Matroids, Complexity and Computation

Michael Snook

VICTORIA UNIVERSITY OF WELLINGTON

Te Whare Wananga o te Upoko o te Ika a Maui

School of Mathematics, Statistics
and Operations Research

Te Kura Mātai Tatauranga, Rangahau Pūnaha

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the requirements for the degree of
Doctor of Philosophy
in Mathematics.

Victoria University of Wellington

2013

Abstract

The node deletion problem on graphs is: given a graph and integer k , can we delete no more than k vertices to obtain a graph that satisfies some property π . Yannakakis showed that this problem is NP-complete for an infinite family of well-defined properties. The edge deletion problem and matroid deletion problem are similar problems where given a graph or matroid respectively, we are asked if we can delete no more than k edges/elements to obtain a graph/matroid that satisfies a property π . We show that these problems are NP-hard for similar well-defined infinite families of properties.

In 1991 Vertigan showed that it is $\#P$ -complete to count the number of bases of a representable matroid over any fixed field. However no publication has been produced. We consider this problem and show that it is $\#P$ -complete to count the number of bases of matroids representable over any infinite fixed field or finite fields of a fixed characteristic.

There are many different ways of describing a matroid. Not all of these are polynomially equivalent. That is, given one description of a matroid, we cannot create another description for the same matroid in time polynomial in the size of the first description. Due to this, the complexity of matroid problems can vary greatly depending on the method of description used. Given one description a problem might be in P while another description gives an NP-complete problem. Based on these interactions between descriptions, we create and study the hierarchy of all matroid descriptions and generalize this to all descriptions of countable objects.

Acknowledgements

First and foremost I would like to thank my supervisor Dillon Mayhew for the many hours he poured into this thesis and his guidance throughout. I would also like to thank the matroid glut at Victoria University for many helpful and interesting conversations.

Contents

1	Introduction	1
1.1	Recent Advances in Matroid Complexity	1
1.2	Results in the Thesis	3
2	Matroid Deletion	7
2.1	Introduction	7
2.2	The General $GF(q)$ π Deletion Problem	9
2.3	Other Matroid Inputs	23
2.4	Oracle Methods	28
3	Edge Deletion	37
3.1	Introduction	37
3.2	Preliminary Constructions	40
3.3	Masks and the Mask Graph	50
3.4	The Reduction	55
4	Basis and Circuit Counting	61
4.1	Introduction	61
4.2	Basis Counting in Representable Matroids	62
4.3	Other Basis Counting Problems	74
4.4	Circuit Counting	84
5	Nonsuccinct Descriptions	89
5.1	Introduction	89
5.2	Additions to the Hierarchy	90
5.3	Connected Cyclic Flats Axioms	93

6	Input Hierarchy	99
6.1	Introduction	99
6.2	The structure of $\mathcal{N}$	102
6.3	The structure of $\mathcal{M}$	114
7	Future Work	117

List of Figures

2.1	Examples of the graphs N_0 and N' for a graph N	15
2.2	Replacing an edge with copies of N_0 and N'	15
2.3	The graph L_2	16
2.4	An edge weighted graph and its corresponding supine matroid. . . .	25
3.1	Example of the difference in series extensions.	39
3.2	The graph N'	44
3.3	The graphs N_2 and N_3	45
3.4	Replacing the edges and vertices of G with N' , N_2 and N_3	46
3.5	The graph G^2 after deleting some edges and vertices and contracting some series extensions.	47
3.6	Creating the graph G^3	48
3.7	Example of the graph G_s made form G^3	49
3.8	Creating the mask graph $M(G^3, \emptyset)$	52
3.9	The graph N''	56
3.10	The graph N_D	57
5.1	The ordering of the 10 inputs.	90
5.2	The ordering of the 12 inputs.	94

Chapter 1

Introduction

One of the first things you learn when studying matroids is that there is an enormous number of ways to describe a matroid. For example, we could list its independent sets or its circuits. Or we could give a matrix representation or a graph if the matroid has such a description. These descriptions vary greatly in size, but the object being described is the same. This can throw a spanner in the works when studying the complexity of matroid problems. There are many complexity theoretic results in graph theory. However, despite the relationship between graphs and matroids, complexity theory in matroids has received relatively little attention by comparison. This could be in part due to the difficulties incurred by the difference in matroid descriptions. A large part of this thesis will be devoted to the study of the complexity of a number of different matroid problems for different descriptions for matroids. We will also examine how these different descriptions affect the complexity of matroid problems and how they relate to each other.

