroid
$\mathcal{R}_t^{\mathbb{F}}$	set of all $\leq t$-boundaried composition operators over $\mathbb{F}$
Π_t	the set $\mathcal{R}_t^{\mathbb{F}} \cup \{\bar{\Upsilon}, \bar{\Upsilon}_0\}$
$\bar{P}(T)$	boundaried matroid parsed by the parse tree $T \in \Pi_t^{**}$

Complexity Aspects of Matroid Branch-Width

Petr Hliněný

Part I

INTRODUCTION

Chapter 1

Motivation and Aims

The motivation for our work lies on the fuzzy boundary between discrete mathematics and theoretical computer science. To introduce our topic, we start with a brief overview of the basic terms of graph theory. We list several well-known and simply stated graph problems which are computationally very hard. From the point of view of the classical complexity theory, all of these problems have the same difficulty – they are *NP*-complete. However, a new, parametric, approach to algorithmic complexity emerged in the past decade, which can show that some of these (and other) classical problems are easier to solve under certain circumstances.

The theory of *parametrized complexity* provides a background for analysis of algorithmic problems that is finer than the classical complexity theory. Briefly saying, a problem is called "fixed-parameter tractable" if there is an algorithm having running time with the (possible) super-polynomial part separated in terms of some natural "parameter", which is supposed to be small even for large input in practice. Successful practical applications of this concept are known, for example, in computational biology or in database theory.

We are, in particular, interested in the case of parameters that tell us how close is the input object to having a "tree-like shape". Even before the notions of *tree-width*, and less known *branch-width*, were formalized in the theoretical work of Robertson and Seymour; computer scientists had noticed that various problems become significantly easier when solved for input instances close to being trees. Moreover, such tree-like structures naturally arise in many practical situations. Examples include some problems in evolutional biology, modeling certain expert systems, or syntactic dependency in natural languages.

Nowadays, strong results are known showing that large classes of graph problems are efficiently solvable for input graphs of bounded tree-width, and most of these are even solvable in linear time. Among others, we mention the work of Courcelle on properties expressible in the monadic second-order logic. Fast implementations of those works are based on Bodlaender's parametrized linear-time algorithm for finding an optimal tree-decomposition of a graph of bounded width.

The aim of our research is to extend the above sketched ideas from graphs to representable matroids. The notion of a *matroid* gives a common abstract view of combinatorial and algebraic dependency; and it is formally introduced in Chapter 2. From a combinatorial point of view, matroids present a wide generalization of graphs, and many structural properties of graphs have natural analogues in matroids. This has shown to be also the case of parametrized complexity for representable matroids of tree-like structure.

The next chapters of Part I of our work are devoted to introducing necessary basic concepts in matroid theory, and the parse-tree formalism used throughout the work. Part II then gives our main complexity results — a parametrized algorithm for a bounded-width branch-decomposition of a representable matroid, and its applications to recognition of MSOL-definable properties and to computation of the Tutte polynomial. Some negative results limiting how far these ideas can be extended to matroids are given at the end.

1.1 Graphs and Hard Problems

We assume that the reader is familiar with the basics of graph theory. We suggest Diestel [14] (or Nešetřil [36] in Czech) for a good introduction. We denote graphs by letters $G, H, \ldots$, and we use the notation $V(G)$, $E(G)$ for the vertex and edge sets of a graph G. Usually, we consider simple graphs (i.e. without loops or multiple edges), and so we use the shortcut $e = uv$ for an edge e with endvertices u, v in a graph. The *degree* of a vertex in a graph is the number of incident edges. We use the notation $G - v$ and $G - e$ for deleting a vertex and an edge from the graph G.

A graph is *connected* is every two vertices are joined by a path. A graph without cycles is a *forest*, and a connected forest is a *tree*. We usually denote trees by letters $T, U, \ldots$ A vertex of degree 1 is called a *leaf*. If T is a tree, and $r \in V(T)$ is a vertex of the tree, then the pair (T, r) is called a *rooted tree*, where r is called the *root*. If the root r of the rooted tree (T, r) is obvious from the context, we shortly denote the rooted tree also by T. A vertex of a tree is also called a *node*. If $v \in V(T)$ is a vertex of the rooted tree (T, r), then the neighbour of v on the unique path joining v to the root r is called the *parent* of v. If u is the parent of v, then v is a *son* of u.

A rooted tree is called *binary* if every non-leaf vertex has two sons. A rooted tree is called *sub-binary* if every vertex has at most two sons. A rooted (sub-)binary tree is *ordered* if the sons of each vertex are exclusively labelled "left" or "right". (If there is only one son, it may be either "left" or "right".) This concept is in a natural correspondence with the usual way of drawing rooted trees in the plane. If $e = uv$ is an edge of (T, r) where u is the parent of v, then

the connected component of $T - e$ not containing u is called a rooted subtree (with root v) of u in T. For an ordered tree (T, r), we naturally call the rooted subtrees of a node u left and right.

Soon after the theory of problem tractability and *NP*-completeness emerged in the seventies, many simply-defined graph questions were shown to be *NP*-complete. We refer interested reader to [17], and to the next section for basic formal definitions. Informally speaking, an *NP*-complete problem likely has no significantly better algorithm than the "brute force" one.

Here we would like to mention few *NP*-complete problems of graphs:

3-colourability: Is there a proper vertex colouring (in which no two adjacent vertices get the same colour) of the given graph by 3 colours?

Colourability: For a given c, is there a proper vertex colouring of the given graph by c colours?

Vertex cover: For a given c, is there a subset U of c vertices in the given graph such that all edges of the graph are incident with a vertex in U?

Hamiltonian cycle: Is there a cycle passing through all vertices in the given graph?

Crossing number: For a given c, is there a plane embedding of the given graph with at most c edge-crossings?

Tree-width: For a given t, is tree-width of the given graph at most t? See Section 1.4.

1.2 Bits of Computing

In this section we briefly introduce few well-known terms and concepts in computing theory. We regard a *computational problem* (shortly a problem) as a mapping from input instances to solutions. A *decision problem* is such that an answer YES or NO is expected. Equivalently, a decision problem is simply a subset of accepted input instances. We use the notation $O(f(n))$ to describe functions $g(n) \in O(f(n))$ such that $g(n) \leq A \cdot f(n) + B$ for some constants A, B and all $n \in \mathbb{N}$. Similarly, we write $g(n) \in \Omega(f(n))$ if $g(n) \geq A \cdot f(n) - B$ for some constants A, B and all $n \in \mathbb{N}$.

We denote by P the class of all decision problems that have algorithms computing the correct solution in polynomial time with respect to the length of input. We denote by *NP* the class of all decision problems for which the answer

YES can be verified in polynomial time. Clearly, $P \subseteq NP$, and the problem whether $P \neq NP$ is one of the major problems in contemporary mathematics. We similarly denote by $\#P$ the class of all enumeration problems in which the structures being enumerated are recognizable in polynomial time. Notice that $NP \subseteq \#P$ since problems in NP have (positive) proofs of polynomial length, and one may enumerate all such proofs in $\#P$.

We say that a problem X is NP-hard if a polynomial-time algorithm for X would provide polynomial-time algorithms for all problems in NP; and that X is NP-complete if X is NP-hard and X belongs to NP. We analogously define $\#P$-hard and $\#P$-complete problems. A more formal approach to this concept, using Turing machines and polynomial reductions, can be found in Garey-Johnson [17], Valiant [50], and other classical literature. We skip the details here to keep our work reasonably short. For example, the problem of satisfiability of boolean formulas is NP-complete, and counting the satisfying assignments for such formulas is $\#P$-complete.

We are now going to shortly present another interesting part of computing theory — finite automata. We refer the reader to Hopcroft-Ullmann [32] for an introduction to automata theory. An automaton presents a primitive way of sequential computing with a fixed-size memory.

Formally, a deterministic finite *automaton* is a quintuple $\mathcal{A} = (K, \Sigma, \delta, q_0, F)$, where K is a finite set of states, Σ is a finite alphabet, $q_0 \in K$ is an initial state, and $F \subseteq K$ is a set of accepting states. A function $\delta : K \times \Sigma \to K$ is the transition function of $\mathcal{A}$. Let Σ^* be the set of all words made from symbols of the alphabet Σ. Suppose that $v = wa \in \Sigma^*$ is a nonempty word ending with the symbol a, and let the function $eval_{\mathcal{A}}$ for $\mathcal{A}$ be defined recursively by $eval_{\mathcal{A}}(\emptyset) = q_0$, and $eval_{\mathcal{A}}(v) = \delta(eval_{\mathcal{A}}(w), a)$. Then a word v is *accepted* by $\mathcal{A}$ if $eval_{\mathcal{A}}(v) \in F$.

For our research we need a slightly extended type of an automaton that reads "words" given as labelled trees instead of sequences. Let Σ be a finite alphabet. We denote by Σ^{**} the class of rooted ordered sub-binary trees with vertices labelled by symbols from Σ. When defining tree automata, we follow [16, Section 6.1], but we restrict our attention only to sub-binary trees. (This does not mean a significant loss of generality.)

Definition. A deterministic finite leaf-to-root *tree automaton* is a quintuple $\mathcal{A} = (K, \Sigma, \delta_t, q_0, F)$, where a set of states K, an alphabet Σ, an initial state q_0, and accepting states F are like in a classical automaton. The transition function δ_t is defined as a mapping from $K \times K \times \Sigma$ to K. Let the function $t\text{-}eval_{\mathcal{A}}$ for $\mathcal{A}$ be defined recursively by $t\text{-}eval_{\mathcal{A}}(\emptyset) = q_0$, and $t\text{-}eval_{\mathcal{A}}(T) = \delta_t(t\text{-}eval_{\mathcal{A}}(T_l), t\text{-}eval_{\mathcal{A}}(T_r), a)$ for $T \in \Sigma^{**}$, where T_l, T_r is the left and right, respectively, rooted subtree of the root of T, and where a is the root symbol. A tree T is accepted by $\mathcal{A}$ if $t\text{-}eval_{\mathcal{A}}(T) \in F$.

A tree language $L \subseteq \Sigma^{**}$ is *finite state* if it is accepted by a finite tree automaton. It is clear that any finite tree automaton can be emulated by a computer in linear time.

1.3 Parametrized Complexity

Many natural problems, appearing in practical applications, are NP-hard, and there is little hope that fast deterministic algorithms could ever be found for them. However, we still need to solve such hard problems in practice. Traditional approaches include approximation or randomized algorithms, or combinations of both. A new promising approach emerged in the past decade: We try to extract the "hard core" of a problem, and to describe it in terms of a suitable "parameter" which (we hope) will be small in practical applications. If we succeeded, we might get an algorithm for this problem which is polynomial when the parameter is fixed, and practical for small values of the parameter.

As a folklore example of the above idea, we present the next simple algorithm for the vertex cover problem (see Section 1.1). Given a graph G and a number k, we ask whether there is a vertex subset $U \subseteq V(G)$, $|U| \leq k$ such that each edge of G is incident with some vertex in U. The algorithm follows:

- First we set U_1 to be the set of all vertices of degree greater than k in G. Clearly, if $v \in V(G)$ has degree greater than k, then not all neighbours of v may belong to U, and so $v \in U$. Thus $U_1 \subseteq U$.
- Let $G_1 = G - U_1$. Then the vertex degrees in G_1 are at most k by definition. Hence if $|V(G_1)| > k(k+1)$, then we cannot even cover the edges of G_1 by k vertices, and the answer is NO.
- Finally, we have a graph G_1 with at most $k(k+1)$ vertices. We check using brute force whether there is a set $U_2 \subseteq V(G_1)$, $|U_2| \leq k - |U_1|$ covering all edges of G_1. If so, then $U = U_1 \cup U_2$.

It is easy to see that the algorithm runs in time $O(\,|V(G)| + |E(G)|\,) + f(k)$ where f is an exponential function. We remark that this time is obviously better for a fixed k than the brute force algorithm for vertex cover with running time $O(\,|V(G)|^k)$.

When speaking about *parametrized complexity*, we closely follow Downey and Fellows [16]. Here we present the basic definition of parametrized tractability. For simplicity, we restrict the definition to decision problems, although an extension to computational problems is straightforward. Let Σ be the input alphabet. A *parametrized problem* is an arbitrary subset $A^p \subseteq \Sigma^* \times \mathbb{N}$. For an instance $(x, k) \in A^p$, we call k the *parameter* and x the input for the problem. (The parameter is sometimes implicit in the context.)

Definition. We say that a parametrized problem A^p is *(nonuniformly) fixed-parameter tractable* if there is a sequence of algorithms $\{\mathcal{A}_i : i \in \mathbb{N}\}$, and a constant c; such that $(x, k) \in A^p$ iff the algorithm $\mathcal{A}_k$ accepts (x, k), and that the running time of $\mathcal{A}_k$ on (x, k) is $O(|x|^c)$ for each k.
We say that a parametrized problem A^p is *uniformly fixed-parameter tractable* if there is an algorithm $\mathcal{A}$, a constant c, and an arbitrary function $f : \mathbb{N} \to \mathbb{N}$; such that $(x, k) \in A^p$ iff the algorithm $\mathcal{A}$ accepts (x, k), and that the running time of $\mathcal{A}$ on (x, k) is $O(f(k) \cdot |x|^c)$.

Naturally, a parametrized problem A^p corresponds to an ordinary problem $A = \{x \in \Sigma^* : \exists k\ (x, k) \in A^p\}$. On the other hand, an ordinary problem may have several natural parametrized versions respecting different parameters. We remark that the parameter is formally a natural number, but that may encode arbitrary finite structures in a standard way.

Among the graph problems given in Section 1.1, we have seen that Vertex cover is uniformly fixed-parameter tractable with respect to the parameter c. Similarly, it is known that Crossing number is uniformly fixed-parameter tractable as well, but that is beyond the scope of our work. Conversely, the Colourability problem is likely not fixed-parameter tractable with respect to c since it is NP-complete already for a fixed value of $c = 3$.

1.4 Graph Tree-Width

It has been noticed by various researchers in the past that problems otherwise hard could often be efficiently solved for trees and for "tree-like" graphs. The formal definition of tree-width of a graph was first given in the pioneer mathematical work of Robertson and Seymour [40] on graph minors, in a different context. Nevertheless, the great computational significance of this definition was soon recognized. We suggest [5] for a short and nice overview.

Definition. Let G be a graph. A *tree-decomposition* of G is a pair (T, β), where T is a tree and $\beta : V(T) \to 2^{V(G)}$ is a mapping satisfying the following:

- For each edge $e = uv \in E(G)$, there is $x \in V(T)$ such that $\{u, v\} \subseteq \beta(x)$.
- (IP) If $x \in V(T)$, and if $y, z \in V(T)$ are two vertices in distinct components of $T - x$, then $\beta(y) \cap \beta(z) \subseteq \beta(x)$.

The width of the decomposition (T, β) of G equals the maximal value of $|\beta(x)| - 1$ over all $x \in V(T)$. The *tree-width* of the graph G is the smallest width over all tree-decompositions of G.

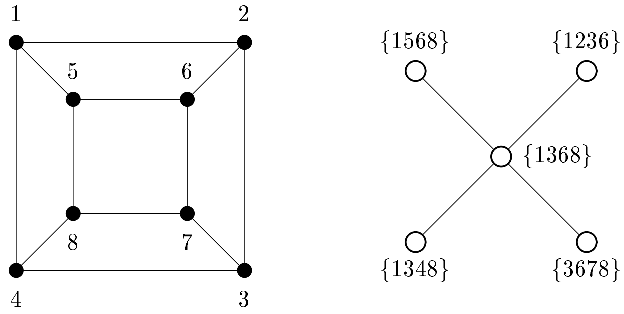


Figure 1.1: An example of a width-3 tree-decomposition of the cube graph.

The vertex subsets $\beta(x) \subseteq V(G)$ for $x \in V(T)$ are called "bags". The condition (IP) is called an interpolation property. Informally speaking, the concept of tree-width measures "how far" is a graph from being a tree in a global sense. It is easy to see that tree-width is non-increasing for subgraphs. See an example in Figure 1.1.

As noted above, graph structures of bounded tree-width often come out in practical situations, but usually not with a given suitable tree-decomposition. So in order to take advantage of bounded tree-width, we need a pre-processing algorithm that finds a (near) optimal tree-decomposition of a given graph rather quickly. Unfortunately, the problem whether a given graph has tree-width at most t is NP-complete [4]. On the other hand, when t is fixed, there is a linear-time parametrized algorithm for finding the optimal tree-decomposition by Bodlaender [6]. (This algorithm is quite involved.) Hence graph tree-width is a uniformly fixed-parameter tractable problem.

Recall the problems from Section 1.1. It is reasonably easy to see the 3-colourability or Hamiltonian cycle problems can be solved efficiently for graphs of a fixed tree-width using standard techniques of dynamic programming. The same applies to Vertex cover. The question of complexity of the Crossing number problem for graphs of fixed tree-width is still open. Conversely, there exist (few) problems which are NP-complete even for trees, such as the bandwidth problem.

In Chapter 6, we present a deep general result of Courcelle [11] showing that all graph properties definable in the monadic second-order logic are recognizable in linear time for graphs of fixed tree-width. This result is the main motivation for our research on matroids, and we shall prove its extension to represented matroids.

Chapter 2

Basics of Matroids

Matroids were introduced by Whitney in the founding paper [54] in 1935 as abstract objects capturing the fundamental properties of dependence that are common to graphs and matrices. Other attempts to extract the essence of dependence appeared at about the same time in works of van der Waerden, Birkhoff and MacLane.

Major development in matroid theory appeared after the war in numerous works of Tutte.[1] The main motivation of Tutte's matroid research was to extend as much graph structural properties to matroids as possible. This, indeed, proved to be a successful idea, and had got many followers. (One may consider Tutte as the father of modern structural matroid theory.) But, surprisingly, the related complexity questions were mostly left behind, so far. In this work we are going to fill a part of this gap.

This chapter gives basic overview of matroids and their properties. We focus mainly on matroids represented over finite fields. We define the notion of an $\mathbb{F}$-represented matroid, and our definition of a boundaried represented matroid. The connectivity and branch-width related definitions are in the next chapter.

2.1 Definition of a Matroid

We follow Oxley [38] for matroid terminology and basic results.

Definition. A *matroid* is a pair $M = (E, \mathcal{B})$ where $E = E(M)$ is the ground set of M (the elements of M), and $\mathcal{B} \subseteq 2^E$ is a nonempty collection of *bases* of M. Moreover, matroid bases satisfy the "exchange axiom"; if $B_1, B_2 \in \mathcal{B}$ and $x \in B_1 - B_2$, then there is $y \in B_2 - B_1$ such that $(B_1 - \{x\}) \cup \{y\} \in \mathcal{B}$.

[1] We note, as a curiosity, an earlier remarkable mathematical achievement of Tutte — a success so great that it had been kept top secret for more than 50 years. We cite: *In January 1941, Tutte went to Bletchley Park, the now legendary organization of code-breakers of Britain. The Bletchley code-breakers, among whom Alan Turing was prominent, had had success in deciphering Enigma codes. But that success was with the naval and air force versions; the army version of Enigma proved to be resistant to analysis. ... Tutte's great contribution was to uncover, from samples of the messages alone, the structure of the machines which generated these codes. ... They had completely recreated the machine without ever having seen one. Tony Sale, who first described this work in a 1997 article in New Scientist, characterized it as the "greatest intellectual feat of the whole war".* From [W.T. Tutte (1917–2002), `http://www.math.uwaterloo.ca/CandO_Dept/Canmath.shtml`].

Subsets of bases are called *independent sets*, and the remaining sets are *dependent*. Clearly, all bases in a matroid have the same cardinality, called the *rank* of the matroid. The minimal dependent sets in M are called *circuits*. The *rank function* $r_M : 2^E \to \mathbb{N}$ has the following meaning: $r_M(A) = n$ for a set $A \subseteq E$ iff the maximal independent subset of A in M has exactly n elements.

If G is a graph, then its *cycle matroid* on the ground set $E(G)$ is denoted by $M(G)$. The bases of $M(G)$ are the spanning forests of G, and circuits of $M(G)$ are the cycles of G. In fact, a lot of matroid terminology is inherited from graphs. Another typical example of a matroid is a finite set of vectors with usual linear dependency. These two matroid examples actually inspired the original definition of a matroid by Whitney [54].

The *dual* matroid M^* of M is defined on the same ground set E, and the bases of M^* are the set-complements of the bases of M. (Matroid duality corresponds to topological duality of planar graphs.) An element e of M is called a *loop* (a *coloop*), if $\{e\}$ is dependent in M (in M^*). The matroid $M \setminus e$ obtained by *deleting* a non-coloop element e is defined as $(E - \{e\}, \mathcal{B}^-)$ where $\mathcal{B}^- = \{B : B \in \mathcal{B}, e \notin B\}$. The matroid M/e obtained by *contracting* a non-loop element e is defined using duality $M/e = (M^* \setminus e)^*$. (This corresponds to contracting an edge in a graph, or to projecting a vector of a matrix.) Contracting a loop means deleting it, and vice versa.

A *minor* of a matroid is obtained by a sequence of deletions and contractions of elements. A matroid family $\mathcal{M}$ is *minor-closed* if $M \in \mathcal{M}$ implies that all minors of M are in $\mathcal{M}$. A matroid N is called an *excluded* minor for a minor-closed family $\mathcal{M}$ if N is a minimal matroid not in $\mathcal{M}$ in the minor order.

2.2 Projective Geometries and Matroid Representations

We now turn our attention to matroids represented by matrices over a fixed commutative field $\mathbb{F}$. We usually consider finite fields in our work, and we use the standard notation $GF(q)$ to denote the (unique) finite field with q elements, where q is a power of a prime. When q itself is a prime, then the field $GF(q) \simeq \mathbb{Z}_q$ is obtained by using normal arithmetic modulo q.

Definition. A *representation* of a matroid M is a matrix $\boldsymbol{A}$ whose columns correspond to the elements of M, and maximal linearly independent subsets of columns form the bases of M. We denote by $M(\boldsymbol{A})$ the matroid represented by a matrix $\boldsymbol{A}$.

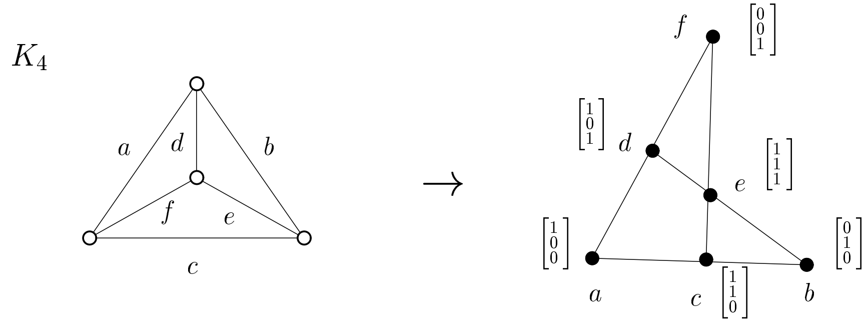


Figure 2.1: An example of a vector representation of the matroid $M(K_4)$. The matroid elements are depicted by dots, and their (linear) dependency is shown using lines. The vectors representing elements are included as well.

For example, the cycle matroid of a graph can be easily represented by a matrix over any field $\mathbb{F}$ (Figure 2.1). We remark that matroids represented over a finite field play an important role in structural matroid theory, similar to the role that graphs embedded on a surface play in structural graph theory.

We say that a rank-r matrix $\boldsymbol{A}$ is in the *standard form* if it has the shape $\boldsymbol{A} = [\boldsymbol{I}_r \mid \boldsymbol{A}']$. The elements of $M = M(\boldsymbol{A})$ corresponding to the unit matrix $\boldsymbol{I}_r$ form a basis B of M, and the matrix $\boldsymbol{A}'$ is called a *reduced representation* of M displaying the basis B. Naturally, the columns of $\boldsymbol{A}'$ correspond to the elements in $E(M) - B$, and the rows of $\boldsymbol{A}'$ correspond to the elements in B.

We add few words about picturing matroids: The matroid elements are shown as dots (not to be mistaken with graph vertices), and the dependent triples are shown using lines. In fact, one line presents a maximal rank-2 subset of the matroid (a flat). Such convention works well for matroids of rank at most 3, but additional tricks must be used to depict matroids of higher ranks. It is often not even possible to show a matroid of rank 4 or higher in an obvious picture.

We denote by $PG(n, \mathbb{F})$ the *projective geometry (space)* obtained from the vector space $\mathbb{F}^{n+1}$. See [38, Section 6.1,6.3] for an overview of projective spaces and of (in)equivalence of matroid representations. For a set $X \subseteq PG(n, \mathbb{F})$, we denote by $\langle X \rangle$ the span (affine closure) of X in the space. The (projective) rank $r(X)$ of X is the maximal cardinality of a linearly independent subset of X. A projective transformation is a mapping between two projective spaces over $\mathbb{F}$ that is induced by a linear transformation between the underlying vector spaces. Clearly, the matroid $M(\boldsymbol{A})$ represented by a matrix $\boldsymbol{A}$ is unchanged when columns are scaled by non-zero elements of $\mathbb{F}$. Hence we may view a

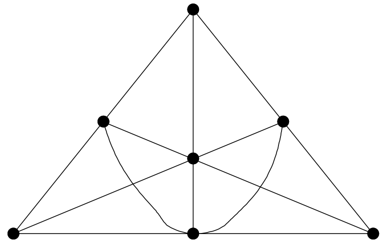


Figure 2.2: Finite projective geometry $PG(2, GF(2))$ over the binary field $GF(2)$ (otherwise known as the Fano plane). The seven elements are depicted as dots and the seven lines as lines in the picture. Any $GF(2)$-represented (also called binary) matroid of rank at most 3 is a multiset of elements of $PG(2, GF(2))$ by definition.

loopless matroid representation $M(\boldsymbol{A})$ as a multiset of points in the projective space $PG(n, \mathbb{F})$ where n is the rank of $M(\boldsymbol{A})$. See an illustration in Figure 2.2.

We call a finite multiset of points in a projective space over $\mathbb{F}$ a *point configuration*. (Multiple-points represent a parallel class of matroid elements.) We say that two point configurations P_1, P_2 in projective spaces over $\mathbb{F}$ are *equivalent* if there is a non-singular projective transformation between the projective spaces that maps P_1 onto P_2 bijectively. One may think that we do not have to include the word "bijectively" in the previous sentence since non-singular projective transformations are always injective on the points, but, in fact, we have to do this to handle multiple-elements in multisets. We represent a loop in a point configuration by the empty subspace $\emptyset$. In the above equivalence, loops are mapped only to loops.

Definition. We define an $\mathbb{F}$-*represented matroid* to be an equivalence class of point configurations over $\mathbb{F}$. (In the above defined meaning of equivalence.)

Obviously, all point configurations in one equivalence class belong to the same isomorphism class of matroids. Thanks to the definition, we do not have to bother with matroid isomorphism or equivalence any more, and we simply speak about equality between represented matroids. When we want to deal with an $\mathbb{F}$-represented matroid, we actually pick an arbitrary point configuration from the equivalence class.

Standard matroidal terms are inherited from matroids to represented matroids. To clearly distinguish between a matroid and a represented matroid, we sometimes add the adjective *abstract* to the former one. We do not label points in our configurations, and so we are dealing with matroids on unlabelled element sets. Notice that $\mathbb{F}$-automorphisms are not allowed in our definition of equivalence, and hence the equivalence on labelled point configurations corresponds to what is known as a strong equivalence of matroid representations [38, Chapter 6].

Since several inequivalent representations over a fixed field may exist for the same matroid, one abstract matroid may have (and usually has) more than one distinct $\mathbb{F}$-represented matroids. As a clear example, we present in Figure 2.3 two point configurations representing the same 9-element rank-3 matroid which are not "equivalent" in any reasonable geometric meaning of equivalence. An exception is with $GF(2)$-representable (binary) matroids which are uniquely representable over any field. Read more in [38].

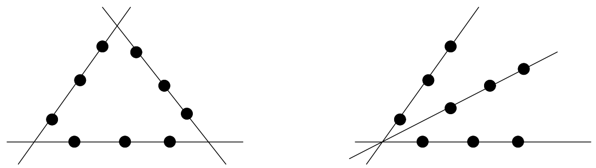


Figure 2.3: Two non-equivalent representations of a 9-element rank-3 matroid.

2.3 Boundaried Represented Matroids

For our work we need to build up large matroids from small pieces. Such task is quite easy for graphs — for an integer k, we mark a k-element subset of graph vertices as the "boundary", and we "glue" two graphs together by identifying their k boundary vertices in order. However, the analogous notion of a "matroid k-sum" is not well defined for general matroids if $k > 2$. We overcome this obstacle by using the geometric structure of represented matroids instead.

In a point configuration P we distinguish an arbitrary projective subspace ∂ as the boundary for P. Actually, we determine ∂ by giving one of its geometric bases as additional points. That will be formalized soon. We glue two point configurations with the boundaries of the same rank k by identifying the points spanning their boundaries in order. This natural geometric definition works well even for represented matroids as equivalence classes of point configurations.

All matroids throughout this section are $\mathbb{F}$-represented for some fixed field $\mathbb{F}$. Hence, for simplicity, if we say "a matroid", then we mean an $\mathbb{F}$-represented matroid. If we speak about a projective space, we mean a projective geometry over the field $\mathbb{F}$, including the empty subspace $\emptyset$ for representing loops. Our definition of a boundaried matroid follows.

Definition. A pair $\bar{N} = (N, \delta)$ is called a *t-boundaried (represented) matroid* if the following holds: $t \geq 0$ is an integer, N is a represented matroid, and $\delta : [1, t] \to E(N)$ is an injective mapping such that $\delta([1, t])$ is an independent set in N.

We call $J(\bar{N}) = E(N) - \delta([1, t])$ the *internal elements* of $\bar{N}$, elements of $\delta([1, t])$ the *boundary points* of $\bar{N}$, and t the *boundary rank* of $\bar{N}$. We denote by $\partial(\bar{N})$ the boundary subspace spanned by $\delta([1, t])$. In particular, the boundary points are not loops. We say that that the boundaried matroid $\bar{N}$ is *based on* the represented matroid $N \setminus \delta([1, t])$ which is the restriction of N to $J(\bar{N})$. The notion of a t-boundaried represented matroid is similar to "rooted configurations" defined in [19], but it is more flexible. The basic operation we will use is the *boundary sum* described in the next definition.

Definition. Let $\bar{N}_1 = (N_1, \delta_1)$, $\bar{N}_2 = (N_2, \delta_2)$ be two t-boundaried represented matroids. We denote by $\bar{N}_1 \bar{\oplus} \bar{N}_2 = N$ the represented matroid defined as follows: Let Ψ_1, Ψ_2 be projective spaces such that the intersection $\Psi_1 \cap \Psi_2$ has rank exactly t. Suppose that, for $i = 1, 2$, $P_i \subset \Psi_i$ is a point configuration representing N_i, such that $P_1 \cap P_2 = \delta_1([1, t]) = \delta_2([1, t])$, and $\delta_2(j) = \delta 1(j)$ for $j \in [1, t]$. Then N is the matroid represented by $(P_1 \cup P_2) - \delta_1([1, t])$.

Informally, the boundary sum $\bar{N}_1 \bar{\oplus} \bar{N}_2 = N$ on the ground set $E(N) = J(\bar{N}_1) \dot{\cup} J(\bar{N}_2)$ is obtained by gluing the representations of N_1 and N_2 on a common subspace (the boundary) of rank t, so that the boundary points of both are identified in order and then deleted. Keep in mind that a point configuration is considered as a multiset. It is a matter of elementary linear algebra to verify that the boundary sum is well defined with respect to equivalence of point configurations. Clearly, $\bar{\oplus}$ is a commutative operation. It is not difficult to prove the following "kind of associativity" of $\bar{\oplus}$ (see Figure 2.4):

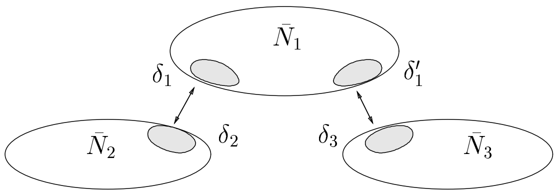


Figure 2.4: Illustration to Lemma 2.3.1.

Lemma 2.3.1. *Let $\bar{N}_i = (N_i, \delta_i)$, $i = 1, 2, 3$ be t-boundaried represented matroids. Suppose that $\delta'_1 : [1, t] \to E(N_1)$ is a mapping such that $\bar{N}'_2 = (\bar{N}_1 \bar{\oplus} \bar{N}_2, \delta'_1)$ is a t-boundaried matroid. Then $\bar{N}'_3 = (N'_3, \delta_1)$ where $N'_3 = (N_1, \delta'_1) \bar{\oplus} \bar{N}_3$ is a t-boundaried matroid as well. Moreover, $\bar{N}'_2 \bar{\oplus} \bar{N}_3 = \bar{N}_2 \bar{\oplus} \bar{N}'_3$.* ∎

We write "$\leq t$-boundaried" to mean t'-boundaried for some $0 \leq t' \leq t$. We now define a composition operator (over the field $\mathbb{F}$) which will be used to generate large boundaried matroids from small pieces.

Definition. A *$\leq t$-boundaried composition* operator is defined as a quadruple $\odot = (R, \gamma_1, \gamma_2, \gamma_3)$, where R is a represented matroid, $\gamma_i : [1, t_i] \to E(R)$ is an injective mapping for $i = 1, 2, 3$ and some fixed $0 \leq t_i \leq t$, each $\gamma_i([1, t_i])$ is an independent set in R, and $(\gamma_i([1, t_i]) : i = 1, 2, 3)$ is a partition of $E(R)$.
The $\leq t$-boundaried composition operator $\odot$ is a binary operator applied to a t_1-boundaried represented matroid $\bar{N}_1 = (N_1, \delta_1)$ and to a t_2-boundaried represented matroid $\bar{N}_2 = (N_2, \delta_2)$. The result of the composition is a t_3-boundaried represented matroid $\bar{N} = (N, \gamma_3)$, written as $\bar{N} = \bar{N}_1 \odot \bar{N}_2$, where a matroid N is defined using boundaried sums: $N' = \bar{N}_1 \bar{\oplus} (R, \gamma_1)$, $N = (N', \gamma_2) \bar{\oplus} \bar{N}_2$.

Speaking informally, a boundaried composition operator is a bounded-rank configuration with three boundaries distinguished by $\gamma_1, \gamma_2, \gamma_3$, and with no other internal points. The meaning of a composition $\bar{N} = \bar{N}_1 \odot \bar{N}_2$ is that, for $i = 1, 2$, we glue the represented matroid N_i to R, matching $\delta_i([1, t_i])$ with $\gamma_i([1, t_i])$ in order. The result is a t_3-boundaried matroid $\bar{N}$ with boundary $\gamma_3([1, t_3])$. One may shortly write the result as $\bar{N} = ((\bar{N}_1 \bar{\oplus} (R, \gamma_1), \gamma_2) \bar{\oplus} \bar{N}_2, \gamma_3)$. Notice that, in general, there are more than one boundaried composition operators with the same ranks. For reference we denote by $t_i(\odot) = t_i$, by $R(\odot) = R$, and by $\gamma_i(\odot) = \gamma_i$. We now show that the meaning of the operands and the resulting boundary in a boundaried composition operator could be easily exchanged.

Lemma 2.3.2. *Suppose that $\odot = (R, \gamma_1, \gamma_2, \gamma_3)$ and $\odot' = (R, \gamma_1, \gamma_3, \gamma_2)$ are $\leq t$-boundaried composition operators, and that $\bar{N}_i$, $i = 1, 2, 3$ are $t_i(\odot)$-boundaried represented matroids. Then $(\bar{N}_1 \odot \bar{N}_2) \bar{\oplus} \bar{N}_3 = (\bar{N}_1 \odot' \bar{N}_3) \bar{\oplus} \bar{N}_2$.*

Proof. This is a simple formal manipulation with the previous definition using Lemma 2.3.1:

$$(\bar{N}_1 \odot \bar{N}_2) \bar{\oplus} \bar{N}_3 = ((\bar{N}_1 \bar{\oplus} (R, \gamma_1), \gamma_2) \bar{\oplus} \bar{N}_2, \gamma_3) \bar{\oplus} \bar{N}_3 =$$

$$= ((\bar{N}_1 \bar{\oplus} (R, \gamma_1), \gamma_3) \bar{\oplus} \bar{N}_3, \gamma_2) \bar{\oplus} \bar{N}_2 = (\bar{N}_1 \odot' \bar{N}_3) \bar{\oplus} \bar{N}_2$$ ∎

Chapter 3

Branch-Width and Parse Trees

The notion of branch-width was defined together with tree-width in the work of Robertson and Seymour on graph minors. These two parameters are very closely related, but not the same. In past years, tree-width got more attention among researchers, despite that branch-width seems to be more suitable in structural graph problems. For us, the main advantage of branch-width is that its definition does not refer to graph vertices, and so it has an immediate generalization to matroids.

We give the formal definition of branch-width next. Moreover, we define a corresponding notion of a boundaried parse tree that "builds up" a represented matroid from elementary pieces along a branch-decomposition, using only fixed amount of information at each tree node. (We are inspired by analogous parse trees known for graphs of bounded tree-width.) The new notion of a boundaried parse tree for a represented matroid is central to our research — we use it to link represented matroids with formal languages and tree automata.

Lastly we remark that it is possible to extend also the definition of tree-width to matroids, which was recently shown in the works of Geelen, Whittle and the author. That is, however, not an easy or obvious task at all, and so we do not include it here.

3.1 Connectivity and Branch Decompositions

The connectivity function λ_M of a matroid M is defined for all subsets $A \subseteq E(M)$ by

$$\lambda_M(A) = r_M(A) + r_M(E - A) - r(M) + 1\,.$$

Here $r(M) = r_M(E)$. Notice that $\lambda_M(A) = \lambda_M(E - A)$. A subset $A \subseteq E$ is *k-separating* if $\lambda_M(A) \leq k$. When equality holds here, A is said to be *exactly* k-separating. A partition $(A, E - A)$ is called a *k-separation* if A is k-separating and both $|A|, |E - A| \geq k$. For $n > 1$, the matroid M is called *n-connected* if it has no k-separation for $k = 1, 2, \ldots, n - 1$, and $|E(M)| \geq 2n$. (A connected matroid

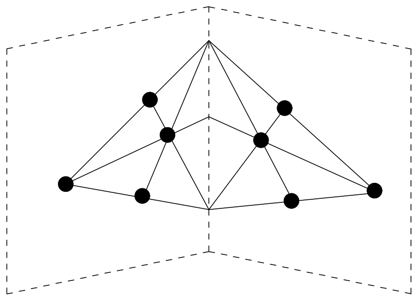


Figure 3.1: An example of a 3-separation in an 8-element matroid of rank 4. The two planes (rank 3) spanning the 4-tuples of elements intersect in a line (rank $2 < 3$).

corresponds to a vertex 2-connected graph.) An example of a 3-separation is in Figure 3.1. Geometrically, the spans of the two sides of a k-separation in a represented matroid intersect in a subspace of rank less than k.

A *sub-cubic tree* is a tree in which all vertices have degree at most three. Let $\ell(T)$ denote the set of leaves of a tree T. The next definition of branch-width for matroids directly extends branch-width of graphs mentioned above.

Definition. Let M be a matroid on the ground set $E = E(M)$. A *branch-decomposition* of M is a pair (T, τ) where T is a sub-cubic tree, and τ is an injection of E into $\ell(T)$, called *labelling*. Let e be an edge of T, and T_1, T_2 be the connected components of $T - e$. We say the e *displays* the partition (A, B) of E where $A = \tau^{-1}(\ell(T_1))$, $B = \tau^{-1}(\ell(T_2))$. The *width* of the edge e in T is $\omega_T(e) = \lambda_M(A) = \lambda_M(B)$.
The width of the branch-decomposition (T, τ) is maximum of the widths of all edges of T, and the *branch-width* of M is the minimal width over all branch-decompositions of M. If T has no edge, then we take its width as 0.

Branch-width of matroids is clearly self-dual. An example of a simple branch-decomposition is presented in Figure 3.2. Branch-width has recently shown to be a very interesting structural matroid parameter. Besides other results, we mention the following recent advances: well-quasi-ordering of matroids of bounded branch-width over finite fields [19], and a size-bound on the excluded minors for matroids of fixed branch-width [18]. The importance of this parameter also in computational complexity is shown later in our work.

We remark that branch-width of a graph is defined analogously, using connectivity function λ_G where $\lambda_G(F)$ is the number of vertices incident both with

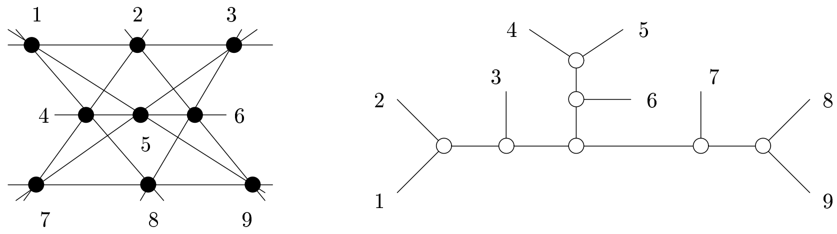


Figure 3.2: An example of a width-3 branch-decomposition of the Pappus matroid.

F and $E(G) - F$. Clearly, $\lambda_{M(G)}(F) \leq \lambda_G(F)$ in a connected graph G, but these numbers may not be equal if the subgraph induced by F or by $E(G) - F$ is not connected. It is still an open conjecture that the branch-width of a graph G is equal to the branch-width of its cycle matroid $M(G)$. The basic relation between graph tree-width and branch-width is given in [41]:

Theorem 3.1.1. (Robertson and Seymour) *Let a graph has tree-width k and branch-width b. Then $b - 1 \leq k \leq \frac{3}{2}b$.*

An important corollary for us is that a class of graphs has bounded tree-width if and only if it has bounded branch-width. Thus it does not matter whether we assume bounded tree-width or bounded branch-width of input graphs in parametrized algorithms (Section 1.4).

3.2 Width of a Matrix

The previous theory allows us to define a notion of a matrix “width” that is invariant on standard matrix row operations, or, in other words, invariant on the projective equivalence of point configurations. Similarly as for graphs and matroids, the hope is that matrices of small “width” are much easier to handle than general matrices, and that fast algorithms exist for problems involving these matrices.

The *branch-width* of a matrix $\boldsymbol{A}$ over $\mathbb{F}$ is the branch-width of the matroid $M(\boldsymbol{A})$. Hence the matrix branch-width of $\boldsymbol{A}$ is not changed when row operations are applied to $\boldsymbol{A}$, and so it is a robust measure of a “complexity” of $\boldsymbol{A}$.

We also remark that our $\mathbb{F}$-represented matroids are in a one-to-one correspondence with linear codes over the field $\mathbb{F}$ since a projective equivalence of point configurations coincides with the standard equivalence of linear codes.

Thus we may define a branch-width of a linear code C as the branch-width of the generator matrix of the code C.

Let us mention that some authors use another matrix "width" parameter defined as follows. For a matrix $\boldsymbol{A} = [a_{i,j}]_{i,j=1}^{n}$, let $G_{\boldsymbol{A}}$ be the graph on the vertex set $\{1,\ldots,n\}$ and the edge set consisting of all $\{i,j\}$ such that $a_{i,j} \neq 0$ or $a_{j,i} \neq 0$. The tree-width of the matrix $\boldsymbol{A}$ is given by the tree-width of the graph $G_{\boldsymbol{A}}$. This definition was, perhaps, inspired by Choleski factorization of sparse symmetric matrices which is related to graph tree-width of the matrix. (See [5] for more details.)

However, this notion of a matrix tree-width is not robust in the above sense — applying a row operation to a matrix $\boldsymbol{A}$ may dramatically change the tree-width of $G_{\boldsymbol{A}}$, while the vector configuration represented by $\boldsymbol{A}$ is still the same. That is why we think that the tree-width of $G_{\boldsymbol{A}}$ is not a good measure of a "complexity" of the matrix / vector configuration $\boldsymbol{A}$. Moreover, the matrices of bounded tree-width are very sparse; and they may be viewed, in fact, just as edge-coloured graphs.

The two above defined matrix "width" parameters are not related to each other. Look at the following example of a matrix $\boldsymbol{D} = \boldsymbol{J}_n - \boldsymbol{I}_n$: The graph $G_{\boldsymbol{D}}$ defined by this matrix is a clique, and so it has tree-width $n-1$. On the other hand, the matroid $M([\boldsymbol{I}|\boldsymbol{D}])$ is 3-connected of branch-width 3 (see on page 57). More similar examples may be given easily.

3.3 Boundaried Parse Trees

Next we present the main outcome of this chapter — that it is possible to generate all represented matroids of branch-width at most $t+1$ using $\leq t$-boundaried composition operators on parse trees.

Let $\bar{\Omega}_t$ denote the *empty* t-boundaried matroid (Ω, δ_0) where $t \geq 0$ and $\delta_0([1,t]) = E(\Omega)$ (t will often be implicit in the context). If $\bar{N} = (N, \delta)$ is an arbitrary t-boundaried matroid, then $\bar{N} \oplus \bar{\Omega}_t$ is actually the restriction of $\bar{N}$ to $E(N) - \delta([1,t])$. Let $\bar{\Upsilon}$ denote the *single-element* 1-boundaried matroid (Υ, δ_1) where $E(\Upsilon) = \{x, x'\}$ are two parallel elements, and $\delta_1(1) = x'$. Let $\bar{\Upsilon}_0$ denote the *loop* 0-boundaried matroid (Υ_0, δ_0) where $E(\Upsilon_0) = \{z\}$ is a loop, and δ_0 is an empty mapping. Let $\mathcal{R}_t^{\mathbb{F}}$ denote the set of all $\leq t$-boundaried composition operators over the field $\mathbb{F}$.

We set $\Pi_t = \mathcal{R}_t^{\mathbb{F}} \cup \{\bar{\Upsilon}, \bar{\Upsilon}_0\}$. Let $T \in \Pi_t^{**}$ be a rooted ordered sub-binary tree with vertices labelled by the alphabet Π_t. Considering a vertex v of T; we set $\varrho(v) = 1$ if v is labelled by $\bar{\Upsilon}$, $\varrho(v) = 0$ if v is labelled by $\bar{\Upsilon}_0$, and $\varrho(v) = t_3(\odot)$ if v is labelled by $\odot$. We call T a *$\leq t$-boundaried parse tree* if the following are true:

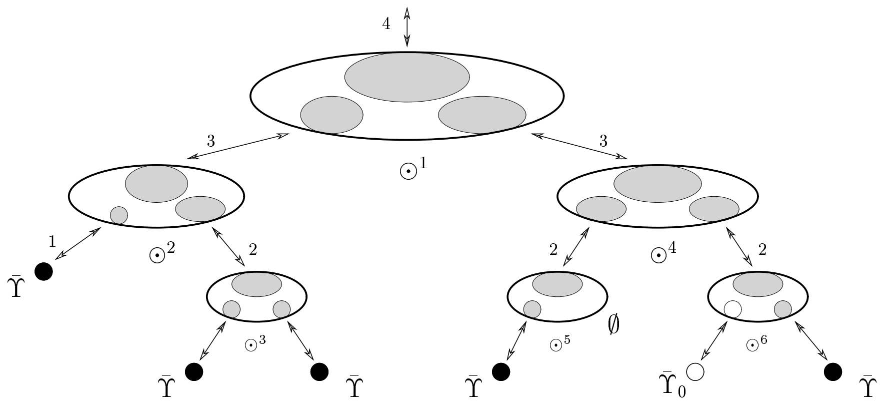


Figure 3.3: An illustration of a boundaried parse tree. The ovals represent composition operators, with shaded parts for the boundaries and edge-numbers for the boundary ranks. (E.g. $\odot^4 = (R^4, \gamma_1^4, \gamma_2^4, \gamma_3^4)$ where $\gamma_1^4, \gamma_2^4 : [1,2] \to E(R^4)$, $\gamma_3^4 : [1,3] \to E(R^4)$.)

– only leaves of T are labelled by $\bar{\Upsilon}$ or $\bar{\Upsilon}_0$;

– if a vertex v of T labelled by a composition operator $\odot^v$ has no left (no right) son, then $t_1(\odot^v) = 0$ ($t_2(\odot^v) = 0$);

– if a vertex v of T labelled by $\odot^v$ has left son u_1 (right son u_2), then $t_1(\odot^v) = \varrho(u_1)$ ($t_2(\odot^v) = \varrho(u_2)$).

Informally, the boundary ranks of composition operators and/or single-element terminals must "agree" across each edge. Notice that $\bar{\Upsilon}$ or $\bar{\Upsilon}_0$ are the only labels from Π_t that "create" elements of the resulting represented matroid $P(T)$ in the next definition. See an illustration example in Figure 3.3.

Definition. Let T be a $\leq t$-boundaried parse tree. The $\leq t$-boundaried represented *matroid* $\bar{P}(T)$ *parsed by* T is recursively defined as follows:

– if T is an empty tree, then $\bar{P}(T) = \bar{\Omega}_0$;

– if T has one vertex labelled by $\bar{\Upsilon}$ (by $\bar{\Upsilon}_0$), then $\bar{P}(T) = \bar{\Upsilon}$ ($= \bar{\Upsilon}_0$);

– if the root r of T is labelled $\odot^r$, and r has the left subtree T_1 and the right subtree T_2 (possibly empty trees), then $\bar{P}(T) = \bar{P}(T_1) \odot^r \bar{P}(T_2)$.

The composition is well defined according to the parse-tree description in the previous paragraph. The represented matroid parsed by T is $P(T) = \bar{P}(T) \oplus \bar{\Omega}$.

Proposition 3.3.1. *The set $\mathcal{R}_t^{\mathbb{F}}$ (and hence also the tree alphabet Π_t) is finite if $\mathbb{F}$ is a finite field.*

Proof. Let $\odot = (R, \gamma_1, \gamma_2, \gamma_3)$ be a $\leq t$-boundaried composition operator. Clearly, the represented matroid R has at most $3t$ elements and rank at most $3t$ as well. (In fact, for further applications one may restrict the set of composition operators to those with R of rank at most $2t$.) So if t and R are fixed, then there are finitely many choices for the mappings $\gamma_1, \gamma_2, \gamma_3$. For finite $\mathbb{F}$ and each value of t, there are finitely many matrices in $\mathbb{F}^{r\times c}$, $r, c \leq 3t$, and hence finitely many choices for the represented matroid R. ∎

This proposition shows why we want to deal with parse trees of matroids represented over finite fields — we need a finite alphabet labelling the parse trees in order to apply automata theory. We remark that, although there are finitely many abstract matroids on at most $3t$ elements up to isomorphism, some of them have infinitely many inequivalent representations over an infinite field. Not only that our approach to the parametrized complexity problems for bounded branch-width requires matroids represented over finite fields; we will show later in Chapter 7 that there are likely no results analogous to ours possible for matroids of branch-width 3 which are represented over an infinite field.

We say that a branch-decomposition (T, τ) of a matroid is *reduced* if all leaves of T are labelled, and all non-leaves have degree exactly 3. It is easy to see that any branch-decomposition can be turned into a reduced one. We say that a t-boundaried represented matroid $\bar{M}$ is spanning if the boundary subspace $\partial(\bar{M})$ is contained in the span $\langle J(\bar{M})\rangle$ of the internal points of $\bar{M}$. We say that a $\leq t$-boundaried parse tree T is *spanning* if, for each nonempty subtree T_1 of T, the boundaried matroid $\bar{P}(T_1)$ is spanning and nonempty.

Theorem 3.3.2. *An $\mathbb{F}$-represented matroid M has branch-width at most $t+1$ if and only if M is parsed by some $\leq t$-boundaried spanning parse tree.*

Proof. Let T be a $\leq t$-boundaried parse tree, and let $M = P(T)$. We define a mapping $\tau : E(M) \to \ell(T)$ by $\tau(x) = v$ if x is the element of M created by the boundaried matroid $\bar{\Upsilon}$ or $\bar{\Upsilon}_0$ labelling v in T. We claim that (T, τ) is a width-$(t+1)$ branch-decomposition of the matroid M. Indeed, consider an edge e of T, and denote by T_e the subtree of T below e and by v the vertex of e farther from the root. Let $F = E(P(T_e))$. Then $\lambda_M(F) \leq r(\partial(\bar{P}(T_e)))+1 \leq t+1$ by the definition of matroid connectivity and the composition operator at the vertex v.

Conversely, suppose that (T, τ) is a reduced width-$(t+1)$ branch-decomposition of a matroid M. We choose an arbitrarily edge e_0 of T, subdivide e_0 with a new vertex r, and make the resulting tree T' rooted at r. We then label each leaf of T' by the boundaried matroid $\bar{\Upsilon}$ or $\bar{\Upsilon}_0$ accordingly. Let P be a point configuration representing M in a projective space Ψ (over $\mathbb{F}$). For an edge e of T', we denote by (A_e, A'_e) the separation of P displayed by e in the branch-decomposition (T, τ). If e was obtained by subdividing e_0, we consider the separation induced by e_0 instead. We denote by Ψ_e the subspace $\Psi_e = \langle A_e\rangle \cap \langle A'_e\rangle$, by B_e an arbitrary basis of Ψ_e, and by $\beta_e : [1, |B_e|\,] \to B_e$ some bijection.

Consider now a vertex v of T' that is not a leaf. Let e_1, e_2 be edges joining v with its left and right sons, respectively, and let e_3 be the edge joining v with its parent. We denote by P_v the point configuration $B_{e_1} \cup B_{e_2} \cup B_{e_3}$, and by R_v the matroid represented by P_v. We take an empty set instead of B_{e_3} if $v = r$. Finally, we define the composition operator $\odot^v = (R_v, \beta_{e_1}, \beta_{e_2}, \beta_{e_3})$. Since (T, τ) is a width-$(t+1)$ branch-decomposition, we know that $\odot^v$ is a $\leq t$-boundaried composition operator. Clearly, for the above labelling, the parse tree T' is spanning, and the represented matroid parsed by T' is M. ∎

Notice an interesting complexity aspect of our parse trees — to describe a n-element rank-r matroid of bounded branch-width representable over a finite field $\mathbb{F}$, it is enough to give only linear amount $O(n)$ of information in the boundaried parse tree, regardless of r. On the other hand, to describe the same matroid by a matrix, one has to provide all $n \cdot r$ entries of the matrix, which typically means to use space of order $\Omega(n^2)$.

Complexity Aspects of Matroid Branch-Width

Petr Hliněný

Part II

COMPLEXITY RESULTS

Chapter 4

Parametrized Algorithm for Branch Decomposition

Our first task is to compute a branch-decomposition of small width for a given represented matroid (if such exists). Actually, following the definitions of Chapter 3, we are going to construct a boundaried parse tree for the given matroid.