1.1 Recent Advances in Matroid Complexity

In this section we briefly discuss some of the recent complexity theory results that relate to matroid theory. The field of matroid complexity has been steadily growing. A startling number of combinatorial problems can be reduced to matroid problems. Thus it is not uncommon for matroid complexity and algorithmic results to be of use in a wide range of fields. This is because it allows the large amount of algorithmic results in matroid theory to be used in applications in many different fields. For example, the matroid intersection problem has been used in evolutionary biology [62] and recently to create a fixed parameter tractable algorithm for finding

polytrees [30]. The matroid matching problem is known to be difficult to solve [42, 52]. In the last few years, a number of polynomial-time approximation schemes have been produced for the matroid matching problem [11, 50, 67]. Furthermore a new randomized algorithm has been added to the extensive list of algorithms for matroid matching [38]. Similarly, the matroid partition problem has been used in quantum computing [5].

Another recent example of a matroid complexity problem leading to complexity results in other fields is the rank reduction problem. The rank reduction problem for matroids is a deletion problem where we are asked if we can find a minimum sized set whose removal reduces the rank by at least k . For graphic matroids, this is equivalent to the k -cut problem. It was shown that this is NP-hard when considering the intersection of two partition matroids [44]. The proof used also shows that the maximum vertex cover problem is NP-hard, answering an open problem of B. Simeone [1]. However the maximum vertex cover problem on bipartite graphs has been independently shown to be NP-hard by Apollonio et al [6] and Caskurlu et al [18].

The complexity of the Tutte polynomial has received a large amount of study [13, 21, 31, 40, 41, 74, 76, 79]. This has led to the complexity of evaluating the Tutte polynomial being known for all points (x, y) , except possibly for $(1, 1)$. The point $(1, 1)$ is known to coincide with the number of bases of a matroid. We consider the complexity of evaluating the Tutte polynomial at this point in Chapter 4. It is known to be #P-complete to evaluate the Tutte polynomial at almost all points. For the handful of points for which it is not #P-complete to evaluate the Tutte polynomial, polynomial-time algorithms to compute most of them have been known for a while [40, 76]. There have been two recent additions to this. These are to compute $(-1, 1)$ of a binary matroid [59] and (j, j^2) for a ternary matroid where $j = e^{2\pi i/3}$ [32]. Due to the difficulty in computing the Tutte polynomial, the recent trend has been to investigate approximation algorithms for the Tutte polynomial [33, 34, 35, 36].

Connections between matroid theory, fixed-parameter tractability and kernelization have recently been discovered [2, 23, 26, 53]. This is important because when a problem is found to be NP-hard, the next step is often to investigate the possibility of fixed-parameter tractability or kernelization. Among other problems, this has provided kernelization for the following problems. The problem ODD CYCLE TRANSVERSAL where we are asked whether we can make a given

graph bipartite by deleting no more than k vertices [45]. The problem MULTIWAY CUT is given an edge-weighted graph with a number of terminal vertices, can we find the minimum weighted set of edges whose removal disconnects all terminals [46]. Furthermore, in the same paper a kernel for ALMOST 2-SAT was provided where in ALMOST 2-SAT we are asked if by removing at most k clauses, can we make a given 2-CNF formula satisfiable.

Other recent matroid complexity results include works on the matroid-greedoid partition problem [10], the matroid center problem [19] and the matroid isomorphism problem [60].