When dealing with graphs of bounded tree-width, an optimal tree decomposition can be found in linear time using the algorithm of Bodlaender [6], and an extension [8] of this algorithm can construct the optimal branch decomposition. Unfortunately, those algorithms are quite specific to graphs, and as such they could hardly be extended to matroids.

Nevertheless, we have been inspired by a basic principle of Bodlaender's algorithm — to construct a resulting decomposition iteratively, each time using the bounded decomposition from the previous step for faster computations in the next step.

The content of this chapter is taken from our original paper [29].

4.1 Overview of the Construction

We start with an overview of our algorithm for construction of a matroid parse tree. We assume that a matroid M is given by a matrix $\boldsymbol{A} \in \mathbb{F}^{r\times n}$ of rank r over a finite field $\mathbb{F}$. The size of M is expressed in terms of the number n of elements, i.e. the number of columns of $\boldsymbol{A}$. Clearly, $r \leq n$. Additionally, there are two *parameters* – a finite field $\mathbb{F}$ and an integer $t > 0$. These parameters form a separate part of the input in our algorithm, but they are considered as constants for the purpose of complexity analysis of the algorithm. (See parametrized complexity in Section 1.3.)

We call a labelled tree T a *partial $\leq$t-boundaried parse tree* if T satisfies all properties of the parse tree definition (page 19) except that some leaves of T may be labelled by arbitrary t'-boundaried represented matroids for $t' \leq t$. The matroid $\bar{P}(T)$ parsed by T is defined analogously to ordinary parse trees.

Notice that a 0-boundaried represented matroid is essentially an ordinary represented matroid, and that the boundary sum of 0-boundaried represented matroids is the ordinary direct sum of matroids. If M is a free matroid (with independent ground set), then any branch decomposition of M has width 1 by definition, and so it gives a 0-boundaried parse tree for M. Let $M \restriction X$ denote the restriction of a represented matroid M to a subset of elements $X \subseteq E(M)$.

Algorithm 4.1.1. Computing a spanning boundaried parse tree of a matroid:

Parameters: A given finite field $\mathbb{F}$, and an integer $t \geq 1$.

Input: A matrix $\boldsymbol{A} = [\boldsymbol{I}_r \,|\, \boldsymbol{A}'] \in \mathbb{F}^{r \times n}$ given in the standard form over $\mathbb{F}$, such that the matroid $M(\boldsymbol{A})$ represented by $\boldsymbol{A}$ has branch-width at most $t+1$.

1. Let $M = M(\boldsymbol{A})$, and let $E(M) = B \cup D$ where a basis B marks the columns of $\boldsymbol{I}_r$ and D the columns of $\boldsymbol{A}'$. We initially set $X = B$, and T to be an arbitrary 0-boundaried parse tree for the free matroid $M \restriction B$.
2. For an arbitrary element $f \in D - X$, we set $X = X \cup \{f\}$, and we add a new single-element leaf representing f in $M \restriction X$ to the parse tree T.
3. If the width of the parse tree T exceeds, say, $10t$, or if $X \supseteq D$, then we compute a new spanning $\leq 3t$-boundaried parse tree T' for $M \restriction X$ using T:
   (a) We start with T' equal to the trivial 0-boundaried partial parse tree having one vertex labelled by $M \restriction X$.
   (b) Let ℓ be a leaf of T' labelled by a boundaried matroid $\bar{N}$ with more than one internal elements. If the boundary rank of $\bar{N}$ is less than $3t$, then we choose an arbitrary internal element e in $\bar{N}$. We append to ℓ in T' a new leaf ℓ_1 representing e, another new leaf ℓ_2 labelled by $\bar{N} \setminus e$, and we re-label ℓ with the corresponding composition operator.
   (c) If the boundary rank of $\bar{N}$ equals $3t$, then there are two $\leq 3t$-boundaried matroids $\bar{N}_1, \bar{N}_2$ such that $\bar{N} = \bar{N}_1 \odot \bar{N}_2$ for some $\leq 3t$-boundaried composition operator $\odot$. Using the decomposition T, we can find $\bar{N}_1, \bar{N}_2$ and $\odot$ efficiently. Then we append to ℓ in T' two new leaves ℓ_1, ℓ_2 labelled by $\bar{N}_1, \bar{N}_2$, respectively, and we re-label ℓ with $\odot$.
   (d) We repeat steps 3b–3c until T' is an ordinary parse tree.

   Finally, we set $T = T'$.
4. We repeat steps 2–3 until $X = D$.

Output: A spanning $\leq 3t$-boundaried parse tree $T \in \Pi_{3t}^{**}$ such that the represented matroid $P(T)$ parsed by T is equal to $M(\boldsymbol{A})$.

Remark. If the above promise that the matroid $M(\boldsymbol{A})$ has branch-width at most $t+1$ is false, then the step (3c) may fail to find $\bar{N}_1$ and $\bar{N}_2$. In such a case, Algorithm 4.1.1 ends up with an error. The algorithm may be formulated in a way that it always finds a spanning $\leq 3t$-boundaried parse tree T for $M(\boldsymbol{A})$ if branch-width of $M(\boldsymbol{A})$ is at most $t+1$, but the output is (possibly) empty if branch-width of $M(\boldsymbol{A})$ exceeds $t+1$.

Remark. Another note concerns the given finite field $\mathbb{F}$: A finite field is uniquely determined by the number of its elements $q = |\mathbb{F}|$. Moreover, it is easy to construct addition and multiplication tables for $\mathbb{F}$ algorithmically from given q.

We claim that Algorithm 4.1.1 finds the $\leq 3t$-boundaried parse tree T for $M(\boldsymbol{A})$ in time $O(n^3)$, where n is the number of columns of $\boldsymbol{A}$. If the matrix $\boldsymbol{A}$ is not in the standard form, we can easily get the standard form in time $O(n^3)$. The step (2) can be implemented in time $O(nr) \leq O(n^2)$. Each iteration of the cycle in step (3) can be implemented in time $O(n)$, and there are at most $n-1$ iterations. Altogether, the main cycle in the algorithm is repeated $n-r$ times.

See that the running time $O(n^3)$ of our algorithm refers to the number n of columns of $\boldsymbol{A}$, not to the size of input. The real input size of the matrix $\boldsymbol{A}$ is $O(n \cdot r)$ which is typically of order up to n^2. We do not attempt to determine how Algorithm 4.1.1 running time depends on the parameters $\mathbb{F}$ and t (which is at least an exponential function). However, we do care that the algorithm is recursive in $\mathbb{F}$ and t — there is one algorithm running for all values of the parameters, *not* a sequence of algorithms (cf. Section 1.3).

4.2 Adding an Element

Step (1) in Algorithm 4.1.1 is easy to implement. We are going to describe the implementation of step (2) of Algorithm 4.1.1. This is a surprisingly nontrivial task despite that we are adding the element f to the parse tree T arbitrarily. The main complication comes from necessity to recompute all boundary subspaces of the separations in T — a straightforward implementation of which would require us to solve systems of linear equations. All next algorithms are parametrized by a finite field $\mathbb{F}$ and integers t and $t' \leq 10t$, where $\mathbb{F}$, t are as in Algorithm 4.1.1.

Let $M = M(\boldsymbol{D})$ be a matroid represented by a matrix $\boldsymbol{D} \in \mathbb{F}^{r \times n}$, and consider a separation (E_1, E_2) in M (i.e. a partition of $E(M)$). The projective subspace $\langle E_1 \rangle \cap \langle E_2 \rangle$ spanned by both sides of the separation is called the *guts of the separation* (E_1, E_2). The rank of the guts equals $\lambda_M(E_1) - 1$ by modularity. Naturally, one may compute a spanning set of *generator vectors* (with respect to $\boldsymbol{D}$) for the guts of (E_1, E_2) from the vectors in $\boldsymbol{D}$. If the rank of the guts is less than t' which is a constant, then the combined size of its independent generator vectors is $O(r)$.

Algorithm 4.2.1. Adding a vector to a separation guts in a represented matroid.

Input: A matrix $\boldsymbol{D} = [\boldsymbol{I}_r \mid \boldsymbol{D}'] \in \mathbb{F}^{r\times n}$ representing the matroid $M = M(\boldsymbol{D})$; a separation (F_1, F_2) of the matroid $M \setminus f$ where f is the element represented by the last column of $\boldsymbol{D}$, and where $\lambda_{M\setminus f}(F_1) \leq t' + 1$; and independent generator vectors for the guts of the separation (F_1, F_2) with respect to $\boldsymbol{D}\setminus f$.

Output: Independent generator vectors for the guts of the separation $(F_1 \cup \{f\}, F_2)$ of M with respect to $\boldsymbol{D}$, computed in time $O(r)$.

Proof. Notice that we cannot even read the whole matrix $\boldsymbol{D}$ in time $O(r)$. This is, however, not a big problem since we are going to use only the unit vectors of $\boldsymbol{I}_r$ and the generator vectors of the guts in the algorithm. Let $\Psi = \langle F_1\rangle \cap \langle F_2\rangle$ be the given guts, and $\Psi' = \langle F_1 \cup \{f\}\rangle \cap \langle F_2\rangle$ be the guts we have to generate. Let $\boldsymbol{f}$ denote the last column of $\boldsymbol{D}$ representing the element f, and let $\boldsymbol{f}_1$ denote the vector obtained from $\boldsymbol{f}$ by setting to 0 the coordinates corresponding to the unit vectors of $\boldsymbol{I}_r$ appearing in F_1. Then $\boldsymbol{f}_1$ belongs to the span $\langle F_2\rangle$ by definition.

We may easily compute $\boldsymbol{f}_1$ in time $O(r)$. Moreover, we may decide whether $\boldsymbol{f}_1 \in \Psi$ in time $O(r)$ since there is a bounded number of generator vectors for Ψ. If $\boldsymbol{f}_1 \in \Psi$, then $\boldsymbol{f}$ belongs to the span $\langle F_1\rangle$ and so $\Psi' = \Psi$. Otherwise, since $\boldsymbol{f}_1$ belongs to the span $\langle F_1 \cup \{f\}\rangle$, we get $\Psi' = \langle \Psi \cup \boldsymbol{f}_1\rangle$. ∎

Let $M = M(\boldsymbol{D})$ be a matroid represented by a matrix $\boldsymbol{D} \in \mathbb{F}^{r\times n}$, and let T be a spanning $\leq t'$-boundaried parse tree for M. A *coordinatization* (with respect to $\boldsymbol{D}$) of the parse tree T is assignment of the appropriate vectors to the boundary points of the matroids $\bar{P}(T_1)$ for all rooted subtrees T_1 of T, as computed from the vectors of $\boldsymbol{D}$. Similarly as above, the combined size of the vectors for each boundary subspace is $O(r)$.

Algorithm 4.2.2. Computing the coordinatization of a spanning $\leq t'$-boundaried parse tree T for the represented matroid $M(\boldsymbol{D})$ over $\mathbb{F}$.

Input: A matrix $\boldsymbol{D} \in \mathbb{F}^{r\times n}$, and a spanning $\leq t'$-boundaried parse tree T for the represented matroid $M(\boldsymbol{D})$.

Output: The coordinatization of T with respect to $\boldsymbol{D}$, computed in time $O(nr)$.

Proof. If T_0 is a rooted subtree of T with one node, then the coordinatization of T_0 is trivial. So assume that the root r_0 of T_0 has the left and right rooted subtrees T_1 and T_2, and that r_0 is labelled by a composition operator $\odot$. (Hence $\bar{P}(T_0) = \bar{P}(T_1) \odot \bar{P}(T_2)$. One of the subtrees may be empty.) Since T is

a spanning parse tree, the boundary of $\bar{P}(T_0)$ is spanned by the boundaries of $\bar{P}(T_1)$ and $\bar{P}(T_2)$.

Thus the vectors assigned to the boundary points of $\bar{P}(T_0)$ with respect to $\boldsymbol{D}$ are linear combinations of the vectors of the boundary points of $\bar{P}(T_1)$ and $\bar{P}(T_2)$. There are finitely many $\leq t'$-boundaried composition operators for a fixed t', and so the scalars of these linear combinations can be precomputed for the parameter t' and each $\odot$. Then, at each node of T, we have to compute a bounded number of linear combinations from a bounded number of vectors of length r, which can be accomplished in time $O(r)$. There are $O(n)$ nodes in T. ∎

Consider three sequences $U_1, U_2, U_3 \subset \mathbb{F}^s$ of vectors where each U_i, $i = 1, 2, 3$ is formed by independent vectors. Then these sequences represent the composition operator $(U, \gamma_1, \gamma_2, \gamma_3)$; where $U = U_1 \cup U_2 \cup U_3$ is the point configuration, and γ_i maps the elements of the sequence in order $U_i = \{\gamma_i(1), \gamma_i(2), \dots, \gamma_i(t_i)\}$ for $i = 1, 2, 3$.

Algorithm 4.2.3. Computing the composition operator from given vectors.

Input: Three independent sequences $U_1, U_2, U_3 \subset \mathbb{F}^s$ of vectors where each sequence has at most t' members.

Output: The $\leq t'$-boundaried composition operator represented by the vectors of U_1, U_2, U_3, computed in time $O(s)$.

Proof. There is a bounded number of $\leq t'$-boundaried composition operators for each t'; and since each one is nothing else than a labelled point configuration, it may be identified by a bounded number of homogeneous linear vector equations and inequations. (The equations are invariant under nonsingular linear transformations.) Hence we find the composition operator represented by (U_1, U_2, U_3) in parametrized time $O(s)$ even by brute force. ∎

Lemma 4.2.4. *Step (2) in Algorithm 4.1.1 is implemented in time $O(n \cdot r)$ for fixed parameters t' and $\mathbb{F}$.*

Proof. Let $N = M \restriction X$ in the step (2) of Algorithm 4.1.1 (before adding f to X). We first compute the coordinatization of the parse tree T for N using Algorithm 4.2.2. Let T_1 be a rooted subtree of T, and let $F_1 = J(\bar{P}(T_1))$, $F_2 = E(N) - F_1$. Then, since the parse tree T is spanning, the boundary subspace of $\bar{P}(T_1)$ is the guts of the separation (F_1, F_2). We denote by T' the rooted subbinary tree obtained from T by adding arbitrarily a new leaf ℓ representing the element f. To turn T' into a parse tree for $N' = M \restriction X \cup \{f\}$, we have to re-compute all composition operators in T'.

Let T_1' be the rooted subtree of T' corresponding to T_1 above, and let $F_1' = J(\bar{P}(T_1'))$, $F_2' = E(N') - F_1'$. Then $F_1' = F_1 \cup \{f\}$ and $F_2' = F_2$ for $\ell \in V(T_1')$, or $F_2' = F_2 \cup \{f\}$ and $F_1' = F_1$ otherwise. We denote by $\boldsymbol{D} = [\boldsymbol{I}_r \mid \boldsymbol{D}']$ the submatrix of the matrix $\boldsymbol{A}$ from Algorithm 4.1.1 representing the matroid N'. We apply Algorithm 4.2.1 to $\boldsymbol{D}$, the separation (F_1, F_2) and f, and to the generator vectors of the boundary points of $\bar{P}(T_1)$ computed above, obtaining generator vectors of the boundary of $\bar{P}(T_1')$. We repeat the same procedure for all $O(n)$ rooted subtrees of T'. Finally, we use Algorithm 4.2.3 to determine the new composition operators to label all internal nodes of T'. ∎

4.3 Getting a better Tree

The heart of implementation of the step (3) in Algorithm 4.1.1 is the next claim. (We remark that the proof of this claim implicitly uses a so called "tangle" [41] – a notion dual to a branch-decomposition.)

Lemma 4.3.1. *Let $\bar{N}$ be a spanning $3t$-boundaried represented matroid such that the branch-width of the matroid $\bar{N} \oplus \bar{\Omega}$ is at most $t+1$. Then there are two $\leq 3t$-boundaried matroids $\bar{N}_1, \bar{N}_2$ such that $\bar{N} = \bar{N}_1 \odot \bar{N}_2$ for some $\leq 3t$-boundaried composition operator $\odot$.*

Proof. Recall that the represented matroid $N' = \bar{N} \oplus \bar{\Omega}$ is the restriction of $\bar{N}$ to the internal elements $J(\bar{N}) = E(N')$. We define a function $g : 2^{E(N')} \to \mathbb{Z}$, for a subset $F \subseteq E(N')$, as $g(F) = \mathrm{r}\left(\partial(\bar{N}) \cap \langle F \rangle\right)$, i.e. as the projective rank of the intersection of the span of F with the boundary of $\bar{N}$. Clearly, $g(E(N')) = 3t$ since $\bar{N}$ is spanning, and $g(F \cup \{f\}) \leq g(F) + 1$ for any F and f.

Let (U, τ) be a width-$(t+1)$ branch decomposition of N'. For an edge $e \in E(U)$, we define $F_i(e) = \tau^{-1}(V(U_i))$ for $i = 1, 2$ where U_1 and U_2 are the connected components of $U - e$. It is an easy exercise to show that there is an edge $e \in E(U)$ such that both $g(F_1(e)), g(F_2(e)) \geq t$. Moreover, the projective rank of $\langle F_1(e) \rangle \cap \langle F_2(e) \rangle$ is at most t since (U, τ) is a width-$(t+1)$ branch decomposition. Alogether for $i = 1, 2$, the projective rank of $\langle F_i(e) \rangle \cap \langle F_{3-i}(e) \cup \partial(\bar{N}) \rangle$ is at most $3t$. Hence the existence of $\bar{N}_1, \bar{N}_2$ and $\odot$ is established by the partition $(F_1(e), F_2(e))$. ∎

The task now is to find the partition (F_1, F_2) of $J(\bar{N}')$ inducing the boundaried matroids $\bar{N}_1, \bar{N}_2$, and the composition operator $\odot$ as in Lemma 4.3.1, efficiently. We use the $\leq t'$-boundaried parse tree T previously constructed in Algorithm 4.1.1. Unlike in the implementation of step (2), we do not work with the vectors representing $M(\boldsymbol{A})$ — instead, we obtain all necessary information from the parse tree T. This results in a more complicated but faster implementation. Again, all algorithms in this section are parametrized by a finite field $\mathbb{F}$ and integers t and $t' \leq 10t$. We first present the following two simple algorithms.

Algorithm 4.3.2. Computing the connectivity function of a separation.

Input: An $\leq t'$-boundaried parse tree T parsing the matroid $N = P(T)$, and a partition (F_1, F_2) of $E(N)$.

Output: The connectivity value $\lambda_N(F_1) - 1$ of the separation (F_1, F_2), computed in time $O(|V(T)|)$.

Proof. In other words, we are computing the projective rank of the guts of separation (F_1, F_2) in N. This is a straightforward application of dynamic programming over T. Let x be a vertex of T, and let T_x be the rooted subtree of T with the root x. Denote by $\bar{M}_x = \bar{P}(T_x)$, and by T_x', T_x'' the left and right rooted subtrees of x in T.

For $\bar{M}_x$ as above and for $E_i = F_i \cap J(\bar{M}_x)$, we call *boundary data* the triple (Σ_1, Σ_2, g) where $\Sigma_i = \partial(\bar{M}_x) \cap \langle E_i \rangle$ for $i = 1, 2$ is the intersection of the span $\langle E_i \rangle$ with the boundary of $\bar{M}_x$, and where g is the projective rank of the guts $\langle E_1 \rangle \cap \langle E_2 \rangle$ of the subseparation (E_1, E_2). Clearly, one may compute boundary data of $\bar{M}_x = \bar{P}(T_x)$ from boundary data of $\bar{P}(T_x')$ and $\bar{P}(T_x'')$, and from the composition operator labelling x in (parametrized) constant time. Hence the whole algorithm is implemented in linear time. ∎

Algorithm 4.3.3. Computing a good partition (F_1, F_2) for Lemma 4.3.1.

Input: A $\leq t'$-boundaried parse tree T parsing the represented matroid $N = P(T)$, and a subset $F_0 \subseteq E(N)$ such that $\lambda_N(F_0) = 3t$.

Output: A partition (F_1, F_2) of the set $E(N) - F_0$ such that $\lambda_N(F_1), \lambda_N(F_2) \leq 3t$, or an answer *NO* if no such partition exists; computed in time $O(|V(T)|)$.

Proof. This is a straightforward extension of the dynamic program implemented in Algorithm 4.3.2. Moreover, for each instance of boundary data in this case, we record one representative of the partition which we compute. Since the ground set of N can be easily implemented so that the operation of set union (of the representatives) takes constant time, the overall computing time is still linear in T. We leave details to the reader. ∎

Now we get to the hard part — computing a valid composition operator for the new node of the parse tree T' in step (3b) or (3c). We (implicitly) know the three boundaries of the composition operator from a defining tri-partition of the matroid elements. We, however, have to choose independent spanning sets of generator points for the boundaries in such a way that the two matching boundaries of adjacent composition operators in the parse tree T' get the same generator points. For a parse tree U, we call a *virtual point* of U any point which is contained in the span of some composition operator in the tree U. The concept of virtual points of the parse tree U allows us to determine relative positions of certain points with respect to the elements of the matroid $P(U)$, without necessity to use absolute vectors in a particular matroid representation.

If (F_1, F_2) is a separation in the represented matroid $P(U)$ parsed by U, then we are going to express spanning generator points for the guts of (F_1, F_2) as a sequence of virtual points of the parse tree U. We naturally say that a set Z of virtual points in U is independent if Z is linearly independent in the point configuration parsed by U. Next is the corresponding extension of Algorithm 4.3.2: The fact that the guts is always spanned by some virtual points of U is implicitly proved in the algorithm.

Algorithm 4.3.4. Computing virtual points spanning the guts of a given separation over a parse tree.

Input: An $\leq t'$-boundaried parse tree T parsing the matroid $N = P(T)$, and a partition (F_1, F_2) of $E(N)$ such that $\lambda_N(F_1) \leq 3t$. (The partition is symmetric, i.e. (F_1, F_2) is considered the same as (F_2, F_1).)

Output: A uniquely determined sequence $\{x_1, x_2, \ldots, x_k\}$, $k = \lambda_N(F_1) - 1$, of independent virtual points of the parse tree T which spans the guts of the separation (F_1, F_2). This is computed in time $O(|V(T)|)$.

Proof. Let x be a vertex of T labelled by $\odot^x$, and let T_x be the rooted subtree of T with the root x. Denote by $\bar{M}_x = \bar{P}(T_x)$, and by T'_x, T''_x the left and right rooted subtrees of x in T. Analogously to Algorithm 4.3.2, we call *boundary data* of $\bar{M}_x$ the quadruple $(\Sigma_1, \Sigma_2, g, Z)$ where $E_i = F_i \cap J(\bar{M}_x)$ and $\Sigma_i = \partial(\bar{M}_x) \cap \langle E_i \rangle$ for $i = 1, 2$, where $g = \mathrm{r}(\langle E_1 \rangle \cap \langle E_2 \rangle)$, and where $Z = \{z_1, \ldots, z_g\}$ is a sequence of independent virtual points of T which spans the guts of (E_1, E_2) in $\bar{M}_x$. Moreover, we require that the points of $Z \cap \partial(\bar{M}_x)$ contained in the boundary span the set $\Sigma_1 \cap \Sigma_2$ (boundary-span condition).

We denote by $\bar{M}'_x = \bar{P}(T'_x)$, $\bar{M}''_x = \bar{P}(T''_x)$, and by $E'_i = F_i \cap J(\bar{M}'_x)$, $E''_i = F_i \cap J(\bar{M}''_x)$. The guts $\langle E_1 \rangle \cap \langle E_2 \rangle$ is clearly spanned by the four sets $\langle E'_1 \rangle \cap \langle E'_2 \rangle$, $\langle E''_1 \rangle \cap \langle E''_2 \rangle$, $\langle E''_1 \rangle \cap \langle E'_2 \rangle$, $\langle E'_1 \rangle \cap \langle E''_2 \rangle$. The latter two sets $\langle E''_1 \rangle \cap \langle E'_2 \rangle$ and $\langle E'_1 \rangle \cap \langle E''_2 \rangle$ are contained in the span of the composition operator $\odot^x$ labelling x in the parse tree T by definition. Hence, using the boundary-span condition above, we can compute boundary data of $\bar{M}_x$ from boundary data of $\bar{M}'_x$ and of $\bar{M}''_x$, and from the composition operator $\odot^x$ in a canonical way.

Since the guts of (E_1, E_2) has bounded rank $g \leq 3t$ in $\bar{M}_x$, boundary data carry only limited amount of information. One node x of the parse tree T is processed in time depending on $\mathbb{F}$ and t, but not depending on the size of T. So the whole algorithm is implemented in (parametrized) linear time. ∎

Lastly, we present an analogue of Algorithm 4.2.3 computing the composition operator from sequences of virtual points. Consider three sequences Z_1, Z_2, Z_3 of virtual points in a common parse tree T, each Z_i, $i = 1, 2, 3$ formed by g_i independent points. Then these sequences represent the composition operator $(Z, \gamma_1, \gamma_2, \gamma_3)$; where $Z = Z_1 \cup Z_2 \cup Z_3$ is the ground point configuration as parsed by T, and γ_i maps the elements of the sequence in order $Z_i = \{\gamma_i(1), \gamma_i(2), \ldots, \gamma_i(g_i)\}$ for $i = 1, 2, 3$.

Algorithm 4.3.5. Computing the composition operator from the given virtual points in a parse tree.

Input: Three independent sequences Z_1, Z_2, Z_3 of virtual points in a common $\leq t'$-boundaried parse tree T, where each sequence has at most $3t$ members.

Output: The $\leq 3t$-boundaried composition operator represented by the virtual points of Z_1, Z_2, Z_3, computed in time $O(|V(T)|)$.

Proof. We argue similarly as in Algorithm 4.2.3. There is a bounded number of $\leq t'$-boundaried composition operators, and each one of them may be identified by a bounded number of linear vector equations and inequations. Moreover, these equations are homogeneous and so invariant under nonsingular vector transformations by the definition of a composition operator. Thus we can decide their validity for the input from information given in the parse tree T.

Let x be a vertex of T labelled by $\odot^x$, and let T_x be the rooted subtree of T with the root x. Denote by T'_x, T''_x the left and right rooted subtrees of x in T, by $\bar{M}_x = \bar{P}(T_x)$, and by $Z_x \subseteq Z_1 \cup Z_2 \cup Z_3$ those of given virtual points of T which are contained in the composition operators in the subtree T_x. We use the following dynamic program:

We call *boundary-combination data* of $\bar{M}_x$ the list of all linear combinations of the virtual points Z_x which result in a point in the boundary $\partial(\bar{M}_x)$. This is a well defined notion according to the definition of a parse tree and to elementary linear algebra; and one can determine boundary-combination data of $\bar{M}_x = \bar{P}(T_x)$ from boundary-combination data of $\bar{P}(T'_x)$ and of $\bar{P}(T''_x)$, and from the composition operator $\odot^x$ labelling x. Since boundary-combination data carry limited amount of information at each tree node, the whole parse tree T can be processed in linear parametrized time.

Consider a homogeneous linear (in)equation EQ over the points $Z_1 \cup Z_2 \cup Z_3$. Then, obviously, we can decide validity of EQ from boundary-combination data at the node y of T in which the last point involved in EQ gets encountered in the set Z_y. This takes constant time. Therefore, for any $\leq 3t$-boundaried composition operator $\odot$, we can decide whether the virtual points of Z_1, Z_2, Z_3 represent $\odot$. In this way we find out the $\leq 3t$-boundaried composition operator represented by Z_1, Z_2, Z_3 in total linear (parametrized) time $O(|V(T)|)$. ∎

Lemma 4.3.6. *Step (3) in Algorithm 4.1.1 is correctly implemented in time $O(n^2)$ for fixed parameters t, $t' \leq 10t$, and $\mathbb{F}$.*

Proof. Let $M = M(\boldsymbol{A})$ be the given matroid on n elements, and let the set X and the $\leq t'$-boundaried parse tree T be as in Algorithm 4.1.1. (So $P(T) = M \restriction X$.) Notice that the size of the partial parse trees considered in the algorithm is at most linear in $|X| \leq n$. The important point of the implementation of step (3) in Algorithm 4.1.1 is that the boundaried matroids labelling the partial spanning tree T' are not explicitly described: Instead of a boundaried matroid $\bar{N}$, we record only the set $J(\bar{N})$ of its internal elements. The boundary subspace of $\bar{N}$ is then implicitly given by the guts of the separation $(J(\bar{N}), X - J(\bar{N}))$, and the boundary points are handled as virtual points in the parse tree T.

Part (3a) of Algorithm 4.1.1 is trivial to implement. Let us look at part (3b): The boundary rank of the boundaried matroid $\bar{N}$ labelling a chosen leaf ℓ of T' is computed in time $O(n)$ by Algorithm 4.3.2. Suppose that the computed rank is less than $3t$. Let $e \in J(\bar{N})$ be an arbitrary element. Then the new composition operator being added to the parse tree T' corresponds to a tri-partition $X_1 = J(\bar{N}) - \{e\}$, $X_2 = \{e\}$, and $X_3 = X - J(\bar{N})$. Rest is described below.

Suppose that the above computed boundary rank of $\bar{N}$ equals $3t$. Then we are in part (3c): We call Algorithm 4.3.3 for $F_0 = X_3 = X - J(\bar{N})$ over the parse tree T. The result is a partition (X_1, X_2) of the set $X - X_3 = J(\bar{N})$, which does exist by Lemma 4.3.1 if branch-width of M is at most $t+1$. The whole tri-partition of X to X_1, X_2, X_3 then determines the new composition operator and the new leaf labels which will be added to T'. On the other hand, if branch-width of M exceeds $t+1$, then this part may possibly fail, and then Algorithm 4.1.1 ends with no parse tree and an error message. Computing time is $O(n)$ here.

So far, we have constructed a tri-partition (X_1, X_2, X_3) of the set X such that $\lambda_{M \restriction X}(X_i) \leq 3t$ for $i = 1, 2, 3$, and we are going to find the composition operator that would "glue" these three parts together in an enlarged parse tree T_1': We call Algorithm 4.3.4 for the separation $(X_i, X - X_i)$ over the parse tree T, for each $i = 1, 2, 3$. Then we call Algorithm 4.3.5 for the resulting three sequences Z_i, $i = 1, 2, 3$ of virtual points over the parse tree T to construct a composition operator $\odot'$. We construct the new tree T_1' from T' by adding two new leaves ℓ_1, ℓ_2 as sons of ℓ. We label ℓ_i, $i = 1, 2$ by the boundaried matroid $\bar{N}_i$ induced on the elements X_i and the boundary Z_i, and we re-label ℓ with $\odot'$. (Rest of T_1' is unchanged.) This is all done in time $O(n)$ again.

It remains to prove correctness of the above construction by induction on $|V(T')|$. At the beginning, when T' has one vertex, the matroid $\bar{N}$ has boundary rank 0, and $X_3 = \emptyset$ so Z_3 is an empty sequence. Otherwise, the sequence Z_3 coincides with the set of boundary points of $\bar{N}$ since they were both determined uniquely by Algorithm 4.3.4 for the same separation $(X_3, J(\bar{N}))$. Therefore, $\bar{N}_1 \odot' \bar{N}_2 = \bar{N}$, and so $\bar{P}(T_1') = \bar{P}(T')$ and $P(T_1') = P(T) = M \restriction X$.

Since each iteration of step (3) adds a new leaf to the constructed parse tree T', there are at most $|X| - 1 = O(n)$ iterations. Thus step (3) is finished in time $O(n^2)$. We remark that our implementation works for an arbitrary finite field $\mathbb{F}$ and an integer $t \geq 1$ given as parameters. ∎

4.4 Conclusion of Algorithm 4.1.1

Using Lemmas 4.2.4 and 4.3.6, and the preceding description, we immediately conclude:

Theorem 4.4.1. *Let us fix an integer $t \geq 1$ and a finite field $\mathbb{F}$, and consider a given n-element $\mathbb{F}$-represented matroid $M = M(\boldsymbol{A})$.*

(a) If branch-width of M is at most $t+1$, then Algorithm 4.1.1 computes a spanning $\leq 3t$-boundaried parse tree T for M in time $O(n^3)$.

(b) If branch-width of M exceeds $t+1$, then Algorithm 4.1.1 (possibly) ends with no output parse tree, but the computation is also finished in time $O(n^3)$. ∎

Unfortunately, Algorithm 4.1.1 does not necessarily produce the optimal parse tree for a given matroid. It is still possible to determine the exact branch-width of a represented matroid by a parametrized polynomial algorithm, using the following facts: The class $\mathcal{B}_k$ of matroids of branch-width at most k is closed under taking minors. By the result of [18], the excluded minors for the class $\mathcal{B}_k$ have size at most $(6^{k+1} - 1)/5$, and hence their number is finite and they can all be found by a brute force algorithmic search. We remark that, unlike for graph minors, it is not known how to test for a fixed matroid minor in polynomial time. However, we can use the bounded-width decomposition obtained by Algorithm 4.1.1 to test for a minor in parametrized polynomial time.

For the next arguments, we need the tools of monadic second-order logic (shortly MSOL) of matroids. If you are not familiar with this area, we suggest to read first Chapter 6, and then continue with the rest of this section. Briefly speaking, we prove here that the property of matroid having a fixed minor can be expressed using a special kind of logic formula; and each such property then can be recognized by a tree automaton when branch-width is bounded. Thus we may combine Algorithm 4.1.1 with those tree automata to determine branch-width of a given matroid exactly.

Lemma 4.4.2. *For every matroid N there is an MSOL formula ψ_N such that $\psi_N(M)$ is true for a matroid M if and only if N is a minor of M.*

Proof. It is well known that a matroid is fully described by the set of its circuits. By definition, N is a minor of M if and only if there are two sets C, D such that $N = M/C \setminus D$. Suppose that $N = M/C \setminus D$ holds. Clearly, a set $X \subseteq E(N)$ is a circuit in N if and only if there is a circuit $Y \subseteq E(M)$ such that $Y - X \subseteq C$.

We identify the elements of the minor N in M by variables $x_1, \ldots, x_n$ where $n = |E(N)|$. For $J \subseteq [1, n]$, we then write

$$\mathrm{circ}(x_j : j \in J; C) \equiv \exists Y, Z\, [\neg\, \mathrm{indep}(Y) \wedge$$

$$\wedge \forall y (y \notin Y \vee \mathrm{indep}(Y - \{y\})) \wedge Z = (Y - \{x_j : j \in J\}) \wedge Z \subseteq C].$$

Now we can express

$$\psi_N \equiv \exists C\, \exists x_1, \ldots, x_n\, [$$

$$\bigwedge\nolimits_{J \in \mathcal{J}_+} \mathrm{circ}(x_j : j \in J; C) \wedge \bigwedge\nolimits_{J \in \mathcal{J}_-} \neg\, \mathrm{circ}(x_j : j \in J; C)],$$

where $\mathcal{J}_+$ is the set of all $J \subseteq [1, n]$ such that $\{x_j : j \in J\}$ actually is a circuit in N, and where $\mathcal{J}_-$ is the complement of $\mathcal{J}_+$. The (bounded) set subtraction used above as a shortcut has an obvious MSOL implementation. ∎

Corollary 4.4.3. *For every $k \geq 1$, there is an MSOL formula ϕ_k such that $\phi_k(M)$ is true for a matroid M if and only if M has branch-width at most k (i.e. $\phi_k(M) \equiv M \in \mathcal{B}_k$).*

∎

Theorem 4.4.4. *Let $\mathbb{F}$ be a finite field, and let $t \geq 1$ be a constant. There is an algorithm that, given a rank-r matrix $\boldsymbol{A} \in \mathbb{F}^{r \times n}$ such that the branch-width of the matroid $M(\boldsymbol{A})$ is at most $t+1$, finds exact branch-width of $M(\boldsymbol{A})$ in time $O(n^3)$.*

Proof. For $k = 2, 3, \ldots, t+1$, the algorithm first pre-computes all the excluded minors for the class $\mathcal{B}_k$, and the formulas ϕ_k from Corollary 4.4.3. Then the algorithm computes the finite tree automaton $\mathcal{A}_k$ from Theorem 6.4.1, which accepts the parse trees for represented matroids described by ϕ_k. (This pre-computation is done in time not depending on M.)

Given an n-element matroid $M(\boldsymbol{A})$ represented over $\mathbb{F}$, the algorithm calls Algorithm 4.1.1 to produce an $\leq 3t$-boundaried parse tree T of $M(\boldsymbol{A})$ in time $O(n^3)$. Then it finds the smallest $k_0 \leq t+1$ such that the parse tree T is accepted by $\mathcal{A}_{k_0}$, where each automaton $\mathcal{A}_k$ is emulated in time $O(n)$. The branch-width of $M(\boldsymbol{A})$ is k_0. ∎

Corollary 4.4.5. *For a finite field $\mathbb{F}$, branch-width of an $\mathbb{F}$-represented matroid is a uniformly fixed-parameter tractable problem.*

∎

Later we will see a radical structural change when moving from represented matroids over finite fields to general abstract matroids, or even to matroids over infinite fields. The situation seems to be slightly different for the problem of branch-width itself, at least when branch-width is 3. We present in Section 8.3 an easy algorithm that decides whether a matroid has branch-width at most 3 in polynomial time. (The algorithm also has a fast practical implementation.) This algorithm is not restricted to represented matroids — it works for all matroids for which the rank function can be efficiently determined. Unfortunately, there seems to be no way how to extend the algorithm to higher values of branch-width. So we ask:

Problem 4.4.6. What is the parametrized complexity of the problem to determine branch-width for a matroid M

(a) if $M = M(\boldsymbol{A})$ is given by a matrix representation over an infinite field;
(b) if M is given by a rank oracle?

Chapter 5

Computing the Tutte Polynomial

The Tutte polynomial is an amazing algebraic parameter of a graph or of a matroid, capturing lots of interesting information. Is is not surprising that computing the Tutte polynomial is a computationally hard task. So far, researchers focused on computing this polynomial for special graph or matroid families. For example, it is known that the Tutte polynomial can be computed efficiently for graphs of bounded tree-width [2, 37]. We contribute a similar results for matroids of bounded branch-width which are represented over finite fields.

We remark that our approach is based on similar ideas to those of Noble [37], but extended to represented matroids. By using a powerful, although not simple, machinery of boundaried parse trees, we are able to provide a more straightforward recursive procedure for computing the Tutte polynomial. That is why we think that our result is interesting and useful for computing the Tutte polynomial also for graphs of bounded branch-width.

The content of this chapter is taken from our original paper [26].

5.1 Definition of the Tutte Polynomial

The Tutte polynomial of a matroid naturally extends the original Tutte's definition of this polynomial for graphs [49]. The *Tutte polynomial* of a matroid M on the ground set E is given by

$$T(M;\, x, y) = \sum_{A\subseteq E} (x-1)^{r_M(E)-r_M(A)}(y-1)^{|A|-r_M(A)} \,.$$

Notice that $T(M;\, x, y) = T(M^*;\, y, x)$ is self-dual. The Tutte polynomial of a graph G is the polynomial $T(G;\, x, y) = T(M(G);\, x, y)$.

Specific evaluations of the Tutte polynomial include such interesting and difficult quantities as the chromatic and flow polynomials of a graph, the critical number of a matroid, the weight enumerator of a linear code, the all-terminal reliability of a network, the partition function of the Ising model in physics, or the Jones and Kaufman bracket polynomials of an alternating knot. See, for example, [52, 53]. It is not surprising that computing or evaluating the Tutte polynomial is $\#P$-hard except for a few special cases, which was proved by Jaeger, Vertigan and Welsh [33]:

Theorem 5.1.1. (Jaeger, Vertigan and Welsh, 1990) *Let $M = M(G)$ be the cycle matroid of a graph G. Evaluating the Tutte polynomial $T(M;\, x, y)$ at $(x, y) = (a, b)$ is $\#P$-hard unless $(a-1)(b-1) = 1$ or $(a, b) \in \{(1, 1), (-1, -1), (0, -1), (-1, 0), (i, -i), (-i, i), (j, j^2), (j^2, j)\}$ where $i^2 = -1$ and $j = e^{2\pi i/3}$.*

It is even $\#P$-hard to evaluate the Tutte polynomial at all but few special points for planar graphs [51]. In view of of those hardness results, researchers have focused on computing the Tutte polynomial for special graph classes. Most noticeable to us are algorithms computing the Tutte polynomial for graphs of bounded tree-width: Andrzejak [2] gives a polynomial-time algorithm for computation of the Tutte polynomial for such graphs, unfortunately with an exponent heavily depending on the tree-width. Independently, Noble [37] provides a low-exponent polynomial-time algorithm for evaluation of the Tutte polynomial at a rational point. (We have noticed that some papers refer to Noble's result as to evaluation of the Tutte polynomial in "linear time", but this is not true – the polynomial is evaluated using a linear number of arithmetic operations, and the cost of one operation depends significantly on the size of the graph, as it is precisely stated in [37, Theorem 1.1].) Another, "logic-type" approach to the same problem, using generating functions definable in the second order monadic logic, is presented by Makowsky in [34]. However, that approach provides no reasonable explicit algorithm.

The aim of our work is to extend these efficient computations of the Tutte polynomial from graphs to matroids represented over finite fields. Our approach is based on similar ideas to those of Noble [37], but extended to represented matroids. By using a powerful, although not simple, machinery of boundaried parse trees, we are able to provide a more straightforward recursive procedure for computing the Tutte polynomial. That is why we think that our result is interesting and useful for computing the Tutte polynomial also for graphs of bounded branch-width. We remark that it is likely not possible to extend our work to matroids represented over infinite fields, as Theorem 7.4.1 shows.

To handle the Tutte polynomial for boundaried represented matroids, we need a formal way to distinguish or to "mark" a subspace of the matroid boundary. Let $\bar\Omega_t$ be the empty t-boundaried matroid as above, and let $\Phi \subseteq \partial(\bar\Omega_t)$ be a subspace of the boundary space of $\bar\Omega_t$. Then the pair $\mathrm{K} = [\bar\Omega_t, \Phi]$ is called the *t-boundary mark*. (We extend projective transformations of point configurations to subspaces in the obvious sense.) The set of all t-boundary marks is denoted by $\mathcal{K}_t$, and the cardinality $|\mathcal{K}_t|$ obviously equals the number of subspaces of the rank-t projective geometry over $\mathbb{F}$.

Let $\bar M = (M, \delta)$ be a t-boundaried represented matroid, and let $J = J(\bar M)$ be the internal elements of $\bar M$. We denote by $Z_t = (z_{\mathrm{K}} : \mathrm{K} \in \mathcal{K}_t)$ a vector of $|\mathcal{K}_t|$ free variables. Moreover, for $A \subseteq J$, we denote by $\mathrm{K}\left(\bar{M}\,|A\right) \in \mathcal{K}_t$ the t-boundary mark formed by the boundary points of $\bar{M}$, and by the subspace $\partial(\bar{M}) \cap \langle A\rangle$ which is the intersection of the boundary of $\bar{M}$ with the span of A. The *boundaried Tutte polynomial* of a t-boundaried matroid $\bar{M}$ is given by

$$T_B\left(\bar{M};\, x, y, Z_t\right) = \sum_{A\subseteq J} (x-1)^{r_M(J)-r_M(A)} \cdot (y-1)^{|A|-r_M(A)} \cdot z_{\mathrm{K}\left(\bar{M}\,|A\right)}\,.$$

Proposition 5.1.2. $T\left(\bar{M} \oplus \bar{\Omega};\, x, y\right) = T_B\left(\bar{M};\, x, y, (1,\ldots,1)\right)$.

Proof. Since $\bar{M} \oplus \bar{\Omega} = M_1$ is a restriction of M to J, we have $J = E(M_1)$. Moreover, the rank function $r_{M_1} : 2^J \to \mathbb{N}$ is a restriction of r_M to the subsets of J. Hence the claim follows directly from the above definition. ∎

5.2 Recursive Formula on Parse Trees

We will combine the machinery of boundaried parse trees with the definition of the boundaried Tutte polynomial to produce a straightforward recursive formula for evaluating the Tutte polynomial over a given matroid parse tree. If M is a matroid (represented over $\mathbb{F}$) of branch-width at most $t+1$, then there is a $\leq t$-boundaried parse tree T_M by Theorem 3.3.2 such that the represented matroid $P(T_M)$ parsed by T_M is isomorphic to M. Thus we may use Proposition 5.1.2 to compute the Tutte polynomial for M from the boundaried one for $\bar{P}(T_M)$.

There is an obvious natural way to apply a composition operator to boundary marks. For that we need the following formalization: For a t-boundary mark $\mathrm{K} = [\bar{\Omega}_t, \Phi]$ where $\bar{\Omega}_t = (\Omega, \delta_0)$, we say that a t-boundaried matroid $\bar{L} = (L, \delta_0)$ is a *representative* for K if $E(L) \supseteq E(\Omega)$, and the span $\langle E(L) - E(\Omega)\rangle = \Phi$.

Let $\odot = (R, \gamma_1, \gamma_2, \gamma_3)$ be a $\leq t$-boundaried composition operator where $t_i = t_i(\odot)$, and let $\mathrm{K}_i \in \mathcal{K}_{t_i}$ for $i = 1, 2$ be a t_i-boundary mark with a representative $\bar{L}_i$. Then we denote by $[\mathrm{K}_1{\odot}\mathrm{K}_2]$ the t_3-boundary mark $\mathrm{K}\left(\bar{L}\,|J\right)$ in the matroid $\bar{L} = \bar{L}_1 \odot \bar{L}_2$ shown by the set $J = J(\bar{L}) = J(\bar{L}_1) \cup J(\bar{L}_2)$. Moreover, we denote by $\sigma(\odot) = t_1 + t_2 - r_R(\gamma_1([1,t_1]) \cup \gamma_2([1,t_2]))$ the projective rank of the intersection of boundaries $\partial(R,\gamma_1) \cap \partial(R,\gamma_2)$ in $\odot$, and by $\varrho(\odot;\, \mathrm{K}_1, \mathrm{K}_2) = r_L(J(\bar{L}_1)) + r_L(J(\bar{L}_2)) - r_L(J(\bar{L}))$ the rank of the intersection of the subspaces marked by K_1 and K_2 in $\odot$. It is easy to see that the outcome of these definitions is independent of a particular choice of the representatives $\bar{L}_1, \bar{L}_2$.

Our recursive computation of the boundaried Tutte polynomial over a boundaried spanning parse tree is based on the following statement. Here $P(x, y, z, \ldots) \restriction z$ stands for the coefficient of the variable z in a polynomial P.

Theorem 5.2.1. *Let $\odot$ be a $\leq t$-boundaried composition operator, and let $t_i = t_i(\odot)$ for $i = 1, 2, 3$. Suppose that $\bar{N}_i$, $i = 1, 2$ are t_i-boundaried spanning represented matroids. Then*

$$T_B\left(\bar{N}_1 \odot \bar{N}_2;\, x, y, Z_{t_3}\right) =$$
$$= \sum_{\mathrm{K}_1 \in \mathcal{K}_{t_1},\ \mathrm{K}_2 \in \mathcal{K}_{t_2}} \left(T_B\left(\bar{N}_1;\, x, y, Z_{t_1}\right) \restriction z_{\mathrm{K}_1}\right) \cdot \left(T_B\left(\bar{N}_2;\, x, y, Z_{t_2}\right) \restriction z_{\mathrm{K}_2}\right) \cdot$$
$$\cdot (x-1)^{\varrho(\odot;\, \mathrm{K}_1,\mathrm{K}_2)-\sigma(\odot)} \cdot (y-1)^{\varrho(\odot;\, \mathrm{K}_1,\mathrm{K}_2)} \cdot z_{[\mathrm{K}_1\odot\mathrm{K}_2]}\,.$$

Proof. Let $\odot = (R, \gamma_1, \gamma_2, \gamma_3)$, let $\bar{N}_i = (N_i, \delta_i)$ for $i = 1, 2$, and let $\bar{N} = (N, \delta) = \bar{N}_1 \odot \bar{N}_2$. We choose an arbitrary set $A \subseteq J = J(\bar{N})$, and we denote by $J_1 = J(\bar{N}_1)$, $J_2 = J(\bar{N}_2)$, and by $A_1 = A \cap J_1$, $A_2 = A \cap J_2$. (Notice that J_1, J_2 form a partition of J.) We denote by $\mathrm{K}_i = \mathrm{K}\left(\bar{N}_i\,|A_i\right)$ for $i = 1, 2$ the boundary mark shown by the set A_i in $\bar{N}_i$. Let $r(\cdot)$ be the rank function of the projective space.

By modularity of $r(\cdot)$ in the projective space, for the point configuration representing N we may write $r_N(J) = r_N(J_1) + r_N(J_2) - r(\langle J_1\rangle \cap \langle J_2\rangle)$. Since both N_1, N_2 are spanning, and $\langle J_1\rangle \cap \langle J_2\rangle \subseteq \langle E(R)\rangle$ in the composition $\odot$, we get $r(\langle J_1\rangle \cap \langle J_2\rangle) = r(\partial(R,\gamma_1) \cap \partial(R,\gamma_2)) = \sigma(\odot)$. Hence $r_N(J) = r_N(J_1) + r_N(J_2) - \sigma(\odot)$.

Considering similarly the points of A in N, we may write $r_N(A) = r_N(A_1) + r_N(A_2) - r(\langle A_1\rangle \cap \langle A_2\rangle)$. Let us choose a representative $\bar{L}_i$ of the boundary mark K_i, $i = 1, 2$, and denote by $\bar{L} = \bar{L}_1 \odot \bar{L}_2$. Then (considering first the point configuration of $\bar{N}$ and then the point configuration of $\bar{L}$) we have $r(\langle A_1\rangle \cap \langle A_2\rangle) = r(\langle J(L_1)\rangle \cap \langle J(L_2)\rangle) = r_L(J(L_1)) + r_L(J(L_2)) - r_L(J(L))$ by modularity of $r(\cdot)$. Hence $r_N(A) = r_N(A_1) + r_N(A_2) - \varrho(\odot;\, \mathrm{K}_1, \mathrm{K}_2)$.

Altogether, we conclude

$$(x-1)^{r_N(J)-r_N(A)} \cdot (y-1)^{|A|-r_N(A)} =$$
$$= (x-1)^{r_{N_1}(J_1)-r_{N_1}(A_1)+r_{N_2}(J_2)-r_{N_2}(A_2)} \cdot (y-1)^{|A_1|-r_{N_1}(A_1)+|A_2|-r_{N_2}(A_2)} \cdot$$
$$\cdot (x-1)^{\varrho(\odot;\, \mathrm{K}_1,\mathrm{K}_2)-\sigma(\odot)} \cdot (y-1)^{\varrho(\odot;\, \mathrm{K}_1,\mathrm{K}_2)}\,.$$

Moreover, the boundary mark of the set A in $\bar{N}$ is given by $[\mathrm{K}_1{\odot}\mathrm{K}_2]$. After multiplying this expression by $z_{[\mathrm{K}_1\odot\mathrm{K}_2]}$, summing over all choices of $A \subseteq J$, and grouping the terms by $\mathrm{K}_1, \mathrm{K}_2$ we get the required formula. ∎

In the next corollaries we provide a "computer science" view of the previous result. To distinguish the Z-variables of distinct Tutte polynomials, we introduce additional vectors of free variables Z'_{t_1} and Z''_{t_2}. Then, we may rewrite the formula of the previous theorem in the following way:

Corollary 5.2.2. *In the setting of Theorem 5.2.1,*

$$T_B\left(\bar{N}_1 \odot \bar{N}_2;\, x, y, Z_{t_3}\right) = T_B\left(\bar{N}_1;\, x, y, Z'_{t_1}\right) \cdot T_B\left(\bar{N}_2;\, x, y, Z''_{t_2}\right),$$

where

$$z'_{\mathrm{K}_1} \cdot z''_{\mathrm{K}_2} = (x-1)^{\varrho(\odot;\,\mathrm{K}_1,\mathrm{K}_2)-\sigma(\odot)} \cdot (y-1)^{\varrho(\odot;\,\mathrm{K}_1,\mathrm{K}_2)} \cdot z_{[\mathrm{K}_1 \odot \mathrm{K}_2]}$$

for each pair $\mathrm{K}_i \in \mathcal{K}_{t_i}$, $i = 1, 2$.

Proof. The key fact to notice here is that each term of the polynomial $T_B\left(\bar{N}_1;\, x, y, Z'_{t_1}\right)$ contains exactly one of the variables z'_{K_1} for $\mathrm{K}_1 \in \mathcal{K}_{t_1}$, and z'_{K_1} is in the first power. An analogous statement applies to $T_B\left(\bar{N}_2;\, x, y, Z''_{t_2}\right)$. Hence the above substitutions of $z'_{\mathrm{K}_1} \cdot z''_{\mathrm{K}_2}$ provide the same result as the formula in Theorem 5.2.1. ∎

Corollary 5.2.3. *Let $T \in \Pi_t^{**}$ be a $\leq t$-boundaried spanning parse tree. If T is an empty tree, then*

$$T_B\left(\bar{P}(T);\, x, y, Z_0\right) = T_B\left(\bar{\Omega}_0;\, x, y, Z_0\right) = z_{\mathrm{K}(\bar{\Omega}_0\,|\,\emptyset)}\,.$$

If T has exactly one vertex labelled by $\bar{\Upsilon}$ or $\bar{\Upsilon}_0$, then

$$T_B\left(\bar{P}(T);\, x, y, Z_1\right) = T_B\left(\bar{\Upsilon};\, x, y, Z_1\right) = (x-1) z_{\mathrm{K}(\bar{\Upsilon}\,|\,\emptyset)} + z_{\mathrm{K}(\bar{\Upsilon}\,|\,J(\bar{\Upsilon}))}\,, \quad or$$

$$T_B\left(\bar{P}(T);\, x, y, Z_0\right) = T_B\left(\bar{\Upsilon}_0;\, x, y, Z_0\right) = z_{\mathrm{K}(\bar{\Upsilon}_0\,|\,\emptyset)} + (y-1) z_{\mathrm{K}(\bar{\Upsilon}_0\,|\,J(\bar{\Upsilon}_0))}\,.$$

If r is the root of T labelled by $\odot$, and T_1, T_2 are the sons of r in T, then

$$T_B\left(\bar{P}(T);\, x, y, Z_{t_3}\right) = T_B\left(\bar{P}(T_1);\, x, y, Z'_{t_1}\right) \cdot T_B\left(\bar{P}(T_2);\, x, y, Z''_{t_2}\right),$$

where

$$z'_{\mathrm{K}_1} \cdot z''_{\mathrm{K}_2} = (x-1)^{\varrho(\odot;\,\mathrm{K}_1,\mathrm{K}_2)-\sigma(\odot)} \cdot (y-1)^{\varrho(\odot;\,\mathrm{K}_1,\mathrm{K}_2)} \cdot z_{[\mathrm{K}_1 \odot \mathrm{K}_2]}$$

for each pair $\mathrm{K}_i \in \mathcal{K}_{t_i(\odot)}$, $i = 1, 2$.