1.2 Results in the Thesis

The node deletion problem is as follows: given a graph and integer k , can we delete no more than k vertices to obtain a graph that satisfies some property π . In 1978 it was shown by Lewis and Yannakakis [51] and Yannakakis [80] that the node deletion problem is NP-hard for a well defined infinite family of properties. We note that by simple counting arguments, there must be an infinite number of NP-complete properties. That is why we are interested in well defined infinite families of properties. Many graph problems can be restated as such a deletion problem. For example, the vertex cover problem is equivalent to the node deletion problem with the property of having no edges. Or the feedback vertex problem can be restated as the node deletion problem with the property of having no directed cycles. The obvious question this raises is what other structures and families of properties can the same be done for? It was asked by Yannakakis whether or not there is an infinite family of properties such that the edge deletion problem is NP-complete. There have been a number of partial results on this, but they are generally for restrictive families of properties [3, 8, 9, 16, 81]. There is a close relationship between the edges of a graph and the elements of the corresponding matroid. Due to this, there is no reason to stop at the edge deletion problem. We consider both the edge deletion problem and the matroid deletion problem. We show that both of these are NP-hard for similar infinite families of properties. The matroid deletion problem will be dealt with in Chapter 2. We will consider different methods of matroid description and families of properties that are NP-hard for these descriptions. We will then deal with the edge deletion problem in Chapter 3. We will show that the edge deletion problem is NP-hard for a family

of properties that is less restrictive than the current results.

In Chapter 4 we consider the difficulty of counting the number of bases of matroids. This is equivalent to computing the Tutte polynomial at the point $(1, 1)$. As this is not a decision problem, it does not belong to the class NP. The counting analogue of the class NP is the class #P. This class was first introduced by Valiant [71]. The notion of completeness in NP carries over to the class #P where #P-complete problems are the hardest problems in the class #P. It has been shown that a number of different basis counting problems are #P-complete. For example, it is #P-complete to count the number of bases of transversal matroids or bicircular matroids ([21] and [31] respectively). In 1991 Vertigan proved that it is #P-complete to count the number of bases of representable matroids over any fixed field. However, no publication was produced. We note that special cases of this result can be derived from the results on transversal or bicircular matroids as there are polynomial-time constructions from both of these to representable matroids. However, as far as we know, these constructions have never been explicitly written down. Our main focus of Chapter 4 will be providing proofs for special cases of this result by providing appropriate constructions. We do not show that it is #P-complete to count bases of matroids representable over any fixed field. However, we provide proofs for the fact that it is #P-complete to count the number of bases of matroids representable over any fixed infinite field, or over fields of a fixed characteristic. We will also show that a number of other basis counting problems are #P-complete.

As well as basis counting, Chapter 4 will also contain results on the difficulty of counting circuits for representable matroids over fixed fields. The motivation is that circuit counting results have been used to show that some basis counting problems are #P-complete [21].

Matroid complexity has been dominated by two approaches to describing matroids. The first describes the matroid to the Turing machine using an oracle. This machine will have a subroutine that when asked a specific question will answer in unit time. For example, an independence oracle would answer whether or not a set is independent. The problem here is that as we are not using a deterministic Turing machine, we cannot obtain NP-completeness results. This is because problems in NP can be verified in polynomial time in terms of the size of their input. However, when using an oracle, there is no input. The second approach is to describe the matroid by some succinct description such as a matrix representation.

However, not all matroids have a succinct description, limiting the matroids that can be studied. The natural assumption with complexity theory in graphs is that the description is polynomial in the size of the graph. However, as matroids have a richer structure, there is no reason to believe this is true for matroids. In fact, this is generally not true. This leads to a third option that has received relatively little attention. This is to describe the matroid via a list of subsets of the ground set. For example, we could list the independent sets or circuits of the matroid. The concern with this approach is that the description could become large and most problems might become artificially easy. This appears to not be the case though as a number of problems have been shown to be NP-complete under these descriptions [55]. Furthermore, the description used can change the complexity of solving the problem. What might be in P for one description may be NP-complete for a different description. Because of this we can create a hierarchy of descriptions where for descriptions A and B , $A \leq B$ if given description B , we can create description A in polynomial time. Such a hierarchy was created by Mayhew for ten natural matroid descriptions [55]. In Chapter 5 we add two more matroid descriptions to this hierarchy.