Proof. This corollary follows directly from the definitions of the boundaried parse tree and of the boundaried Tutte polynomial, and from Corollary 5.2.2. ∎

The last corollary provides a straightforward recursive procedure for computing the boundaried Tutte polynomial on a boundaried parse tree T in $|V(T)|$ steps; each step involving one multiplication of polynomials, and a bounded number of variable substitutions.

Remark. One may consider "element-weighted" extensions of the Tutte polynomial, defined by various researchers. It is easy to see that our Theorem 5.2.1 can be simply extended to handle also those polynomials. However, since such an extension does not add anything substantially new to our result, we are making no formal statements about that issue here.

5.3 Complexity Review

We now review what is the running time of an algorithm based on Corollary 5.2.3 and Theorem 4.4.1. One important remark we want to make here is about considering the length of arithmetic operations in combinatorial algorithms: Usually, combinatorial algorithms work with integer numbers of length proportional to $\log n$ where n is the length of the input. It is common (and well justified by practical applications) to consider that one arithmetic operation in such a case takes constant time. However, when computing or evaluation the Tutte polynomial, we deal with numbers of length proportional to n, which is too much to neglect.

Theorem 5.3.1. *Let $u(n)$ be the number of elementary integer arithmetic operations (sum or product) needed to compute the product of two two-variable polynomials of degrees at most n. Let $v(n)$ be time needed to multiply (or sum) two n-bit integers. Assume that $\mathbb{F}$ is a finite field, t is an integer constant, and that $u(n) \geq \Theta(n^2)$. If M is an n-element rank-r $\mathbb{F}$-represented matroid of branch-width at most t given by a matrix $\mathbf{A} \in \mathbb{F}^{r\times n}$, then the Tutte polynomial $T(M;\, x, y)$ can be computed in time $O(n \cdot u(n) \cdot v(n))$.*

Proof. Firstly, we compute a $\leq t$-boundaried parse tree T of M in time $O(n^3) \leq \Theta(n \cdot u(n))$, using the algorithm of Theorem 4.4.1.

Then we follow the formulas in Corollary 5.2.3. Computing the Tutte polynomial for a parse tree with at most one vertex takes constant time. We turn our attention to the main formula $T_B\left(\bar{P}(T);\, x, y, Z_{t_3}\right) = T_B\left(\bar{P}(T_1);\, x, y, Z'_{t_1}\right) \cdot T_B\left(\bar{P}(T_2);\, x, y, Z''_{t_2}\right)$. Notice that the set $\mathcal{K}_{t_i}$ has bounded size for a constant $t_i \leq t$, and that the variables of Z'_{t_1}, Z''_{t_2} appear only in the first power in the above polynomials. We may compute the product of these polynomials as a bounded number of partial products, thus using only $O(u(n))$ elementary arithmetic operations. Then we apply a bounded number of variable substitutions with bounded-degree polynomials in time $O(n^2)$.

Altogether, we have to compute $O(n)$ such products of polynomials. Looking at the definition of the boundaried Tutte polynomial, we see that the coefficients of the involved polynomials are not larger that $2^{O(n)}$ since $|E(P(T))| = n$. Hence the coefficients have length of at most $O(n)$ bits, and so one elementary arithmetic operation with them can be done in time $O(v(n))$. Finally, we substitute $Z_t = (1, \ldots, 1)$ in time $O(n^2)$. By combining all these estimates, we get the required upper time-bound of $O(n \cdot u(n) \cdot v(n))$. ∎

Corollary 5.3.2. *Assume that $\mathbb{F}$ is a finite field, and that t is an integer constant. If M is an n-element rank-r $\mathbb{F}$-represented matroid of branch-width at most t given by a matrix $\mathbf{A} \in \mathbb{F}^{r\times n}$, then the Tutte polynomial $T(M;\, x, y)$ can be computed in time $O(n^6 \log n \log\log n)$.*

Proof. A two-variable polynomial of degree at most n has at most $(n+1)^2$ terms. Hence the trivial algorithm can multiply two such polynomials using $u(n) = O(n^4)$ elementary arithmetic operations with the coefficients. A well known algorithm by Strassen multiplies two n-bit numbers in time $v(n) = O(n \log n \log\log n)$. The rest follows from Theorem 5.3.1. ∎

We remark that the input size of the matroid M in our algorithm is $O(nr)$, which is typically of order n^2. It is shown in [37] that to compute the Tutte polynomial of a graph of bounded tree-width and size n takes time at least $\Omega(n^3)$. The performance of our algorithm does not match this lower bound, but it is still much better than the algorithm for the Tutte polynomial of a graph of tree-width t provided in [2] whose exponent depends on t. We also show that, with our more general matroidal algorithm, we can almost match the performance of [37] when evaluating the Tutte polynomial at a rational point.

Corollary 5.3.3. *Let $\mathbb{F}$ be a finite field, and let t be an integer constant. Suppose that a, b are rational numbers given as fractions of integers $a = \frac{p_a}{q_a}$, $b = \frac{p_b}{q_b}$ such that the combined length of p_a, q_a, p_b, q_b is l bits, and that M is an n-element rank-r $\mathbb{F}$-represented matroid of branch-width at most t given by a matrix $\boldsymbol{A} \in \mathbb{F}^{r\times n}$. Then the Tutte polynomial of M can be evaluated as $T(M;\, a, b)$ at a, b in time $O(\, n^3 + n^2 l \cdot \log(nl) \cdot \log\log(nl)\,)$.*

Proof. We slightly modify the procedure of Theorem 5.3.1. This time $T_B\left(\bar{P}(T);\, a, b, Z_t\right)$ is a polynomial of degree 1 with variables Z_t. Since Z_t contains a bounded number of variables, each of the polynomials in the product $T_B\left(\bar{P}(T_1);\, a, b, Z'_{t_1}\right) \cdot T_B\left(\bar{P}(T_2);\, a, b, Z''_{t_2}\right)$ has bounded number of terms, and hence we may compute the product in total time $O(v(k)) = O(k \log k \log\log k)$ where $k = O(nl)$ is the number of bits sufficient to describe the coefficients. (We do not have to consider division since we evaluate the result as a fraction of two integers.) The rest follows easily.

The reader may ask whether our procedure is valid when $a = 1$; an evaluation of $(x-1)^{\varrho(\odot;\, \mathrm{K}_1, \mathrm{K}_2) - \sigma(\odot)}$ at $x = a$ could mean division by zero when substituting the $z'_{\mathrm{K}_1} \cdot z''_{\mathrm{K}_2}$ variables. (See in Corollary 5.2.3.) Fortunately, this cannot happen as we show next. If $\varrho(\odot;\, \mathrm{K}_1, \mathrm{K}_2) \le \sigma(\odot) - 1$ for some $\mathrm{K}_1, \mathrm{K}_2$, then at least one of the sides in the product, say $T_B\left(\bar{N}_1;\, x, y, Z'_{t_1}\right) \restriction z_{\mathrm{K}_1}$, contains the factor $(x-1)$. Otherwise, the term $T_B\left(\bar{N};\, x, y, Z_t\right) \restriction z_{[\mathrm{K}_1 \odot \mathrm{K}_2]}$ of the Tutte polynomial for $\bar{N}$ would have the factor $(x-1)$ in a negative power by Theorem 5.2.1, which contradicts the definition of the polynomial. Hence the term $z'_{\mathrm{K}_1} \cdot z''_{\mathrm{K}_2}$ does not appear at all in our evaluation at $x = a = 1$. ∎

Noble [37] provides an algorithm evaluating the Tutte polynomial $T(G;\, a, b)$ at a, b for a graph G of bounded tree-width in time $O((v+p) \cdot el \cdot \log e \log\log e \cdot \log l \log\log l)$, where v is the number of vertices, e is the number of edges, and p the the size of the largest parallel class in G. Note that $n = e$ in our setting. The performance of our algorithm in Corollary 5.3.3 is quite close to Noble [37]. In fact, one can compute the parse tree for the cycle matroid of a graph of bounded branch-width in linear time using the graphic algorithm of Bodlaender and Thilikos [8], and so one can match the performance of [37] with our algorithm for cycle matroids of simple graphs.

Chapter 6

Properties Definable in the MSO Logic

We now get to our main result — showing that a large class of matroid problems expressed in certain language (so called MSO logic) can be solved quickly for represented matroids of bounded branch-width. For graphs, an analogous result is known from late eighties: The so called "MS_2-theorem" for graphs was proved by Courcelle, and similar statements were given independently by others.

Our matroid extension of that result is formulated using the notations of parse trees and tree automata. Our proof is inspired by Abrahamson-Fellows' [1] approach to the MS_2-theorem, adapted to matroids. The content of this chapter is taken from our original paper [27].

6.1 Analogue of the Myhill-Nerode Theorem

This section uses standard automata-theoretical arguments, following [16, Section 6.4]. We start with the classical Myhill-Nerode theorem for tree automata. We denote by $\Sigma^{**}_{(x)}$ the class of all rooted ordered sub-binary trees labelled by $\Sigma \cup \{x\}$ such that exactly one vertex which is a leaf is labelled by $x \notin \Sigma$. For $T_0 \in \Sigma^{**}$, $T \in \Sigma^{**}_{(x)}$, we denote by $T_0 \bullet_x T$ the tree obtained from T by replacing the leaf of label x with the subtree T_0. Suppose that $\mathcal{L} \subset \Sigma^{**}$ is a tree language. For $T_1, T_2 \in \Sigma^{**}$ we define $T_1 \sim_{\mathcal{L}} T_2$ if and only if the following holds $T_1 \bullet_x T \in \mathcal{L} \iff T_2 \bullet_x T \in \mathcal{L}$ for all $T \in \Sigma^{**}_{(x)}$. Obviously, $\sim_{\mathcal{L}}$ is an equivalence on Σ^{**}.

Theorem 6.1.1. (Myhill-Nerode theorem for tree automata, e.g. [16, Section 6.1]) *A tree language $\mathcal{L} \subset \Sigma^{**}$ is finite state if and only if the equivalence $\sim_{\mathcal{L}}$ has finite index over Σ^{**}.*

We also need to prove the next technical property of parse trees for matroids. (Recall the formalism of matroid parse trees from Section 3.3.)

Lemma 6.1.2. *Let T be a $\le t$-boundaried parse tree, let v be a vertex of T, and let T_v denote the subtree of T rooted at v. Then there exists a $\le t$-boundaried parse tree T' such that the matroids parsed by these trees satisfy $P(T) = \bar{P}(T_v) \mathbin{\bar{\oplus}} \bar{P}(T')$. Moreover, the tree T' depends only on $T - V(T_v)$, but not on T_v.*

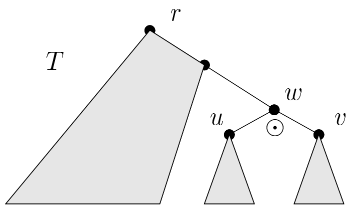


Figure 6.1: An illustration to the proof of Lemma 6.1.2.

Proof. We prove this statement by induction on the distance between v and the root r in T. If $v = r$, then the claim is trivial. Otherwise, let w be the parent of v in T. Up to symmetry, we may assume that u is a left son of w and v is the right son of w. By the inductive assumption, there is a $\le t$-boundaried parse tree T'_w such that $P(T) = \bar{P}(T_w) \mathbin{\bar{\oplus}} \bar{P}(T'_w)$. (See in Figure 6.1.)

We define the tree T'_v as follows: the root of T'_v is a new vertex w', the left subtree of w' is T_u, and the right subtree of w' is T'_w. Let $\odot = (R, \gamma_1, \gamma_2, \gamma_3)$ be the composition operator labelling w in T. We label w' in T'_v by the composition operator $\odot' = (R, \gamma_1, \gamma_3, \gamma_2)$. Then $\bar{P}(T'_v) = \bar{P}(T_u) \odot' \bar{P}(T'_w)$, and $\bar{P}(T_v) \mathbin{\bar{\oplus}} \bar{(}T'_v) = (\bar{P}(T_u) \odot' \bar{P}(T'_w)) \mathbin{\bar{\oplus}} \bar{P}(T_v) = (\bar{P}(T_u) \odot \bar{P}(T_v)) \mathbin{\bar{\oplus}} \bar{P}(T'_w) = \bar{P}(T_w) \mathbin{\bar{\oplus}} \bar{P}(T'_w) = P(T)$ using Lemma 2.3.2.

The second conclusion immediately follows from the fact that we have not used information about T_v when defining T'_v. ∎

Consider a fixed finite field $\mathbb{F}$. Let $\mathcal{B}_t$ be the set of all $\mathbb{F}$-represented matroids that have branch-width at most t. Let $\mathcal{T}_t \subset \Pi^{**}_{t-1}$ be the language of all $\le (t-1)$-boundaried parse trees over $\mathbb{F}$ with the alphabet Π_{t-1}, and let $\bar{\mathcal{B}}_t$ be the set of all $\le (t-1)$-boundaried matroids parsed by the trees from $\mathcal{T}_t$. We know from Theorem 3.3.2 that $N \in \mathcal{B}_t$ if and only if $N = \bar{N}' \mathbin{\bar{\oplus}} \bar{\Omega}$ for some $\bar{N}' \in \bar{\mathcal{B}}_t$. However, notice that not all $\le (t-1)$-boundaried matroids based on members of $\mathcal{B}_t$ necessarily belong to $\bar{\mathcal{B}}_t$.

It is easy to see that $\mathcal{T}_t$ itself is finite state. Suppose that $\mathcal{M}$ is a set of represented matroids. Let us denote by $\mathcal{M}_t = \mathcal{M} \cap \mathcal{B}_t$. We say that $\mathcal{M}$ is *t-finite state* if the collection of all parse trees parsing the members of $\mathcal{M}_t$ is finite state (i.e. recognizable by a tree automaton). For clarity, we sometimes use the expression "$\mathcal{M}$ is *t-finite state for branch-width at most t*". Moreover, we define an equivalence $\approx_{\mathcal{M}_t}$ for $\bar{N}_1, \bar{N}_2 \in \bar{\mathcal{B}}_t$ as follows: $\bar{N}_1 \approx_{\mathcal{M}_t} \bar{N}_2$ if and only if the boundary ranks of $\bar{N}_1, \bar{N}_2$ are equal, and if $\bar{N}_1 \mathbin{\bar{\oplus}} \bar{M} \in \mathcal{M}_t \iff \bar{N}_2 \mathbin{\bar{\oplus}} \bar{M} \in \mathcal{M}_t$ for all $\bar{M} \in \bar{\mathcal{B}}_t$ of the same boundary rank.

Theorem 6.1.3. *Let $t \geq 1$, and $\mathbb{F}$ be a finite field. A set of $\mathbb{F}$-represented matroids $\mathcal{M}$ is t-finite state if and only if the equivalence $\approx_{\mathcal{M}_t}$ has finite index over $\bar{\mathcal{B}}_t$.*

Proof. Let t be now fixed, and let $\Sigma = \Pi_{t-1}$. We denote by $\mathcal{V} \subseteq \mathcal{T}_t$ the collection of all parse trees corresponding to members of $\mathcal{M}_t = \mathcal{M} \cap \mathcal{B}_t$. The tree language $\mathcal{V}$ is finite state if and only if $\mathcal{M}$ is t-finite state by the definition. By Theorem 6.1.1, $\mathcal{V}$ is finite state if and only if the equivalence $\sim_{\mathcal{V}}$ has finite index over Σ^{**}. Recall that, for $T_1, T_2 \in \Sigma^{**}$, we define $T_1 \sim_{\mathcal{V}} T_2$ if and only if the following holds $T_1 \bullet_x T \in \mathcal{V} \iff T_2 \bullet_x T \in \mathcal{V}$ for all $T \in \Sigma^{**}_{(x)}$. Hence it is enough to prove that $\sim_{\mathcal{V}}$ has infinite index if and only if $\approx_{\mathcal{M}_t}$ has infinite index.

Suppose the latter — that there exist infinitely many $\bar{N}_k \in \bar{\mathcal{B}}_t$, $k \in \mathbb{N}$ such that, for all indices $i \neq j$, there is $\bar{M}_{i,j} \in \bar{\mathcal{B}}_t$ for which $\bar{N}_i \,\bar{\oplus}\, \bar{M}_{i,j} \in \mathcal{M}_t$ but $\bar{N}_j \,\bar{\oplus}\, \bar{M}_{i,j} \notin \mathcal{M}_t$, or vice versa. We may assume without loss of generality that all $\bar{N}_k$, $k \in \mathbb{N}$ are t'-boundaried matroids where $0 \leq t' \leq t-1$ is fixed. Let us denote by $T_k \in \mathcal{T}_t$ the parse tree for $\bar{N}_k$, and by $U_{i,j} \in \mathcal{T}_t$ the parse tree for $\bar{M}_{i,j}$. Let $\odot^o = (R^o, \gamma_1^o, \gamma_2^o, \gamma_3^o)$ be a composition operator such that $\gamma_3^o : \emptyset \to E(R^o)$, $\gamma_1^o, \gamma_2^o : [1, t'] \to E(R^o)$, and $\gamma_1^o(n), \gamma_2^o(n)$ are parallel elements for each $n \in [1, t']$.

We define a new tree $W_{i,j} \in \Sigma^{**}_{(x)}$ as follows: $W_{i,j}$ has a root r labelled $\odot^o$, the right subtree of r is $U_{i,j}$, and the left son of r is a leaf labelled x. Notice that $P(T_i \bullet_x W_{i,j}) = \bar{N}_i \,\bar{\oplus}\, \bar{M}_{i,j}$, etc. Thus $T_i \bullet_x W_{i,j} \in \mathcal{V}$ but $T_j \bullet_x W_{i,j} \notin \mathcal{V}$, or vice versa; and so the trees $W_{i,j}$ certify that the parse trees T_k, $k \in \mathbb{N}$ are pairwise nonequivalent in $\sim_{\mathcal{V}}$.

Now suppose the former — that there are infinitely many trees $T_k \in \Sigma^{**}$, $k \in \mathbb{N}$ such that, for all indices $i \neq j$, there is $W_{i,j} \in \Sigma^{**}_{(x)}$ for which $T_i \bullet_x W_{i,j} \in \mathcal{V}$ but $T_j \bullet_x W_{i,j} \notin \mathcal{V}$, or vice versa. We may assume without loss of generality that $T_k \in \mathcal{T}_t$ are parse trees for all $k \in \mathbb{N}$ since all trees from $\Sigma^{**} - \mathcal{T}_t$ are equivalent in $\sim_{\mathcal{V}}$, and that all $\bar{P}(T_k)$, $k \in \mathbb{N}$ are t'-boundaried matroids where $0 \leq t' \leq t-1$ is fixed. Consider, up to symmetry, indices $i \neq j$ such that $T_i \bullet_x W_{i,j} \in \mathcal{V} \subseteq \mathcal{T}_t$. Our assumptions now imply that also $T_j \bullet_x W_{i,j} \in \mathcal{T}_t$, as can be easily checked from the definition of a $\leq(t-1)$-boundaried parse tree.

We denote by $\bar{N}_k = \bar{P}(T_k)$. By Lemma 6.1.2, there exists a tree $U_{i,j} \in \mathcal{T}_t$ such that $P(T_i \bullet_x W_{i,j}) = \bar{N}_i \,\bar{\oplus}\, \bar{P}(U_{i,j})$ and $P(T_j \bullet_x W_{i,j}) = \bar{N}_j \,\bar{\oplus}\, \bar{P}(U_{i,j})$. Let $\bar{M}_{i,j} = \bar{P}(U_{i,j})$. Recall that $T_i \bullet_x W_{i,j} \in \mathcal{V}$ but $T_j \bullet_x W_{i,j} \notin \mathcal{V}$, or vice versa, for all $i \neq j$. Then, by the definition of $\mathcal{V}$, the t'-boundaried matroids $\bar{M}_{i,j} \in \bar{\mathcal{B}}_t$ certify that the t'-boundaried matroids $\bar{N}_k \in \bar{\mathcal{B}}_t$, $k \in \mathbb{N}$ belong to pairwise different equivalence classes of $\approx_{\mathcal{M}_t}$. ∎

Remark. The property "$\approx_{\mathcal{M}_t}$ has finite index" is also known as "t-cutset regularity". We add an informal remark to this interesting concept since its formal definition may sound confusing. The true meaning of $\approx_{\mathcal{M}_t}$ having finite index is that, regardless of a choice of $\bar{N} \in \bar{\mathcal{B}}_t$, only a bounded amount of information concerning membership in $\mathcal{M}_t$ may "cross" the boundary of $\bar{N}$.

6.2 Monadic Second-Order Logic

Our paper has been inspired by the well-known "MS_2-theorem" (Theorem 6.2.1) for graphs. This theorem is a high-level theoretical tool for establishing that various hard graph properties, expressible in monadic second-order theory of graphs, are t-finite state for graphs of tree-width at most t, i.e. recognizable by finite tree automata. This section defines the monadic second-order logic, and states the associated results.

Definition. The syntax of the *monadic second-order logic MS_2 of graphs* includes variables for vertices, edges, and their sets, the quantifiers $\forall, \exists$ applicable to these variables, the logical connectives $\wedge, \vee, \neg$, and the next binary relations:

1. $=$, the equality for vertices, edges, and their sets,
2. $v \in W$, where v is a vertex and W is a vertex set variables,
3. $e \in F$, where e is an edge and F is an edge set variables,
4. $\mathrm{inc}(v, e)$, where v is a vertex variable and e is an edge variable, and the relation tells whether v is incident with e.

Actually, the above language is often called an *extended* second-order monadic logic, as opposed to a basic second-order monadic logic MS_1 which has only variables for vertices and vertex sets, and a relation for adjacent vertices.

Parse trees for graphs of bounded tree-width had been defined before in [1], and they are analogous to our parse trees for matroids of bounded branch-width in Section 3.3. Let $\mathcal{G}$ be a graph family. Analogously to the previous section, we say that $\mathcal{G}$ is *t-finite state* if the collection of graph parse trees corresponding to the tree-width-t members of $\mathcal{G}$ is finite state. The MS_2-theorem (in a tree-automata formulation) follows.

Theorem 6.2.1. (Courcelle [10, 11, 12]) *Let $t \geq 1$. If $\mathcal{G}$ is a family of graphs described by a sentence in the second-order monadic logic MS_2, then $\mathcal{G}$ is t-finite state for graphs of tree-width at most t.*

Similar results were independently obtained also by Arnborg, Lagergren, and Seese [3], and later by Borie, Parker, and Tovey [9].

As noted in the introduction, the concept of a branch-width is very close to that of a tree-width in graphs. If we want to extend Theorem 6.2.1 to matroids, we have to define a similar monadic second-order logic of matroids. The trouble is that there is no analogue of graph vertices in matroids. We thus start in the first step with description of an *edge-only monadic second-order logic of graphs MS_e*. The syntax of MS_e includes variables for edges and edge sets, the quantifiers $\forall, \exists$ applicable to these variables, the logical connectives $\wedge, \vee, \neg$, and the next predicates:

1. $=$, the equality for edges and their sets,

2. $e \in F$, where e is an edge and F is an edge set variables,

3. $\mathrm{star}(F)$, where F is an edge set variable, and the predicate tells whether F is the set of all edges incident with some vertex (a "star").

It is clear that any sentence in MS_e is expressible in MS_2. For example, $\mathrm{star}(F) \equiv \exists v \forall e\,(\neg\,\mathrm{inc}(v,e) \vee e \in F)$. It is known that the above mentioned MS_1 logic is strictly weaker than MS_2. On the other hand, we show that the MS_e logic is equivalent to the MS_2 logic for connected graphs.

Lemma 6.2.2. *In the universe of connected graphs on more than 2 vertices, any sentence in the MS_2 logic is expressible in the MS_e logic.*

Proof. Let ϕ be a MS_2 formula. If v is a vertex variable in ϕ, then we replace it with an edge-set variable F_v and add a predicate $\mathrm{star}(F_v)$ in conjunction. Notice that $F_v = F_w$ implies $v = w$ in a connected multigraph on more than 2 vertices. We then replace each predicate $\mathrm{inc}(v,e)$ by $e \in F_v$. The only tricky part in the proof is replacing vertex-set variables in ϕ. Notice that equality of vertex-sets can be written as $V = W \equiv \forall v(v \in V \leftrightarrow v \in W)$.

Let G be a connected graph, and let $U \subseteq V(G)$, $W = V(G) - U$. We choose an arbitrary $u \in U$. We express the set U as a pair (F_u, X_U) where F_u are the edges incident with u and X_U are the edges of G between U and W. We then add the condition $\mathrm{star}(F_u) \wedge \forall Y(Y \not\subseteq X_U \vee \neg \mathit{cycle}(Y) \vee \mathit{even\_cycle}(Y))$ where $\mathit{cycle}(C) \equiv \mathit{deg}(C,2)$ and $\mathit{even\_cycle}(C) \equiv \exists D, D'(D \cup D' = C \wedge \mathit{deg}(D,1) \wedge \mathit{deg}(D',1))$. Here $\mathit{deg}(X,n) \equiv \forall F(\neg\,\mathrm{star}(F) \vee |F \cap X| = n \vee |F \cap X| = 0)$. (The relations $\subseteq, \cup, \cap$, and the predicates $|X| = 0, 1, 2$ have obvious implementations.)

Let us have the pair (F_u, X_U) for some $u \in U$ as above. Then $v \in U$ if and only if there exists a path with ends u, v that uses edges of X_U even number of times. This can be expressed as $\exists P_1, P_2, P_3, P_4[(P_2 \cup P_4 \subseteq X_U) \wedge ((P_1 \cup P_3) \cap X_U = \emptyset) \wedge \forall F(\neg\,\mathrm{star}(F) \vee (F = F_u \wedge |F \cap (P_1 \cup P_2)| = 1 \wedge |F \cap (P_3 \cup P_4)| = 0) \vee (F = F_v \wedge |F \cap (P_1 \cup P_4)| = 1 \wedge |F \cap (P_2 \cup P_3)| = 0) \vee (|F \cap (P_1 \cup P_2)| = 1 \wedge |F \cap (P_1 \cup P_4)| = 1) \vee (|F \cap (P_2 \cup P_3)| = 1 \wedge |F \cap (P_3 \cup P_4)| = 1) \vee (|F \cap (P_1 \cup P_2 \cup P_3 \cup P_4)| = 0))]$. The meaning of the formula is that $P = P_1 \cup P_2 \cup P_3 \cup P_4$ forms a path between u, v, and P_2 (P_4) are the odd (even) edges of $P \cap X_U$. ∎

6.3 MSO Logic of Matroids

We now define the *monadic second-order logic of matroids MS_M*. Notice that the logic speaks about abstract matroids in general.

Definition. The syntax of MS_M includes variables for matroid elements and element sets, the quantifiers $\forall, \exists$ applicable to these variables, the logical connectives $\wedge, \vee, \neg$, and the next predicates:

1. $=$, the equality for elements and their sets,

2. $e \in F$, where e is an element and F is an element set variables,

3. $\mathrm{indep}(F)$, where F is an element set variable, and the predicate tells whether F is independent in the matroid.