We go further in Chapter 6 and study the hierarchy that all such matroid descriptions are embedded in. The theory of computability has been studied in great detail. Given two sets of natural numbers A and B , we say that A is Turing reducible to B ($A \leq_T B$) if given an oracle for membership of B , we can decide membership of A . So A and B are in the same Turing degree if $A \leq_T B$ and $B \leq_T A$. The Turing degrees are equivalence classes of sets. The hierarchy of the Turing degrees has been widely studied and the structure of the Turing degrees is well known. The Turing degrees are essentially a measure of how powerful certain oracles are. We have a similar notion of how powerful certain matroid descriptions are based on whether or not we can translate from one to the other in polynomial time. However, the structure that the matroid descriptions are embedded in has not received the same amount of study. Using ideas from computability theory, we create a hierarchy of matroid descriptions much like the Turing degrees. As matroids are countable, we generalize this hierarchy to all descriptions of countable objects. Reader beware, there's very little matroid theory in these parts.

Basic knowledge of matroids and complexity theory will be assumed. For references on matroids see [58] and for complexity theory see [7, 24, 27, 28]. Note that it is common practice to refer to both the ground set of a matroid and the

edge set of a graph by E . Whenever we use E we will be referring to the ground set of a matroid. The edge set of a graph will be denoted $\mathcal{E}$. The graph G will have edge set $\mathcal{E}(G)$ and the matroid M will have ground set $E(M)$. Moreover, the set of vertices of the graph G will be denoted by $V(G)$.

Chapter 2

Matroid Deletion

2.1 Introduction

The node deletion problem for graphs can be stated as follows: given a graph and integer k , can we delete at most k vertices to obtain a graph that satisfies a property π ? (A graph property is a class of graphs that is closed under isomorphism.) Alternatively, what is the largest induced subgraph that belongs to π ? It was shown by Lewis and Yannakakis [51] and Yannakakis [80] that this problem is NP-hard for an infinite class of properties. This gives rise to the question: can the same be done for matroids?

The main focus of this chapter will be to show that for an infinite class of properties, the π deletion problem is NP-hard when the matroid is described by a $GF(q)$ representation or lists of subsets of the ground set, such as the independent sets. We will also show that if the π deletion problem is not polynomial-time reducible to deciding if a matroid satisfies π , then it cannot be solved in polynomial time by a Turing machine equipped with an oracle, for many natural oracles. If the property we are interested in can be recognized in polynomial time, then we can replace NP-hard with NP-complete and all the results presented in this chapter will hold.

We will define a *matroid property* π to be a class of matroids which will be closed under isomorphism. We will often refer to being in the class as satisfying π or not being in the class as violating π . We call a property π *non-trivial* if

1. there are infinitely many matroids that satisfy π ;
2. the matroid consisting of a single independent element satisfies π ; and

3. there exist matroids that violate π .

A property is non-trivial in a class of matroids if it obeys the same conditions when restricted to matroids in the class.

Suppose π is only satisfied by a finite number of matroids. That is, there are only m matroids that satisfy π . Then there will exist a maximum sized matroid that satisfies π . Let $\{1, \dots, t\}$ be the ground set of this maximum sized matroid. To decide if a matroid M is at most k deletions away from satisfying π , we can simply check every subset of at most t elements of the ground set of M and test if they are one of the matroids $M_i \in \{M_1, \dots, M_m\}$. This can be done in polynomial time as the number of matroids $M_1, \dots, M_m$ does not change with the size of the input. The largest such M_i will tell us whether or not M is at most k deletions away from satisfying π . Therefore to obtain any meaningful complexity results it is necessary that our property π satisfies at least conditions 1 and 3.

Suppose we have a matroid property π . If for all matroids M that satisfy π , $M \setminus e$ satisfies π for all $e \in E(M)$, then we say π is *hereditary*. A matroid property that is hereditary and does not satisfy 2 can only be satisfied by a collection of loops. Either it is satisfied by all loops, or there is a maximum number of loops that satisfy it. Either way, the π deletion problem for such a property would be solvable in polynomial time. Thus we assume that the property π is non-trivial. An example of a matroid property that is non-trivial and hereditary is the class of graphic matroids.