Recall that if M is the cycle matroid of a graph G, then a set $F \subseteq E(M)$ is independent if and only if F contains no circuits in G. It is well known that such a graphic matroid M can be represented over any field. Two non-isomorphic graphs may have isomorphic cycle-matroids if they are not 3-connected. On the other hand, we show that a (vertex-) 3-connected simple graph may be handled via its cycle-matroid.

Lemma 6.3.1. *Let M be the cycle-matroid of a loopless 3-connected graph G. Then any sentence about G in the MS_2 logic can be expressed as a sentence about M in the MS_M logic.*

Proof. According to Lemma 6.2.2, we only need to express the predicate $\mathrm{star}(F)$ in the MS_M logic. We actually describe the complement $\bar{F} = E(G) - F$. The predicate $\mathrm{star}(F)$ is equivalent to the statement that $\bar{F}$ is a 2-connected closed set (of edges), and every strictly larger closed set contains all edges of G. (It is not enough to consider connected $\bar{F}$ since graph 1-connectivity has no matroidal analogue.)

We write $\mathit{closed}(X) \equiv \neg\exists e \exists C(C \subseteq X \cup \{e\} \wedge e \in C \wedge e \notin X \wedge \mathit{circuit}(C))$, and $\mathit{connected}_2(X) \equiv \forall e \neq e'(\{e,e'\} \not\subseteq X \vee \exists C(C \subseteq X \wedge e, e' \in C \wedge \mathit{circuit}(C)))$, where $\mathit{circuit}(C) \equiv (\neg\,\mathrm{indep}(C) \wedge \forall e(e \notin C \vee \mathrm{indep}(C - \{e\})))$. We leave remaining technical details to the reader. ∎

In order to apply Lemma 6.3.1 to an arbitrary graph G, we have to make the graph 3-connected by adding extra vertices. Let $G \uplus H$ denotes the graph obtained from disjoint copies of G and H by adding all edges between them.

Lemma 6.3.2. *Let G be a loopless graph. Then any sentence about G in the MS_2 logic can be expressed by an MS_2 sentence about $G \uplus K_3$.*

Proof. Clearly, the graph $G' = G \uplus K_3$ is a loopless 3-connected graph. It is enough to translate an MS_2 formula ϕ about G into an MS_2 formula ϕ' about G': Indeed, we identify the three added vertices in G' by variables a_1, a_2, a_3 and predicates $apex(a_i) \equiv \forall v(v = a_i \vee adj(a_i, v, 1))$ where $adj(x, y, n)$ says that there are exactly n edges joining x to y. (If there are more than three vertices satisfying $apex(x)$ in G', then they are all equivalent.) We then add predicates excluding a_1, a_2, a_3 from every other vertex or vertex-set variable in ϕ, and predicates excluding edges incident to a_1, a_2, a_3 from every other edge or edge-set variable in ϕ. Again, we leave technical details to the reader. ∎

In fact, one may use a more involved construction with subdividing all edges of G in the above proof, thus handling also possible loops in G. However, the arguments would be too complicated to be presented here.

It is also possible to define a *counting MS_2 logic* of graphs by adding predicates $\mathrm{mod}_{p,q}$ for integer constants $0 \leq p < q$, where the interpretation of $\mathrm{mod}_{p,q}(X)$ for a (vertex or edge) set variable X is that $|X| \bmod q = p$. The counting MS_2 logic is known to be stronger [12] than plain MS_2 over all graphs. However, counting gives no advantage when working over already "well tree-structured" graphs. The full statement of Theorem 6.2.1 actually reads:

Theorem 6.2.1'. (Courcelle) *If $\mathcal{G}$ is a family of graphs described by a sentence in the counting second-order monadic logic MS_2, then $\mathcal{G}$ is t-finite state for graphs of tree-width at most t.*

Similarly, we define the *counting MS_M logic* of matroids by adding the predicates $\mathrm{mod}_{p,q}$ for sets of elements.

Theorem 6.3.3. *Let G be a loopless graph, and let M be the cycle matroid of $G \uplus K_3$. Then any sentence about G in the counting MS_2 logic can be expressed as a sentence about M in the counting MS_M logic.*

Proof. Let ϕ be an MS_2 formula about G. We now apply Lemmas 6.3.2,6.3.1. Hence we may express ϕ as an MS_M formula ϕ' about M, up to possible counting predicates in ϕ. If $\mathrm{mod}_{p,q}(F)$ is a counting predicate in ϕ where F is an edge-set variable, then we may clearly replace it by the corresponding MS_M predicate $\mathrm{mod}_{p,q}(F)$ where F is (the same) element set in ϕ'.

To handle a possible counting predicate $\mathrm{mod}_{p,q}(U)$ for a vertex-set variable U in ϕ, we use a little trick. It is easy to see that if H is an arbitrary connected graph and $X \subset V(H)$, $X \neq V(H)$, then there exists an acyclic edge set F_X in H such that all edges in F_X are incident with X, and that each connected component of F_X induced by X is incident with exactly one edge of F_X not belonging to this component. Then $|F_X| = |X|$. It is routine work to write down an MS_2 formula relating F_X to X. We thus, after applying Lemma 6.3.2, replace the predicate $\mathrm{mod}_{p,q}(U)$ with the predicate $\mathrm{mod}_{p,q}(F_U)$ in the graph $G \uplus K_3$, and then apply Lemma 6.3.1. ∎

6.4 MS_M-Theorem for Represented Matroids

In this section we formulate and prove a natural extension of Theorem 6.2.1 to matroids represented over finite fields. We follow the ideas of the Abrahamson-Fellows' proof [1] of Theorem 6.2.1, in the form published in [16, Section 6.5].

Theorem 6.4.1. *Let $\mathbb{F}$ be a finite field, and $t \geq 1$. If $\mathcal{M}$ is a set of $\mathbb{F}$-represented matroids described by a sentence in the counting second-order monadic logic MS_M of matroids, then $\mathcal{M}$ is t-finite state for branch-width at most t.*

Recall the sets $\mathcal{B}_t$ and $\bar{\mathcal{B}}_t$ defined for $\mathbb{F}$, $\mathcal{M}_t = \mathcal{M} \cap \mathcal{B}_t$, and the equivalence relation $\approx_{\mathcal{M}_t}$ defined over $\bar{\mathcal{B}}_t$ by $\mathcal{M}_t$, from Section 6.1. Let ϕ be the sentence in MS_M logic that describes the set $\mathcal{M}$. Let us consider an integer $0 \leq u \leq t-1$, and write $\approx_\phi$ for $\approx_{\mathcal{M}_{u+1}}$ restricted to exactly u-boundaried matroids over $\mathbb{F}$. We are going to prove the theorem by induction on the length of ϕ.

In order to use induction, we must slightly generalize the setting. We allow a formula ϕ with (possible) free variables, and we associate with such ϕ an equivalence relation $\approx_\phi$ on the set of all u-boundaried represented matroids over $\mathbb{F}$ that are partially "equipped" with distinguished elements and sets corresponding to the free variables $Free(\phi)$ in ϕ. The relation $\approx_\phi$ naturally generalizes the relation $\approx_{\mathcal{M}_t}$ to partially equipped boundaried matroids.

Let $Free(\phi) = Fr(\phi) \cup FR(\phi)$ be the partition of the free variables into those $Fr = Fr(\phi)$ for elements and $FR = FR(\phi)$ for sets of elements. We define a *partial equipment signature* as a triple $\sigma = (Fr, FR, f)$ where $f : Fr \to \{0, 1\}$. A boundaried represented matroid $\bar{M}$ is said to be *σ-partially equipped* if it has distinguished elements and element sets corresponding to (some) free variables in σ. Formally, for each variable $X \in FR$ there is a distinguished subset $S_X \subseteq E(\bar{M} \oplus \bar{\Omega})$, and for each $x \in Fr$ such that $f(x) = 0$ there is a distinguished element $e_x \in E(\bar{M} \oplus \bar{\Omega})$. We say that a partial equipment signature $\sigma' = (Fr, FR, f')$ is a *complement* of σ if $f'(x) = 1 - f(x)$ for all $x \in Fr$.

The importance of a complementary partial equipment signature σ' is in the following fact: If $\bar{N}_1, \bar{N}_2$ are boundaried matroids such that $\bar{N}_1$ is σ-partially equipped and $\bar{N}_2$ is σ'-partially equipped, then the free variables from $Fr \cup FR$ have consistent full interpretation over the whole matroid $\bar{N}_1 \oplus \bar{N}_2$. A represented matroid M is *fully equipped* for ϕ if all free variables from $Free(\phi)$ have interpretation in $E(M)$. We write $M \models \phi$ to mean that the formula ϕ is true with the associated full interpretation for $Free(\phi)$.

Definition. Let σ be a partial equipment signature for a formula ϕ, and let σ' be the complement of σ. Suppose that $\bar{N}_1, \bar{N}_2$ are σ-partially equipped u-boundaried matroids. We define $\bar{N}_1 \approx_\phi \bar{N}_2$ if and only if $\bar{N}_1 \oplus \bar{M} \models \phi \iff \bar{N}_2 \oplus \bar{M} \models \phi$ for every σ'-partially equipped u-boundaried matroid $\bar{M}$.

We may now precisely formulate our induction statement.

Lemma 6.4.2. *Let ϕ be a formula in the MS_M logic of matroids, and let σ be a partial equipment signature for ϕ. Then $\approx_\phi$ has finite index on the universe of σ-partially equipped u-boundaried matroids.*

Proof. Let $\sigma = (Fr, FR, f)$ where $Free(\phi) = Fr \cup FR$. Unless stated otherwise, we implicitly consider σ-partially equipped u-boundaried matroids. We first show that the statement holds for atomic formulas. The proofs for equality formulas are trivial. For example, if ϕ is $x = y$ where $x, y \in Fr$ and $f(x) = f(y) = 0$, then $\approx_\phi$ has index 2. Similarly, if $f(x) = 0$ and $f(y) = 1$, then $\approx_\phi$ has index 1, etc. If ϕ is $x \in X$, then $\approx_\phi$ has index 2 for $f(x) = 0$, while $\approx_\phi$ has index 1 for $f(x) = 1$. If ϕ is $\mathrm{mod}_{p,q}(X)$ where $X \in FR$, then $\approx_\phi$ clearly has index q.

The only interesting atomic formula is $\phi = \mathrm{indep}(X)$. Suppose that $\bar{N}_i = (N_i, \delta_i)$, $i = 1, 2$ are σ-partially equipped boundaried matroids, and that $F_X^i \subseteq E(N_i)$ is an interpretation of the set variable X in $\bar{N}_i$. By the definition of a boundary sum and elementary linear algebra, $\bar{N}_1 \approx_\phi \bar{N}_2$ if and only if either both F_X^1 and F_X^2 are dependent, or the subspaces $\langle F_X^1 \rangle \cap \partial(\bar{N}_1)$ and $\langle F_X^2 \rangle \cap \partial(\bar{N}_2)$ are equivalent in the linear transformation $\mathcal{L}$ matching the boundary points of $\bar{N}_1$ to those of $\bar{N}_2$ in order (i.e. $\delta_2(j) = \mathcal{L}(\delta_1(j))$ for $j \in [1, u]$). Hence $\approx_\phi$ has index $1 + p(\mathbb{F}, u)$ where $p(\mathbb{F}, n)$ stands for the number of all distinct subspaces of the finite projective space $PG(n, \mathbb{F})$.

For the inductive step, we consider that the formula ϕ is created from shorter formula(s) in one of the following ways: $\phi = \neg\phi_1$, $\phi = \phi_1 \wedge \phi_2$, $\phi = \exists x \phi_1(x)$ for $x \in Fr(\phi_1)$, or $\phi = \exists X \phi_1(X)$ for $X \in FR(\phi_1)$. One may easily express $\vee$ or $\forall$ symbols using these. We assume that $\approx_{\phi_1}$ ($\approx_{\phi_2}$) has finite index. The first case is quite easy — the equivalence $\approx_\phi$ is, in fact, the same as $\approx_{\phi_1}$.

Let us look at the second case $\phi = \phi_1 \wedge \phi_2$. Notice that $Free(\phi) = Free(\phi_1) \cup Free(\phi_2)$, and let σ_i denote the restriction of σ to $Free(\phi_i)$, $i = 1, 2$. Suppose that $\bar{N}_1 \not\approx_\phi \bar{N}_2$ are two σ-partially equipped boundaried matroids, and that $\bar{M}$ is such that $\bar{N}_1 \bar{\oplus} \bar{M} \models \phi$ but $\bar{N}_2 \bar{\oplus} \bar{M} \models \neg\phi$. Then, for some $i \in \{1, 2\}$, the boundaried matroids $\bar{N}_1, \bar{N}_2, \bar{M}$ (considered now with restricted σ_i-partial equipment) must satisfy $\bar{N}_1 \bar{\oplus} \bar{M} \models \phi_i$ but $\bar{N}_2 \bar{\oplus} \bar{M} \models \neg\phi_i$. Thus the equivalence classes of $\approx_\phi$ are unions of equivalence classes of $\approx_{\phi_1} \cap \approx_{\phi_2}$, and so $\approx_\phi$ has finite index.

Let us look at the third case $\phi = \exists x \phi_1(x)$ for $x \in Fr(\phi_1)$. Again, suppose that $\bar{N}_1 \not\approx_\phi \bar{N}_2$ and $\bar{M}$ are such that $\bar{N}_1 \bar{\oplus} \bar{M} \models \phi$ but $\bar{N}_2 \bar{\oplus} \bar{M} \models \neg\phi$. In other words, $\bar{N}_1 \bar{\oplus} \bar{M} \models \exists x \phi_1(x)$. Let $e_x \in E(\bar{N}_1 \bar{\oplus} \bar{M})$ be the interpretation of the variable x that satisfies ϕ_1 over $\bar{N}_1 \bar{\oplus} \bar{M}$, and let $\sigma_1 = (Fr \cup \{x\}, FR, f_1)$ be the partial equipment signature of $\bar{N}_1$ induced by σ and the interpretation e_x of x.

(That means $f_1(x) = 1$ iff $e_x \in E(\bar{M} \bar{\oplus} \bar{\Omega})$.) Moreover, $\bar{N}_2 \bar{\oplus} \bar{M} \models \neg\phi_1$ for all σ_1-partially equipped u-boundaried matroids $\bar{N}_2$ and complementary σ_1'-partially equipped matroids $\bar{M}$, since $\bar{N}_2 \bar{\oplus} \bar{M} \models \forall x \neg\phi_1(x)$. Therefore, if $\bar{N}_1 \not\approx_\phi \bar{N}_2$ when considered as σ-partially equipped boundaried matroids, then $\bar{N}_1 \not\approx_{\phi_1} \bar{N}_2$ for some σ_1-partial equipments of $\bar{N}_1$ and of $\bar{N}_2$. So $\approx_\phi$ has finite index as well.

Finally, we are left with the fourth case $\phi = \exists X \phi_1(X)$ for $X \in FR(\phi_1)$. The arguments are very similar to those in the previous paragraph, and we leave details to the reader. ∎

We apply Lemma 6.4.2 to the (closed) formula ϕ describing the set $\mathcal{M}$ in Theorem 6.4.1 and to $u = 0, 1, \ldots, t-1$, showing that $\approx_{\mathcal{M}_{u+1}}$ has finite index for each u when restricted to exactly u-boundaried matroids. (Notice that we have established above more than what was required — we have proved that $\approx_{\mathcal{M}_{u+1}}$ has finite index over all u-boundaried matroids, not only over boundaried matroids from $\bar{\mathcal{B}}_{u+1}$.) Hence $\approx_{\mathcal{M}_t}$ has finite index for each t, and the proof of Theorem 6.4.1 is finished by Theorem 6.1.3.

Remark. We note in passing, that the proofs of Lemma 6.4.2 and of Theorem 6.1.3 are constructive in the following sense: If ϕ is a given MS_M formula, then we can construct the equivalence classes of $\approx_\phi$ as defined above. These equivalence classes then, essentially, provide the states of the constructed finite tree automaton in Theorem 6.1.3. Hence there is an algorithm that computes this finite tree automaton for the given width t, finite field $\mathbb{F}$, and formula ϕ.

If G is a loopless graph of tree-width t, then the graph $G' = G \uplus K_3$ has tree-width at most $t+3$, and so branch-width at most $t+4$ by [41]. Hence the branch-width of the cycle matroid $M = M(G')$ of G' is at most $t+4$ as well. The cycle matroids of graphs are representable over any finite field, as already noted above. Therefore Courcelle's Theorem 6.2.1 for (loopless) graphs follows from Theorem 6.4.1 and Theorem 6.3.3.

It may be, on the other hand, possible to prove Theorem 6.4.1 using the logic method of "interpretability" in tree-like structures. That is the approach originally taken for graphs by Courcelle, and by Arnborg, Lagergren and Seese [3]. The general interpretability method for arbitrary structures is also surveyed in [44]. However, to apply the interpretability method onto represented matroids, one would still need to develop a theory of matroid parse trees similar to that in Section 3.3, and to argue how an MS_M formula acts on a boundary of a fixed rank, as we do here. We prefer the direct combinatorial approach in our paper.

6.5 Algorithmic Aspects

Since the computation of a finite tree automaton may be easily emulated by a linear-time algorithm, we get the following corollary of Theorem 6.4.1.

Corollary 6.5.1. *Let $t \geq 1$, let $\mathbb{F}$ be a finite field, and let ϕ be a sentence in the counting second-order monadic logic MS_M of matroids. There is an algorithm that, for given $\leq t$-boundaried parse tree T over $\mathbb{F}$, decides in linear time whether ϕ is true for the $\mathbb{F}$-represented matroid $P(T)$ parsed by T.*

∎

To use Corollary 6.5.1 in a practical computation, we first have to construct a parse tree for the given represented matroid of bounded branch-width. This can be done by Theorem 4.4.1. Moreover, by the above remark, we may construct the tree automaton from Theorem 6.4.1 algorithmically. So we claim:

Theorem 6.5.2. *Let $t \geq 1$, let $\mathbb{F}$ be a finite field, and let ϕ be a sentence in the counting second-order monadic logic MS_M of matroids. There is an algorithm that, given a rank-r matrix $\mathbf{A} \in \mathbb{F}^{r\times n}$ of branch-width at most t, tests whether ϕ is true for the matroid $M(\mathbf{A})$ in time $O(n^3)$. Hence the matroid property described by ϕ is uniformly fixed-parameter tractable with respect to the (combined) parameter $\langle \mathbb{F}, t, \phi\rangle$.*

∎

The main result of [19] implies that, for any minor-closed matroid family $\mathcal{M}$ and any k, there are only finitely many $\mathbb{F}$-representable excluded minors for $\mathcal{M}$ of branch-width at most k. (This result is non-constructive.) So, by Lemma 4.4.2, membership in $\mathcal{M}$ can be expressed via an MS_M sentence with respect to $\mathbb{F}$ and k, and we may conclude:

Corollary 6.5.3. *Let $\mathbb{F}$ be a finite field, let t be an integer constant, and let $\mathcal{M}$ be a minor-closed matroid family. There is an algorithm that, given a rank-r matrix $\mathbf{A} \in \mathbb{F}^{r\times n}$ of branch-width at most t, tests whether the matroid $M(\mathbf{A})$ belongs to $\mathcal{M}$ in time $O(n^3)$. Hence the matroid family $\mathcal{M}$ is (non-uniformly) fixed-parameter tractable with respect to the (combined) parameter $\langle \mathbb{F}, t\rangle$.*

∎

More similar algorithmic applications can be found in other parts of this work.

We now show that it is crucial to consider matroids represented over finite fields in Theorem 6.4.1. This is explained in the next example: Let $\mathbb{Q}$ denote the rational numbers. We are going to show that even a quite simple matroidal property may generate an equivalence relation of an infinite index over ≤ 2-boundaried

$\mathbb{Q}$-represented matroids, and so this property is not recognizable by a finite tree automaton by Theorem 6.1.3. Analogous examples exist for other infinite fields.

We say that a matroid M is *identically self-dual* if, for each basis $B \subseteq E(M)$ of M, the set $E(M)-B$ is also a basis of M. It is easy to describe this property in MS_M logic. We write $\mathit{id\_selfdual} \equiv \forall X\, \exists Y\, (\mathit{part}(X,Y) \wedge (\neg \mathit{basis}(X) \vee \mathit{basis}(Y)))$, where $\mathit{basis}(B) \equiv \mathrm{indep}(B) \wedge \forall e(e \in B \vee \neg\, \mathrm{indep}(B \cup \{e\}))$ and $\mathit{part}(X,Y) \equiv \forall e((e \in X \vee e \in Y) \wedge \neg(e \in X \wedge e \in Y))$. Then, using Theorem 6.1.3 and arguments very similar to those presented in Chapter 7, we can prove (see [27] for details):

Proposition 6.5.4. *The ≤ 2-boundaried parse trees, parsing the $\mathbb{Q}$-represented branch-width-3 matroids described by the MS_M sentence $\mathit{id\_selfdual}$, cannot be recognized by a finite tree automaton. Moreover, the same conclusion holds even when the input is restricted to matroids represented by square integer matrices with entries from the set $\{-1, 1, 3\}$.*

∎

Lastly, we discuss the following interesting result of Seese [43].

Theorem 6.5.5. (Seese) *Suppose that $\mathcal{G}$ is a family of graphs with a decidable monadic second-order (MS_2) theory. Then there is a number t such that the tree-width of each $G \in \mathcal{G}$ is at most t.*

Notice that this theorem speaks about the whole MS_2 theory, not about a particular sentence. The theorem does not seem to directly apply to our case, but Seese's proof idea might. The idea is that, having sufficiently large grid minors in the family $\mathcal{G}$, one may encode an algorithmically unsolvable problem by a suitable MS_2 sentence over an (arbitrarily large) grid. Hence if the MS_2 theory is decidable on $\mathcal{G}$, then the size of the largest grid minor in $\mathcal{G}$ has to be bounded, and so $\mathcal{G}$ has bounded tree-width by [41].

An analogous excluded-grid statement for matroid branch-width is still an open problem, but many matroid researchers strongly believe it is true at least over finite fields. Having such a result at hand, one could prove a result analogous to Theorem 6.5.5 for represented matroids.

Chapter 7

Hard Problems for Branch-Width 3

To conclude Part II of our work, we present bounds for how far the previous results could be extended to matroids in general. While studying the previous chapter, the reader may have asked why only matroids represented over finite fields appear in our statements. One reason was already mentioned above — the notion of a "k-sum" is not well defined for abstract matroids if $k > 2$. However, it is not just that we cannot formulate our ideas for other matroids; the reasons are much deeper.

In Chapter 6, we have shown that Theorem 6.4.1 is false for $\mathbb{Q}$-represented matroids of branch-width 3. Moreover, we are going to prove now that there is likely no polynomial algorithm for deciding MSOL sentences in general, or for computation of the Tutte polynomial, when dealing with bounded-width matroids represented over infinite fields (like over the rational numbers). This even applies to matroids of branch-width three, and it obviously covers abstract matroids. Our proof uses a reduction to the *Partition* problem — one of the 6 basic NP-complete problems in Garey-Johnson.

The content of this chapter is taken from our original paper [28].

7.1 Matroid Class of Spikes

The class of matroids called spikes plays a quite important role in structural matroid theory. Let us denote by $\boldsymbol{D}(x_1, \ldots, x_n) = [d_{i,j}]_{i=1}^{n}$ an $n \times n$ matrix such that $d_{i,j} = 1$ if $i \neq j$, and $d_{i,i} = x_i$ for $i \in [1, n]$.

Definition. Let $n \geq 3$ and N be a matroid circuit on the ground set $e_0, e_1, \ldots, e_n$. Denote by N' an arbitrary simple matroid obtained from N by adding n new elements f_i for $i \in [1, n]$ such that e_0, e_i, f_i are triangles. Then the matroid $S = N' \setminus e_0$ obtained by deleting the element e_0 is called a *rank-n spike*. The pairs $\{e_i, f_i\}$, $i \in [1, n]$ are called the *legs* of the spike.

Spikes more or less explicitly appear in several research papers in structural matroid theory. Perhaps the first explicit description of a spike is given in [15], and among recent papers we mention [19]. There seems to be no "usual definition" of a spike; the above definition was suggested by Whittle. Actually, since

$$
\begin{array}{c}
\begin{array}{cccccccccc}
e_1 & e_2 & \cdots & e_{n-1} & e_n & f_1 & f_2 & \cdots & f_{n-1} & f_n
\end{array}
\\
\begin{array}{c} e_1 \\ e_2 \\ \vdots \\ e_{n-1} \\ e_n \end{array}
\left[\begin{array}{ccccc|ccccc}
1 & 0 & \cdots & 0 & 0 & x_1 & 1 & \cdots & 1 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & x_2 & 1 & 1 & 1 \\
\vdots & 0 & \ddots & 0 & \vdots & \vdots & 1 & \ddots & 1 & \vdots \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & x_{n-1} & 1 \\
0 & 0 & \cdots & 0 & 1 & 1 & 1 & \cdots & 1 & x_n
\end{array}\right]
\begin{array}{c} e_1 \\ e_2 \\ \vdots \\ e_{n-1} \\ e_n \end{array}
\end{array}
$$

Figure 7.1: Matrix $[\boldsymbol{I}_n \,|\, \boldsymbol{D}(x_1, \ldots, x_n)]$ representing the spike $S(x_1, \ldots, x_n)$.

we deal only with representable spikes, we focus on matrix representations of spikes, as presented in Figure 7.1 and Proposition 7.1.1.

Consider an arbitrary field $\mathbb{F}$, and $n \geq 3$. When speaking about a field, we mean a commutative field. The following basic properties of spikes are known, for example [15, 19], and they are easy to show using elementary linear algebra.

Proposition 7.1.1. *Let $\mathbb{F}$ be a field and $n \geq 3$. A rank-n spike S is representable over $\mathbb{F}$ if and only if S is represented by a matrix $[\boldsymbol{I}_n \,|\, \boldsymbol{D}(x_1, \ldots, x_n)]$ for some $x_1, \ldots, x_n \in \mathbb{F} - \{1\}$.*

Proof. One direction is clear. Conversely, we look at a spike S. We represent the basis $\{e_1, \ldots, e_n\}$ of S by the unit vectors of $\boldsymbol{I}_n$. Then the element e_0 from the definition of a spike is, up to scaling, represented by the vector $\boldsymbol{1}$. Hence the element f_i colinear with e_0, e_i is represented by a vector $\boldsymbol{1} + (x_i - 1)\boldsymbol{e}_i$ for $i \in [1, n]$ and some $x_i \in \mathbb{F} - \{1\}$. ∎

According to the above convention, the matrix $\boldsymbol{D} = \boldsymbol{D}(x_1, \ldots, x_n)$ is a reduced representation of a spike that displays the basis $\{e_1, \ldots, e_n\}$. The rows of $\boldsymbol{D}$ are labelled by $e_1, \ldots, e_n$ and the columns by $f_1, \ldots, f_n$ in this order. (See in Figure 7.1.) We denote that spike by $S = S(x_1, \ldots, x_n)$, and call S a *rank-n spike over* $\mathbb{F}$. The next two claims are true and known for spikes in general, but we intentionally prove them for representable spikes to illustrate further concepts.

Lemma 7.1.2. *Let S be a rank-n spike over a field $\mathbb{F}$ for $n \geq 5$. Then there are no circuits smaller than 4 in S, and all circuits of length 4 are formed by pairs of legs of S.*

Proof. Let $E' = \{e_1, \dots, e_n\}$ and $F' = \{f_1, \dots, f_n\}$ be as in the definition of a spike, and let S have the reduced representation $\boldsymbol{D} = \boldsymbol{D}(x_1, \dots, x_n)$ over $\mathbb{F}$. Suppose that C is a circuit in S. Then $C \not\subseteq E'$. If $C \subseteq F'$, then clearly $C = F'$ and $|C| \geq 5$. If $C \cap F' = \{f_i\}$, then $|C \cap E'| \geq n-1 \geq 4$ since all except possibly one entry of the column f_i in $\boldsymbol{D}$ are nonzeros (Figure 7.1). One may dually argue in the case $C \cap E' = \{e_j\}$. So if $|C| \leq 4$, then $|C \cap E'| = |C \cap F'| = 2$. A simple check then shows that $C = \{e_i, f_i, e_j, f_j\}$ for some $i \neq j \in [1, n]$. ∎

Lemma 7.1.3. *Let S be a rank-n spike over a field $\mathbb{F}$ for $n \geq 3$, and let $E(S) = \{e_1, \dots, e_n,\ f_1, \dots, f_n\}$.*
(a) S is 3-connected.
(b) Let T be an arbitrary cubic tree with n leaves, and let T' be obtained from T by adding two new leaves l'_i, l''_i, $i \in [1, n]$ to each leaf l_i of T. Let $\tau'(e_i) = l'_i$ and $\tau'(f_i) = l''_i$ for $i \in [1, n]$. Then (T', τ') is a width-3 branch-decomposition of S.

Proof. (a) Let $X \subseteq E(S)$, $|X| \geq 2$, let $E' = \{e_1, \dots, e_n\}$ and $F' = \{f_1, \dots, f_n\}$, and let $\boldsymbol{D} = \boldsymbol{D}(x_1, \dots, x_n)$ be the reduced representation of S. We denote by $\boldsymbol{D}_X$ the submatrix of $\boldsymbol{D}$ determined by the rows in $X \cap E'$ and the columns in $F' - X$, and by $\boldsymbol{D}'_X$ the submatrix of $\boldsymbol{D}$ determined by the rows in $E' - X$ and the columns in $X \cap F'$. Then $\lambda_S(X) = r(\boldsymbol{D}_X) + r(\boldsymbol{D}'_X)$. If, say, $\boldsymbol{D}_X$ is an empty matrix, then $r(\boldsymbol{D}'_X) \geq 2$ since there are no parallel rows or columns in $\boldsymbol{D}$. Otherwise, $r(\boldsymbol{D}_X) + r(\boldsymbol{D}'_X) \geq 1 + 1 = 2$. Hence $\lambda_S(X) \geq 2$, and there are no 1- or 2-separations in S.

(b) Let $J \subseteq [1, n]$, and $X = \{e_j, f_j : j \in J\}$. Then both matrices $\boldsymbol{D}_X, \boldsymbol{D}'_X$ contain all 1's, and so $\lambda_S(X) = 2$. Therefore the width of every edge of T in the decomposition (T', τ') is 3, and the widths of the leaf edges are 2. ∎

Lemma 7.1.4. *Let $\boldsymbol{D}_k = \boldsymbol{D}(y_1, \dots, y_k)$. If $y_i = 1$ for more than one index $i \in [1, k]$, then the determinant $|\boldsymbol{D}_k| = 0$. If $y_i = 1$ for exactly one index $i \in [1, k]$, then $|\boldsymbol{D}_k| \neq 0$. Otherwise, if $y_i \neq 1$ for all $i \in [1, k]$, then*

$$|\boldsymbol{D}(y_1, \dots, y_k)| = \left(\prod_{i=1}^{k} (y_i - 1) \right) \cdot \left(1 + \sum_{i=1}^{k} \frac{1}{y_i - 1} \right).$$

Proof. We use simple matrix row operations:

$$|\boldsymbol{D}_k| = \begin{vmatrix} y_1 & 1 & \cdots & 1 \\ 1 & y_2 & \cdots & 1 \\ \vdots & & \ddots & \vdots \\ 1 & 1 & \cdots & y_n \end{vmatrix} = \begin{vmatrix} y_1 & 1 & \cdots & 1 \\ 1-y_1 & y_2 - 1 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 1-y_1 & 0 & \cdots & y_n - 1 \end{vmatrix}$$

The answer is clear if some $y_i = 1$. Otherwise, we expand the determinant

$$|\boldsymbol{D}_k| = (y_1 - 1) \left(\left(\frac{1}{y_1 - 1} + 1 \right) \prod_{i=2}^{k} (y_i - 1) + \sum_{i=2}^{k} \prod_{j=2,\, j \neq i}^{k} (y_j - 1) \right).$$ ∎

Corollary 7.1.5. *Let $S = S(x_1, \dots, x_n)$ be a rank-n spike over $\mathbb{F}$ for $n \geq 3$, and let $E(S) = \{e_1, \dots, e_n,\ f_1, \dots, f_n\}$ as above. Consider a set $X \subseteq E(S)$. If there are $i, i' \in [1, n]$ such that $e_i, f_i \in X$, $e_{i'}, f_{i'} \notin X$, and that $|\{e_j, f_j\} \cap X| = 1$ for $j \neq i, i'$, then X is a basis of S. If $|\{e_j, f_j\} \cap X| = 1$ for all $j \in [1, n]$, then X is a basis of S iff $|\boldsymbol{D}(x_j : j \in [1, n], f_j \in X)| \neq 0$. Otherwise, X is not a basis of S.*

∎

We say that a spike S on the ground set $\{e_1, \dots, e_n,\ f_1, \dots, f_n\}$ as above is a *free spike*, if every set $X \subseteq E(S)$ such that $|\{e_j, f_j\} \cap X| = 1$ for all $j \in [1, n]$ is a basis. Clearly, there is at most one free spike of each rank up to isomorphism.

7.2 The *Partition* Problem

Definition. The *Partition* problem is defined as follows:
Input: A multiset T of positive integers.

Question: Is there a partition (T_1, T_2) of T into two parts such that their sums are equal, i.e. $\sum_{x \in T_1} x = \sum_{y \in T_2} y$?

Definition. The #*Partition* problem is defined as follows:
Input: A multiset T of positive integers.

Output: The number of distinct partitions (T_1, T_2) of T into two parts such that their sums are equal, i.e. $\sum_{x \in T_1} x = \sum_{y \in T_2} y$.

The *Partition* problem is actually one of the six basic *NP*-complete problems mentioned in [17]. It is not difficult to see that the same reduction as in [17] works also for counting in #*Partition*. Hence we have:

Proposition 7.2.1. ([17]) *The Partition problem is NP-complete, and the #Partition problem is #P-complete.*

We want to extend the *Partition* problem to an arbitrary infinite field $\mathbb{F}$. If $\mathbb{F}$ has characteristic 0, then $\mathbb{F}$ contains the integers, and so the extension is trivial. So let us assume that $\mathbb{F}$ has characteristic $p > 0$. That is the place where we hit a surprising obstacle — the *Partition* problem itself is trivial for $p = 2$! Fortunately, the reduction for *Partition* in [17] shows an easy way around this obstacle. Another question arises about how to present the input numbers. Here we use a trick, based on the fact that infinite fields of characteristic $p > 0$ may be viewed as infinite-dimensional vector spaces over $GF(p)$, with the addition and scalar multiplication as in $GF(p)$. So we consider the following problem instead.

Definition. The p-VectorSum problem for a prime p is defined as follows: Let $F_0 \subset GF(p)^\omega$ be the set of all ω-dimensional vectors over $GF(p)$ having finitely many nonzero coordinates.
Input: A multiset $T \subseteq F_0$ of nonzero vectors, and a nonzero vector $\boldsymbol{t} \in F_0$.

Question: Is there a subset $T_1 \subseteq T$ such that $\sum_{\mathbf{a}\in T_1} \mathbf{a} = \boldsymbol{t}$?

The size of the input multiset T is $m \cdot n$ where $|T| = n$ and m is the length of the minimal prefix covering all nonzero coordinates of vectors in T. The #p-VectorSum problem is the natural counting counterpart of p-VectorSum.

Proposition 7.2.2. *The p-VectorSum problem is NP-complete for every prime number p, and the corresponding #p-VectorSum problem is #P-complete.*

Proof. We will use only vectors with coordinates in $\{-1, 0, 1\}$. Notice that $-1 = 1$ if $p = 2$. Let $\sum(X)$ shortly denotes $\sum_{x\in X} x$, and let us write c^k for k-times repetition of the coordinate c.

Claim 1. Let $T^o = \{(0^3, 1, 0^3), (0^4, 1, 0^2), (-1, 0^3, 1, 0^2), (0^5, 1, 0), (-1, 0^4, 1, 0), (0^6, 1), (-1, 0^5, 1)\}$, and let $T = \{(0, 1, -1, 0^4), (0, 1, 0, -1, 0^3), (0^2, 1, 0, -1, 0^2), (0^2, 1, 0^2, -1, 0), (0^3, 1, 0^2, -1), (1, 0^6)\} \cup T^o$ be sets of vectors. Let $\boldsymbol{u} = (0, 1, 0^5)$. There are exactly 7 sets $U_1, \ldots, U_7 \subseteq T$ such that $\sum(U_i) = \boldsymbol{u}$, $i \in [1, 7]$. Moreover, $U_i \cap T^o = \{\boldsymbol{u}_i\}$ for $i \in [1, 7]$ where $\{\boldsymbol{u}_i : i \in [1, 7]\} = T^o$.

Having Claim 1 at hand, the rest of the proof proceeds in the same way as the traditional reduction [17] of Partition to $3SAT$, except the last step. Let $\varphi = c_1 \wedge \ldots \wedge c_n$ be a SAT formula in conjunctive normal form such that each clause c_i contains exactly 3 literals, and let $x_1, \ldots, x_m$ be the variables involved in φ. We interpret satisfiability of φ via vectors from $GF(p)^{m+7n+3n}$. The coordinates of the vectors are divided into three sections; variable, clause, and incidence sections of lengths m, $7n$, and $3n$.

The incidence section of vector coordinates is indexed by the literals of all clauses of φ. For a variable x_i, $i \in [1, m]$, we construct two vectors $\mathbf{a}_i$, $\mathbf{a}'_i$, both having 1 at the position $i - 1$ of the variable section. Moreover, $\mathbf{a}_i$ has 1 at the positions of the incidence section indexed by literals which are x_i. Similarly, $\mathbf{a}'_i$ has 1 at the positions of the incidence section indexed by literals which are $\neg x_i$. All remaining coordinates are set to 0. The meaning of this piece of our construction is that we choose a $True$ or $False$ value for x_i by picking $\mathbf{a}_i$ or $\mathbf{a}'_i$, respectively, in the selected subset.

The chosen valuation of $x_1, \ldots, x_m$ satisfies φ if each clause is valued $True$, i.e. if each clause triple (of coordinates) in the incidence section has a nonzero coordinate. There are $2^3 - 1 = 7$ true valuations for each clause, and that is where we apply Claim 1. For a clause c_j, $j \in [1, n]$, we make a "shifted" copy T_j

of the set T from Claim 1 such that the coordinates of vectors from T now start at the position $7(j - 1)$ of the clause section. The seven vectors from $T^o_j \subset T_j$, moreover, have the coordinate triples for the clause c_j in the incidence section set to $(1, 1, 0)$, $(1, 0, 1)$, $(0, 1, 1)$, $(1, 0, 0)$, $(0, 1, 0)$, $(0, 0, 1)$, $(0, 0, 0)$, respectively. All remaining coordinates are set to 0.

Finally, we set $\mathbf{v} = (1^m, \overbrace{0, 1, 0^6, 1, \ldots, 0^6, 1, 0^5}^{7n}, 1^{3n})$, and $U = \bigcup_{j\in[1,n]} T_j \cup \bigcup_{i\in[1,m]} \{\mathbf{a}_i, \mathbf{a}'_i\}$. Assume that $V \subseteq U$ is such that $\sum(V) = \mathbf{v}$. Then the variable section of the coordinates guarantees that exactly one of $\mathbf{a}_i, \mathbf{a}'_i$ for $i \in [1, m]$ belongs to V, which determines the value for x_i. The clause section guarantees that exactly one vector from each set T^o_j is selected in V for $j \in [1, n]$. And since the incidence section for the clause c_j has each coordinate 1, which cannot be obtained just using a vector from T^o_j, the clause c_j, $j \in [1, n]$ must be true in the selected valuation. The converse direction proceeds similarly. Therefore, the solutions to "$V \subseteq U$ such that $\sum(V) = \mathbf{v}$" are in a one-to-one correspondence with satisfying assignments to the formula φ.

The size of W is clearly polynomial in the size of φ, and we formally pad all other coordinates of the vectors with 0's. So the p-VectorSum problem for the input $U, \mathbf{v}$ is NP-complete. Moreover, the one-to-one correspondence between solutions to $U, \mathbf{v}$ and satisfying assignments to φ proves also that the #p-VectorSum problem is #P-complete. ∎

7.3 Recognizing Free Spikes is Hard

Recall the notation from Section 7.1. We reduce the problem to recognize that the given rank-n spike S over $\mathbb{Q}$ is *not* the free spike to finding a solution to the Partition problem.

Lemma 7.3.1. *Let $\mathbb{F}$ be an arbitrary field, $n \geq 3$, and $z_1, \ldots, z_n \in \mathbb{F} - \{0\}$.*
(a) The matrix $[\boldsymbol{I}_n \,|\, \boldsymbol{D}(\frac{1}{z_1}+1, \frac{1}{z_2}+1, \ldots, \frac{1}{z_n}+1)]$ represents the free spike if and only if there is no subset $Z_1 \subset \{z_1, \ldots, z_n\}$ such that $\sum_{y\in Z_1} y = -1$.
(b) Let b be the number of bases of the spike $S = S(\frac{1}{z_1}+1, \ldots, \frac{1}{z_n}+1)$, and let c be the number of subsets $Z_1 \subset \{z_1, \ldots, z_n\}$ such that $\sum_{y\in Z_1} y = -1$. Then $b = n(n-1)2^{n-2} + 2^n - c$.

Proof. Notice that $z_i = \frac{1}{y_i - 1}$ for $y_i = \frac{1}{z_1} + 1$. So the part (a) directly follows from Lemma 7.1.4 and Corollary 7.1.5. Again by Corollary 7.1.5, there are exactly $n(n-1)2^{n-2}$ bases of S which contain two elements of some leg of S. The free spike S has exactly 2^n bases which intersect each leg in one element, and there are no other bases than described above. If S is not the free spike, then each solution to $\sum_{y\in Z_1} y = -1$ reduces the number of bases by 1 by Lemma 7.1.4. Hence (b) follows. ∎

Theorem 7.3.2. *Let $n \geq 3$, and S be the rank-n matroid represented by the matrix $[\boldsymbol{I}_n \,|\, \boldsymbol{D}(x_1, \dots, x_n)]$ over the rational numbers $\mathbb{Q}$ where $x_1, \dots, x_n \in \mathbb{Q} - \{1\}$. Then branch-width of S is 3, and it is NP-hard to recognize that S is not the rank-n free spike.*

Proof. Let $T = \{t_1, t_2, \dots, t_n\}$ be a multiset of positive integers considered as an input to the PARTITION problem. Let $t = t_1 + t_2 + \ldots + t_n$, and let us denote by $z_i = \frac{-2t_i}{t}$ and $x_i = \frac{1}{z_i} + 1$ for $i \in [1, n]$. Assume that $I \subset [1, n]$ is such that the multiset partition $(\{t_i : i \in I\}, \{t_i : i \in [1, n] - I\})$ is a solution to the PARTITION problem. That is equivalent to $\sum_{i \in I} z_i = -1$, and hence we are done by Lemma 7.3.1(a) and by Proposition 7.2.1. ∎

We are going to extend the previous result to an arbitrary infinite field $\mathbb{F}$ of characteristic $p > 0$. As in Section 7.2, we consider a countable vector subspace $GF(p)^\omega$ of $\mathbb{F}$, and "numbers" from the set $F_0 \subset GF(p)^\omega$ of all the vectors having finitely many nonzero coordinates. Notice that the zero vector in F_0 coincides with $0 \in \mathbb{F}$. We create a set F_1 of all finite-length symbolic expressions formed over F_0 using valid operations of the field $\mathbb{F}$. Formally, F_1 is the language over the alphabet $F_0 \cup \{0, 1, -, ^{-1}, +, \cdot, (,)\}$ defined recursively as follows.

- $F_0 \subset F_1$ and $0, 1 \in F_1$.
- If $a, b \in F_1$, then $-(a) \in F_1$, $(a) + (b) \in F_1$, and $(a) \cdot (b) \in F_1$.
- If $a \in F_1$ such that $\mathrm{eval}(a) \neq 0$, then $(a)^{-1} \in F_1$.

The mapping $\mathrm{eval} : F_1 \to \mathbb{F}$ is the homomorphism defined by $\mathbb{F}$-evaluations of the symbolic expressions. Notice that each expression in F_1 is a finite word.

The definition of $\mathbb{F}_1$ may seem unnatural at the first look, but it actually is analogous to standard handling of rational numbers — algorithms computing with rational numbers usually input and work with symbolic fractions $x = \frac{a}{b}$, and not with the decimal expansion of x. Thus instead of "arithmetic in $\mathbb{F}$", we actually consider arithmetic of the symbolic expressions in $\mathbb{F}_1$, and we allow symbolic input from $\mathbb{F}_1$. The understanding behind this convention is that, for any particular infinite field $\mathbb{F}$, an oracle is given for computing in $\mathbb{F}$, and then the symbolic expressions my be easily parsed in time polynomial in their size. However, one should keep in mind that there are no means to decide the $\mathbb{F}$-value of a symbolic expression without such an oracle.

Theorem 7.3.3. *Let $n \geq 3$, and let $\mathbb{F}$ be an infinite field. If S is the rank-n matroid represented by the matrix $[\boldsymbol{I}_n \,|\, \boldsymbol{D}(x_1, \dots, x_n)]$ over $\mathbb{F}$ where $x_1, \dots, x_n \in \mathbb{F} - \{1\}$, then it is NP-hard to recognize that S is not the rank-n free spike.*

Proof. If $\mathbb{F}$ has characteristic 0, then $\mathbb{F}$ contains the rational numbers as a subfield, and hence the statement is proved in Theorem 7.3.2. Otherwise, for characteristic $p > 0$, we use a reduction to the p-VECTORSUM problem.

Let $F_0 \subset GF(p)^\omega$ be the set of all ω-dimensional vectors over $GF(p)$ having finitely many nonzero coordinates. Suppose that $T = \{\boldsymbol{t}_1, \boldsymbol{t}_2, \dots, \boldsymbol{t}_n\} \subset F_0 - \{\boldsymbol{0}\}$ is a multiset of nonzero vectors, and that $\boldsymbol{t} \in F_0 - \{\boldsymbol{0}\}$. We denote by $z_i = -\boldsymbol{t}_i \cdot \boldsymbol{t}^{-1} \in F_1$ and $x_i = z_i^{-1} + 1 \in F_1$ for $i \in [1, n]$ as symbolic expressions. Then, by Lemma 7.3.1(a), the matrix $[\boldsymbol{I}_n \,|\, \boldsymbol{D}(x_1, \dots, x_n)]$ does not represent the rank-n free spike if and only if there is a subset $J \subseteq [1, n]$ such that $\sum_{j \in J} z_j = -1$ in $\mathbb{F}$, so if $\sum_{j \in J} \boldsymbol{t}_j = \boldsymbol{t}$ in $GF(p)^\omega$.

Since we have only used symbolic operations in $\mathbb{F}_1$, the reduction to a p-VECTORSUM instance $T, \boldsymbol{t}$ is finished in polynomial time. The reduction itself does not depend on $\mathbb{F}$ except on the characteristic p. So we are done by Proposition 7.2.2. ∎

In contrast to Theorem 6.5.2, previous Theorem 7.3.3 together with the next simple lemma imply that MS_M sentences are likely not tractable for branch-width-3 matroids over infinite fields. The same obviously applies also to abstract matroids. Thus we have, in some sense, even stronger reason than given in Proposition 6.5.4, why the results of the previous chapters are formulated only for matroids represented over finite fields.

Lemma 7.3.4. *There is a sentence ψ in the second-order monadic logic MS_M of matroids; such that ψ is true for a rank-n spike S, $n \geq 5$, if and only if S is the free spike.*

Proof. First, using Lemma 7.1.2, we identify the legs of the spike S by the predicate $leg(e, f) \equiv \forall x\, \exists y \neq x (x = e \vee x = f \vee circuit(\{e, f, x, y\}))$, where $circuit(C) \equiv (\neg\, \mathrm{indep}(C) \wedge \forall e (e \notin C \vee \mathrm{indep}(C - \{e\})))$. Then S is the free spike if and only if every set intersecting each leg in one element is a basis. Hence we write $\psi = free\_spike \equiv \forall F\, [\mathrm{indep}(F) \vee \exists e, f\, (leg(e, f) \wedge (e, f \in F \vee e, f \notin F))]$. We leave remaining technical details of the predicates to the reader. ∎

Corollary 7.3.5. *Let $n \geq 5$, and let $\mathbb{F}$ be an infinite field. Let $\psi = free\_spike$ be the sentence in second-order monadic logic MS_M of matroids defined above. If S is a rank-n matroid represented by the matrix $[\boldsymbol{I}_n \,|\, \boldsymbol{D}(x_1, \dots, x_n)]$ over $\mathbb{F}$ where $x_1, \dots, x_n \in \mathbb{F} - \{1\}$, then it is NP-hard to decide whether ψ is true for S.* ∎

7.4 Spikes and the Tutte Polynomial

Lastly we show that there is likely no nontrivial extension of Theorem 5.3.1 (computing of the Tutte polynomial for a represented matroid of bounded branch-width) possible in the case of an infinite field $\mathbb{F}$. Recall from Lemma 7.1.3 that any spike has branch-width at most 3, and that a width-3 branch-decomposition of a spike can be easily found.

Theorem 7.4.1. *Let $n \geq 3$, and S be the rank-n matroid represented by the matrix $[\boldsymbol{I}_n \,|\, \boldsymbol{D}(x_1, \ldots, x_n)]$ over the rational numbers $\mathbb{Q}$ where $x_1, \ldots, x_n \in \mathbb{Q} - \{1\}$. Then branch-width of S is 3, and it is #P-hard to compute the Tutte polynomial $T(S; x, y)$ for S.*

Proof. We repeat the proof of Theorem 7.3.2, and apply Lemma 7.3.1(b). Thus counting all bases of S is a #P-hard problem. The last step is in realizing that the evaluation $T(S; 1, 1)$ equals the number of bases of S. ∎

In a way analogous to Theorem 7.3.3, we simply extend the previous statement for an arbitrary infinite field $\mathbb{F}$.

Theorem 7.4.2. *Let $n \geq 3$, and let $\mathbb{F}$ be an infinite field. If S is the rank-n matroid represented by the matrix $[\boldsymbol{I}_n \,|\, \boldsymbol{D}(x_1, \ldots, x_n)]$ over $\mathbb{F}$ where $x_1, \ldots, x_n \in \mathbb{F} - \{1\}$, then it is #P-hard to compute the Tutte polynomial $T(S; x, y)$ for S.* ∎

Chapter 8

Excluded Minors for Branch-Width 3

The central interest of our work is in matroid branch-width. Structures of matroids of branch-width one and two are trivial. So in this chapter we closely look at structural properties of branch-width three. We review some known results implying that the classes of matroids of branch-width at most k have finite number of excluded minors. Knowing these excluded minors would provide a useful characterization of branch-width.

We prove a conjecture formulated by Dharmatilake that there are exactly 10 such excluded minors for branch-width three among binary matroids (Theorem 8.2.1). Moreover, we present some results about the ternary and quaternary excluded minors. The arguments in our paper are assisted by the computer program MACEK [30], which was developed by the author for efficient general structural computations with represented matroids.

Besides the main result, we describe a simple polynomial algorithm for testing branch-width three in an abstract matroid. The content of this chapter is taken from our original paper [23].

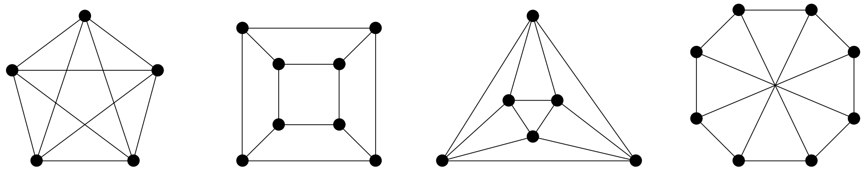
8.1 Properties of Branch-Width 3

Recall that $\mathcal{B}_k$ denotes the class of matroids of branch-width at most k. The next elementary properties of branch-width are well-known [22].

Lemma 8.1.1. *For any fixed $k \geq 1$, the class $\mathcal{B}_k$ is closed under minors, duality, direct sums, and 2-sums.*

Clearly, $\mathcal{B}_1$ consists of matroids with no dependencies, and so the only excluded minor for $\mathcal{B}_1$ is a loop. It was shown in [41] that the class $\mathcal{B}_2$ coincides with the class of direct sums of series-parallel networks. Hence there are two excluded minors for $\mathcal{B}_2$, namely the uniform matroid $U_{2,4}$ and the graphic matroid $M(K_4)$. The smallest matroids not in $\mathcal{B}_3$ are the uniform matroids $U_{3,7}, U_{4,7}$.

By Robertson-Seymour Theorem [42], there is a finite set of excluded minors for graphs of branch-width at most k for all k, but those sets are not known for $k > 3$. (After all, there are only few natural minor-closed properties of graphs for which the set of excluded minors is known.) The excluded minors for graphs of branch-width at most 3 were found by Dharmatilake and others in [13]. The same list was independently found later in [7]. See the graphs in Figure 8.1.



K_5 $\quad Q_3$ $\quad O_6$ $\quad V_8$

Figure 8.1: The four excluded minors for graphs of branch-width at most 3.

$$R_{10} \quad \left[\begin{array}{c|ccccc} & -1 & 1 & 0 & 0 & 1 \\ & 1 & -1 & 1 & 0 & 0 \\ \boldsymbol{I} & 0 & 1 & -1 & 1 & 0 \\ & 0 & 0 & 1 & -1 & 1 \\ & 1 & 0 & 0 & 1 & -1 \end{array}\right]$$

Figure 8.2: The matroid R_{10} by Bixby, in a totally unimodular representation.

Theorem 8.1.2. (Dharmatilake, Chopra, Johnson, Robertson) *A graph has branch-width at most* 3 *if and only if it has no minor isomorphic to one of the graphs* $\{K_5, Q_3, O_6, V_8\}$.

Since the matroids of all these four graphs have branch-width greater than 3, the theorem gives the graphic (and cographic as duals) excluded minors for the class B_3. It is easy to find out that the regular matroid R_{10} (Figure 8.2) is also an excluded minor for B_3. In addition to Theorem 8.1.2, Dharmatilake used a specialized computer program to search for small binary matroids (up to 12 elements) that are excluded minors for $\mathcal{B}_3$. He found three more non-regular matroids denoted by N_{11}, N_{23}, N_{11}^* (Figure 8.3). Let

$$\mathcal{F}_2 = \{M(K_5), M(K_5)^*, M(Q_3), M(O_6), M(V_8), M(V_8)^*, R_{10}, N_{11}, N_{11}^*, N_{23}\}.$$

Notice that $M(Q_3), M(O_6)$ are dual to each other, and that R_{10} and N_{23} are both self-dual. Dharmatilake then conjectured [13]:

Conjecture 8.1.3. (Dharmatilake, 1994) A binary matroid has branch-width at most 3 if and only if it has no minor isomorphic to one of the members of $\mathcal{F}_2$.

We prove this conjecture next in Sections 8.2,8.4. In addition, we present some results about ternary and quaternary excluded minors for the class $\mathcal{B}_3$. We use the following theorem [22, 21] in our proof.

$$N_{11} \quad \left[\begin{array}{c|cccccc} & 1 & 0 & 0 & 1 & 1 & 1 \\ & 0 & 1 & 0 & 1 & 1 & 0 \\ \boldsymbol{I} & 0 & 0 & 1 & 1 & 0 & 1 \\ & 1 & 1 & 1 & 1 & 0 & 0 \end{array}\right] \qquad N_{23} \quad \left[\begin{array}{c|ccccc} & 1 & 1 & 0 & 1 & 1 \\ & 1 & 0 & 1 & 1 & 0 \\ \boldsymbol{I} & 0 & 1 & 1 & 1 & 0 \\ & 1 & 1 & 0 & 0 & 1 \\ & 1 & 0 & 1 & 0 & 1 \end{array}\right]$$

Figure 8.3: The matroids N_{11}, N_{23} by Dharmatilake, in binary representations.

Theorem 8.1.4. (Hall, Oxley, Semple, Whittle) *If N is an excluded minor for the class $\mathcal{B}_3$, then N has at most* 14 *elements.*

In fact, another recent paper [18] gives a surprisingly short proof that there are finitely many excluded minors for the class $\mathcal{B}_k$ for every k.

Theorem 8.1.5. (Geelen, Gerards, Robertson, Whittle) *If N is an excluded minor for the class $\mathcal{B}_k$, then N has at most $(6^{k+1} - 1)/5$ elements.*

8.2 Finding the Excluded Minors

Here we state the major result of our paper — a proof of Conjecture 8.1.3.

Theorem 8.2.1. *A binary matroid has branch-width at most* 3 *if and only if it has no minor isomorphic to one of the members of $\mathcal{F}_2$ (see on page 67).*

Before proving the theorem itself, we present three short lemmas. The first lemma is proved in [22, Lemma 7.4].

Lemma 8.2.2. *Every excluded minor N for the class $\mathcal{B}_3$ is* 3*-connected, and the only* 3*-separations in N have one side of size at most* 4.

Let R_{12} be the regular matroid from Figure 8.4.

Lemma 8.2.3. *No regular excluded minor for the class $\mathcal{B}_3$ has an R_{12}-minor.*

Proof. Let M be a regular matroid with an R_{12}-minor. A supplementary result of Seymour's decomposition theorem for regular matroids [47] states that a regular matroid with an R_{12}-minor has an exact 3-separation in which both sides have at least 6 elements. However, then M cannot be an excluded minor for $\mathcal{B}_3$ by Lemma 8.2.2. ∎

We denote by F_7 the Fano matroid, see Figure 2.2 on page 12.

$$R_{12} \quad \left[\begin{array}{c|cccccc} & 1 & 1 & 1 & 0 & 0 & 0 \\ & 1 & 1 & 0 & 1 & 0 & 0 \\ & 1 & 0 & 0 & 0 & 1 & 0 \\ \boldsymbol{I} & 0 & 1 & 0 & 0 & 0 & 1 \\ & 0 & 0 & 1 & 0 & -1 & -1 \\ & 0 & 0 & 0 & 1 & -1 & -1 \end{array}\right]$$

Figure 8.4: The matroid R_{12} by Seymour, in a regular representation.

Lemma 8.2.4. *Let M be a 3-connected binary matroid on at most 14 elements with an F_7-minor. If M has branch-width 4, then M has an N-minor for $N \in \mathcal{F}_2$.*

Proof. Verifying this lemma is clearly a matter of a finite case check. We have done the case analysis with help of the computer program Macek [30]. Details of this computation are presented in Section 8.4. ∎

Proof of Theorem 8.2.1. Recall the set $\mathcal{F}_2 = \{M(K_5), M(K_5)^*, M(Q_3), M(O_6), M(V_8), M(V_8)^*, R_{10}, N_{11}, N_{11}^*, N_{23}\}$ that is closed under duality. Let N be a binary excluded minor for the class $\mathcal{B}_3$ of matroids of branch-width at most 3. Then N is 3-connected by Lemma 8.2.2.

We first consider the case that N is a regular matroid. Then, by Seymour's decomposition theorem for regular matroids [47] (also [38, Section 13.2]), one of the following is true: N is graphic, or N is cographic, or N has an R_{10}- or R_{12}-minor. The last case is not possible here due to Lemma 8.2.3. If N has an R_{10}-minor, then $N \simeq R_{10} \in \mathcal{F}_2$. It remains to consider, up to duality, the case that N is a graphic matroid. Then, using Theorem 8.1.2, N is isomorphic to one of the graphic members of $\mathcal{F}_2$.

Otherwise, N is not a regular matroid. So by Tutte's characterization of regular matroids [48] (also [38, Section 13.1]), binary N must have an F_7- or F_7^*-minor. Then, up to duality, the proof is finished in Lemma 8.2.4 and Theorem 8.1.4. ∎

Remark. It would be possible to do a computer search similar to Lemma 8.2.4 also for regular matroids, thus avoiding use of the previous theoretical results about regular matroids.

It is natural to ask about other, non-binary excluded minors for $\mathcal{B}_3$. The paper [22] remarks that it is *"certainly feasible to write a computer program that would quickly find all excluded minors that are representable over a given field"*. We think that, while making this remark, the authors did not fully understand

the effects of so called "exponential combinatorial explosion". Similarly as in Lemma 8.2.4, we have done the computer search for ternary excluded minors on up to 12 elements, and for quaternary ones on up to 10 elements. More details can be found in Section 8.4.

Proposition 8.2.5. *Let $\mathcal{F}_3$ be the set of (pairwise non-isomorphic) excluded minors for $\mathcal{B}_3$ that are ternary but not binary. Then $\mathcal{F}_3$ contains no matroids on less than 9 elements, 18 matroids on 9 elements, 31 matroids on 10 elements, and no matroid on 11 or 12 elements.*

Proposition 8.2.6. *Let $\mathcal{F}_4$ be the set of (pairwise non-isomorphic) excluded minors for $\mathcal{B}_3$ that are quaternary but neither ternary nor binary. Then $\mathcal{F}_4$ contains no matroids on less than 8 elements, 5 matroids on 8 elements, 90 matroids on 9 elements, and 32 matroids on 10 elements.*

Remark. The computer searches in the previous two propositions are not complete since they reached only up to 12 and 10, respectively, elements instead of 14. Unfortunately, the numbers of matroids in the ternary and quaternary searches grew enormously. (For example, there were more than 16000 quaternary matroid representations searched on 10 elements, compared to about 2400 binary ones searched on 14 elements. The total number of quaternary matroids on 10 elements is even larger.) We estimated that finishing the easier search in ternary matroids would take at least several months on a single home computer, which is not worth the effort, we think.

8.3 Testing Branch-width Three

In this section we present a small detour dedicated to testing branch-width three in 3-connected matroids. We do so because we need a simple and fast practical algorithm for this problem in our computer analysis.

A nice elementary linear time algorithm for finding graphs of branch-width three is given by Bodlaender and Thilikos in [7]. Unlike other linear time algorithms known for testing, for example, bounded tree-width of graphs, this algorithm has reasonably small constant and it is suitable for practical implementation. It is probably possible to generalize the method of this algorithm to matroids of branch-width three, but a major problem would be that the excluded minors for $\mathcal{B}_3$ were needed in the algorithm. Hence, moreover, a possible generalization of the above mentioned algorithm to matroids would also be much more complicated and not so practical to implement.

In contrast to the above approach, we use a polynomial algorithm with higher exponent, but which is very simple to implement and fast in practical computations. The algorithm is based on the next interesting result of [22, Theorem 4.1]. We also acknowledge an informal suggestion from Geoff Whittle about a possibility to develop such an algorithm.

A *partitioned matroid* is a pair (M,P) where M is a matroid on the ground set E, and $P=(E_1,\dots,E_p)$ is a partition (into nonempty parts) of the set E. If the context is clear, we briefly refer to the partitioned matroid as to M. We generalize the connectivity function to subsets of P by $\lambda_M(Q)=\lambda_M(\bigcup_{X\in Q}X)$. A partition $(Q,P-Q)$ of P is a k-separation if $\lambda_M(Q)\le k$ and $k\le|\bigcup_{X\in Q}X|\le|E|-k$. A partitioned matroid (M,P) is n-connected if it has no k-separation for $k=1,\dots,n-1$.

We define a *partitioned branch-decomposition* (T,τ) of (M,P) analogously to a normal branch-decomposition, but using a bijection $\tau:P\to\ell(T)$ (i.e. the leaves are labeled by the sets of P). A subset $Q\subseteq P$ is *displayed* by an edge e in T if $Q=\tau^{-1}(\ell(T'))$ where T' is a component of $T-e$.

Theorem 8.3.1. (Hall, Oxley, Semple, Whittle) *Let (M,P) be a 3-connected partitioned matroid of branch-width 3, and let $Q\subset P$ be a 3-separating set that is not displayed in any width-3 branch-decomposition of M. Then, for $R=Q$ or $R=P-Q$, the following holds: $|R|\in\{2,3\}$, and $|X|=1$ for all $X\in R$.*

To turn this theorem into a simple greedy algorithm for finding a width-3 branch-decomposition, we need one more technical lemma. A subset $F\subseteq P$ is 3-*branched* in a partitioned matroid (M,P) if the partitioned matroid (M,P_F) has branch-with 3, where $P_F=\{E(M)-\bigcup_{X\in F}X\}\cup F$.

Lemma 8.3.2. *Let (M,P) be a 3-connected partitioned matroid of branch-width 3 with $|P|\ge 3$ and $|E(M)|\ge 6$. Then there is a 3-separating subset $Q\subseteq P$ in M such that Q is 3-branched, and that*

1. *$4\le|Q|\le 6$ and $|X|=1$ for all $X\in Q$, or*
2. *$2\le|Q|\le 4$ and $|X|>1$ for precisely one $X\in Q$, or*
3. *$|Q|=2$ and $|X|>1$ for both $X\in Q$.*

Proof. Let (T,τ) be a width-3 branch-decomposition of the partitioned matroid (M,P). We say that a leaf l of T is *single* if $|\tau^{-1}(l)|=1$. A subtree T_1 of T is *nice* if T_1 contains at least 4 leaves from $\ell(T)$, or if T_1 contains at least 2 leaves from $\ell(T)$ and some of them is not single. Clearly, a nice subtree can be obtained from T by deleting any leaf which is single if possible. Let T_0 be a nice subtree of T that has the smallest size. We prove that the τ-preimages of the leaves of T_0 define our set Q.

Notice that T_0 is a proper subgraph of the cubic tree T on more than 2 vertices, and so T_0 has a vertex v of degree 2. Let T_1,T_2 be the two components

of T_0-v. We proceed by a contradiction to (1),(2),(3), respectively: If T_0 has at least 7 leaves in $\ell(T)$, then one of T_1,T_2 is nice, and smaller than T_0. If T_0 has at least 5 leaves in $\ell(T)$ and not all of them are single, then one of T_1,T_2 is nice. Finally, if T_0 has at least 3 leaves in $\ell(T)$ and at least 2 of them are not single, then one of T_1,T_2 is nice again. ∎

Notice that any set Q from the lemma can be displayed in some width-3 branch-decomposition of (M,P) by Theorem 8.3.1. The greedy algorithm for finding a width-3 branch-decomposition of a given 3-connected matroid M is now pretty obvious: We initially set the partition $P=(\{x\}:x\in E(M))$. Then we find a set Q according to Lemma 8.3.2, replace the sets of Q in P by a single part, and repeat the whole process again. When we get to a partition P with only two parts, we have found a width-3 branch-decomposition of M. If we fail to find Q at any step, then the branch-width of M must be bigger than 3 by Theorem 8.3.1.

We present a formal description of the algorithm implementation in Figure 8.5. Let $n=|E(M)|$. To make the implementation faster, at each program pass we precompute the triangles and triads formed by the remaining singleton elements in S, and store them in the set T. Actually, we better use an (internal) linear order on S to prevent repetitions of the same triples in T. Then, unless a 4-element line or coline is found, there are only $O(n^2)$ triples in T. Hence the next search over all pairs in $U\cup V$ takes only at most $O(n^4)$ iterations. Notice that the set Q, which we possibly find there, is already branched into two branches given by the sets X,Y. In total the algorithm needs $O(n^5)$ rank evaluations in M to finish. (We cannot tell the absolute computation time since the length of one rank evaluation depends on the given representation of M.)

8.4 Computing Details

This section provides a detailed description of the computation we use in the proof of Lemma 8.2.4. The related computer files and intermediate results of the computation can be found in [31].

As already noted above, we have used the computer program MACEK [30], which was written by the author for general structural computations with represented matroids. This program can input and output matrices over different fields (and so called partial fields), perform usual elementary matrix operations, look for matroid minors and equivalence (subject to a particular matrix representation), test some matroid-structural properties like branch-width 3, and generate non-equivalent 3-connected matrix extensions.

```
# Finding a width-3 branch-decomposition of a 3-connected matroid M.
begin
   input  3-connected matroid M on the ground set E, |E| ≥ 6
   # Variable S keeps the remaining singletons of E, and P the constructed partition.
   set  P = ∅,  S = E
   while  |P| + |S| > 2  do
      # Variable T collects triangles and triads, and Q gets the new part for P.
      set  Q = T = ∅
      for all pairs {x, y} ⊆ S do
         set  K = cl_M({x, y}) ∩ S,  L = cl*_M({x, y}) ∩ S
         for X=K,L do if |X| = 3 then set  T = T ∪ {X}
         for X=K,L do if |X| > 3 then set  Q = X;  break
      done
      if Q ≠ ∅ then
         # A 4-element line or coline Q is good for Theorem 8.3.1.
         set  Q = any 4-element subset (or whole) of Q
      else
         # Searching through all remaining possibilities for Q (cf. Lemma 8.3.2).
         set  U = all disjoint pairs from (P ∪ T ∪ {{x, y} : x, y ∈ S})
         set  V = {{{x}, Y} : x ∈ S, Y ∈ P}
         for {X, Y} ∈ U ∪ V do
            # If X ∪ Y is 3-separating, then it is clearly also branched into X, Y.
            if λ_M(X ∪ Y) ≤ 3 then set  Q = X ∪ Y ;  break
         done
      fi
      if Q = ∅ then break
      # When Q was found above, update the partition P and the singletons S.
      set  S = S − Q,  P = {Q} ∪ {X : X ∈ P ∧ X ∩ Q = ∅}
      exec  remember the (sub)branching of Q in P for output
   done
   if |P| + |S| > 2  then output "No width-3 branch-decomposition exists."
   else output "A width-3 branch-decomposition found here: ..."
end.
```

Figure 8.5:

Remark. We want to emphasize that the MACEK program we use is a general toolkit for matroid computations, and not a specialized closed program prepared only for one task. Also, the reader may get feeling that our main advantage over Dharmatilake's work [13] is in today's faster computers. This is, however, not so since Dharmatilake used the resources of the Ohio Supercomputer Center for his search, while our computation was carried out on a usual cheap home computer (with AMD 800MHz processor).

Let us look at a 3-connected matroid M with an F_7-minor. By Seymour's splitter theorem, there is a sequence of 3-connected matroids $N_0 = F_7, N_1, \ldots, N_t = M$ such that N_{i+1} is a single-element extension or coextension of N_i. Since binary matroids have unique representations up to matrix equivalence, we actually consider a one-row or one-column extension of the matrix of N_i in the search. At the t-th step of our computation, for $t = 1, \ldots, 7$, we generate all one-row or one-column extensions on $7 + t$ elements from the branch-width-3 extensions on $6 + t$ elements. Then we put aside the new extensions having branch-width bigger than 3, and verify that all of them contain a minor in the set $\mathcal{F}_2$. We continue the next step with the remaining (branch-width-3) extensions.

For reader's information, we have generated the following numbers of branch-width-3 extensions (the total number of all extensions is higher) at each step: 2 extensions on 8 elements, 4 on 9 elements, 14 on 10 elements, 38 on 11 elements, 125 on 12 elements, 432 on 13 elements, and 1551 on 14 elements. We carried out the master computation on a PC computer with AMD Duron 800MHz processor and 256MB memory, running Linux kernel 2.4.8, glibc 2.2.2, and using compiler gcc 2.96. The whole computation took about 1 day with no self-checks. We also verified the results on various university computers with Sparc–Solaris, Alpha–OSF, or Intel–NetBSD, using other versions of gcc 2.7, 2.8 or 2.95.

Notice that, if we had not known the set $\mathcal{F}_2$, our approach would automatically provide the new excluded minors at each step.

Let us now move to Proposition 8.2.5. All non-binary matroids contain a $U_{2,4}$-minor. Unfortunately, $U_{2,4}$ (the 2-whirl) is one of the exceptions in Seymour's splitter theorem, but an enhancement of this theorem presented in [38, Section 11.3] implies that all ternary extensions of $U_{2,4}$ that are not whirls contain a single-element extension or coextension of $\mathcal{W}^3$ (the 3-whirl) as a minor. All whirls have branch-width at most 3. Therefore we perform a similar computation as above starting from the self-dual matroid $\mathcal{W}^3$. We must not forget to exclude those matroids having some ternary (i.e. regular) member of $\mathcal{F}_2$ as a minor. For details of the computation we refer to [31].

To present a similar computational proof for Proposition 8.2.6, we use the following result of [45]: A non-binary non-ternary matroid representable over some field has a $U_{2,5}$- or $U_{3,5}$-minor. Clearly, such a matroid must contain a $U_{2,5}$-minor, unless it is isomorphic to $U_{n-2,n}$ which has branch-width 3. The computation is then analogous to the previous two. Again, we should exclude those matroids having some member of $\mathcal{F}_2$ or some quaternary member of $\mathcal{F}_3$ as a minor. (In particular, we must do this computation after the previous one.) One new theoretical problem arises for quaternary matroids — one matroid may have non-equivalent quaternary representations. Fortunately, in this case there may be only at most two non-equivalent representations of the same matroid obtained by the automorphism $\omega \to \omega^2$ of the field $GF(4)$, and such pairs can be easily detected in the resulting list. Again, we refer the reader to [31] for details.

Remark. The numbers of matroids and excluded minors for $\mathcal{B}_3$ generated in Propositions 8.2.5,8.2.6 suggest that it is likely infeasible to search all (including non-representable) matroids on up to 14 elements for the excluded minors for $\mathcal{B}_3$. However, it looks like there are only few excluded minors on more than 10 elements, and hence it may be possible to strengthen Theorem 8.1.4 so that it would suffice to exhaustively search only matroids up to 10 elements.

Bibliography

[1] K.A. Abrahamson, M.R. Fellows, *Finite Automata, Bounded Treewidth, and Well-Quasiordering*, In: Graph Structure Theory, Contemporary Mathematics 147, American Mathematical Society (1993), 539–564.

[2] A. Andrzejak, *An Algorithm for the Tutte Polynomials of Graphs of Bounded Treewidth*, Discrete Math. 190 (1998), 39–54.

[3] S. Arnborg, J. Lagergren, D. Seese, *Problems easy for Tree-decomposible Graphs (extended abstract)*, Proc. 15th Colloq. Automata, Languages and Programming, Lecture Notes in Computer Science 317, Springer-Verlag (1988), 38–51.

[4] S. Arnborg, D.G. Corneil, A. Proskurowski, *Complexity of Finding Embeddings in a k-tree*, SIAM J. Alg. Disc. Meth. 8 (1987), 277–284.

[5] H.L. Bodlaender, *A Tourist Guide through Treewidth*, Acta Cybernetica 11 (1993), 1–21.

[6] H.L. Bodlaender, *A Linear Time Algorithm for Finding Tree-Decompositions of Small Treewidth*, SIAM J. Computing 25 (1996), 1305–1317.

[7] H. Bodlaender, D.M. Thilikos, *Graphs with Branch-Width at most Three*, Technical Report UU-CS-1997-37, Department of Computer Science, Utrecht University, Utrecht, the Netherlands, 1997.

[8] H.L. Bodlaender, D.M. Thilikos, *Constructive Linear Time Algorithms for Branchwidth*, Proceedings 24th ICALP, Lecture Notes in Computer Science 1256, 1997, 627–637.

[9] R.B. Borie, R.G. Parker, C.A. Tovey, *Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families*, Algorithmica 7 (1992), 555–582.

[10] B. Courcelle, *Recognizability and Second-Order Definability for Sets of Finite Graphs*, technical report I-8634, Universite de Bordeaux, 1987.

[11] B. Courcelle, *Graph Rewriting: an Algebraic and Logic Approach*, In: Handbook of Theoretical Computer Science Vol. B, Chap. 5, North-Holland 1990.

[12] B. Courcelle, *The Monadic Second-Order Logic of Graphs I. Recognizable sets of Finite Graphs* Information and Computation 85 (1990), 12–75.

[13] J.S. Dharmatilake, *Binary Matroids of Branch-Width* 3, PhD. dissertation, Ohio State University, 1994.

[14] R. Diestel, Graph theory, Graduate Texts in Mathematics 173, Springer-Verlag, New York 1997, 2000.

[15] G. Ding, B. Oporowski, J. Oxley, D. Vertigan, *Unavoidable Minors of Large* 3-*Connected Matroids*, J. Combin. Theory Ser. B 71 (1997), 244–293.

[16] R.G. Downey, M.R. Fellows, Parametrized Complexity, Springer-Verlag New York, 1999, ISBN 0-387-94833-X.

[17] M.R. Garey, D.S. Johnson, Computers and Intractability, W.H. Freeman and Company, New York 1979.

[18] J.F. Geelen, A.H.M. Gerards, N. Robertson, G.P. Whittle, *On the Excluded Minors for the Matroids of Branch-Width k*, manuscript, 2002.

[19] J.F. Geelen, A.H.M. Gerards, G.P. Whittle, *Branch-Width and Well-Quasi-Ordering in Matroids and Graphs*, J. Combin. Theory Ser. B 84 (2002), 270–290.

[20] J. Geelen, J. Oxley, D. Vertigan, G. Whittle, *Weak Maps and Stabilizers of Classes of Matroids*, Advances in Appl. Math. 21 (1998), 305–341.

[21] R. Hall, *Excluded Minors for the Matroids of Branch-Width Three*, M.Sc. Thesis, Victoria University of Wellington, 2000.

[22] R. Hall, J. Oxley, C. Semple, G. Whittle, *On Matroids of Branch-Width Three*, submitted, 2001.

[23] P. Hliněný, *On the Excluded Minors for Matroids of Branch-Width Three*, Electronic Journal of Combinatorics 9 (2002), `http://www.combinatorics.org`, #R32.

[24] P. Hliněný, *Equivalence-Free Exhaustive Generation of Represented Matroids*, submitted, 2002.

[25] P. Hliněný, *Branch-Width, Parse Trees, and Monadic Second-Order Logic for Matroids (Extended Abstract)*, STACS 2003, Lecture Notes in Computer Science, to appear.

[26] P. Hliněný, *The Tutte Polynomial for Matroids of Bounded Branch-Width*, submitted, 2002.

[27] P. Hliněný, *Branch-Width, Parse Trees, and Monadic Second-Order Logic for Matroids*, submitted, 2002.

[28] P. Hliněný, *It is Hard to Recognize Free Spikes*, submitted, 2002.

[29] P. Hliněný, *A Parametrized Algorithm for Matroid Branch-Width*, submitted, 2002.

[30] P. Hliněný, *Structural Computations with Represented Matroids – the* MACEK *Program*, `http://www.mcs.vuw.ac.nz/research/macek`, 2002.

[31] P. Hliněný, *Computing the Excluded Minors for Branch-Width* 3 *over Small Fields*, `http://www.mcs.vuw.ac.nz/research/macek/bw3`, 2002.

[32] J. Hopcroft, J. Ullmann, Introduction to Automata Theory, Adisson-Wesley 1979.

[33] F. Jaeger, D.L. Vertigan, D.J.A. Welsh, *On the Computational Complexity of the Jones and Tutte Polynomials*, Math. Proc. Camb. Phil. Soc. 108 (1990), 35–53.

[34] J.A. Makowsky, *Colored Tutte Polynomials and Kauffman Brackets for Graphs of Bounded Tree Width*, submitted.
Extended abstract: Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms, Washington DC, 2001, 487–495,

[35] B.D. McKay, *Isomorph-free exhaustive generation*, J. Algorithms 26 (1998), 306–324.

[36] J. Nešetřil, Teorie Grafů, (in Czech), SNTL 1979.

[37] S.D. Noble, *Evaluating the Tutte Polynomial for Graphs of Bounded Tree-Width*, Combin. Probab. Computing 7 (1998), 307–321.

[38] J.G. Oxley, Matroid Theory, Oxford University Press, 1992,1997, ISBN 0-19-853563-5.

[39] J. Oxley, D. Vertigan, G. Whittle, *On Inequivalent Representations of Matroids over Finite Fields*, J. Combin. Theory Ser. B 67 (1996), 325–343.

[40] N. Robertson, P.D. Seymour, *Graph Minors – A Survey*, Surveys in Combinatorics, Cambridge Univ. Press 1985, 153–171.

[41] N. Robertson, P.D. Seymour, *Graph Minors X. Obstructions to Tree Decomposition*, J. Combin. Theory Ser. B 52 (1991), 153–190.

[42] N. Robertson, P.D. Seymour, *Graph Minors XX. Wagner's Conjecture*, submitted.

[43] D. Seese, *The Structure of Models of Decidable Monadic Theories of Graphs*, Ann. Pure and Appl. Logic 53 (1991), 169–195.

[44] D. Seese, F. Schlottmann, *Logical Ways to Reduce Complexity by Avoiding Grids: A Survey on Results and Problems*, manuscript, 2002.

[45] C. Semple, G. Whittle, *On Representable Matroids Having Neither $U_{2,5}$- nor $U_{3,5}$-minors*, In: Matroid Theory, Contemporary Math. 197 (Amer. Math. Soc. 1995), 377–386.

[46] C. Semple, G. Whittle, *Partial Fields and Matroid Representation*, Advances in Appl. Math. 17 (1996), 184–208.

[47] P.D. Seymour, *Decomposition of Regular Matroids*, J. Combin. Theory Ser. B 28 (1980), 305–359.

[48] W.T. Tutte, *A Homotopy Theorem for Matroids*, Trans. Amer. Math. Soc. 88 (1958), 144-174.

[49] W.T. Tutte, *A Ring in Graph Theory*, Proc. Camb. Phil. Soc. 43 (1947), 26–40.

[50] L.G. Valiant, *The Complexity of Enumeration and Reliability Problems*, SIAM J. Computing 8 (1979), 410–421.

[51] D.L. Vertigan, D.J.A. Welsh, *The Computational Complexity of the Tutte Plane: the Bipartite Case*, Combin. Probab. Computing 1 (1992), 181–187.

[52] D.J.A. Welsh, *The Tutte Polynomial*, Random Structures Algorithms 15 (1999), 210–228.

[53] D.J.A. Welsh, *Matroid Theory*, Academic Press, London 1976. ISBN 0 12 744050 X.

[54] H. Whitney, *On the abstract properties of linear dependence*, Amer. J. Math. 57 (1935), 509–533.