For an element $e \in E(M)$ a *series extension* of e removes e and replaces e with two elements e_1 and e_2 such that any circuit C that contained e is replaced with $(C \setminus e) \cup \{e_1, e_2\}$. An element e_2 is *parallel* with an element e_1 if $\{e_1, e_2\}$ is a circuit and for all other circuits C such that $e_i \in C$, $(C \setminus e_i) \cup e_j$ is a circuit for $i \neq j$, $i, j \in \{1, 2\}$. A *parallel class* is a set of elements that are parallel with all elements in the set. A *parallel extension* of an element e is constructed by adding an element parallel with e . For some matroid modification m , we say a property π is *closed under m* if performing m on a matroid that satisfies π does not produce a matroid that violates π . Note though that this allows the possibility that applying m to a matroid that violates π can create a matroid that satisfies π . Because of this, we use a slightly stronger condition. We say a property π is *completely closed under m* if applying m to a matroid does not change whether or not the matroid satisfies π . Using the same example as before, the class of graphic matroids is an example of a property that is completely closed under series and parallel extensions.

The remainder of this chapter is broken up into three sections. Section 2 will look at deletion problems when the input is a $GF(q)$ representation of a matroid. We will show that this class of problems is NP-hard for properties that are non-trivial, hereditary and completely closed under series and parallel extensions.

In Section 3 we will consider a different method for describing our matroids. This will be via a list of the independent sets. We show that the corresponding deletion problem is also NP-hard for a class of non-trivial hereditary properties. We also show that this can be generalized to many other methods of describing a matroid by listing subsets and that the problem remains NP-hard for these methods of description.

In the fourth section we will no longer be concerned with NP-hardness results. Instead, we will consider a Turing machine equipped with an independence oracle. We will show that if the π deletion problem is not polynomially reducible to deciding if a matroid satisfies π , then it cannot be solved by such a Turing machine in polynomial time, for many properties. That is, either the π deletion problem is no harder than deciding if a matroid satisfies π or there does not exist a Turing machine equipped with such an oracle that can solve the π deletion problem in polynomial time.

2.2 The General $GF(q)$ π Deletion Problem

In this section we deal with the following problem:

THE GENERAL $GF(q)$ π DELETION PROBLEM

INSTANCE: A $GF(q)$ matrix A and an integer k .

QUESTION: Can we delete at most k elements from $M[A]$ to obtain a matroid M' that satisfies π ?

To find the complexity of this problem we will first show that the following restricted problem is NP-hard.

THE RESTRICTED $GF(q)$ π DELETION PROBLEM

INSTANCE: A $GF(q)$ matrix A with a distinguished basis of the column space B and an integer k .

QUESTION: Can we delete at most k elements from $E(M[A]) - B$ to obtain a matroid M' that satisfies π ?

For both these problems, the input will be a $\text{GF}(q)$ matrix. This is because it can be difficult to create a $\text{GF}(q)$ representation of a matroid or even to decide if a matroid has a $\text{GF}(q)$ representation. For example, it is even difficult to decide if a matroid is binary [66]. Constructing a $\text{GF}(q)$ representation remains difficult even if we are told that the given matroid has a $\text{GF}(q)$ representation. So to make sure that the difficulty in solving the above problems does not come from the difficulty of creating a $\text{GF}(q)$ representation, we will always assume that our $\text{GF}(q)$ representable matroids are given to us as a $\text{GF}(q)$ matrix.

We will consider any set $F \subseteq E(M)$ (or $F \subseteq (E(M) - B)$ for the restricted version) of k or less elements whose deletion produces a matroid that satisfies π to be a solution to these deletion problems. These sets are not strictly speaking a solution to the deletion problem because the deletion problem is a yes or no question. However, given a minimum set whose deletion produces a matroid that satisfies π , it can be very easily decided if the answer to the deletion problem is yes or no. Thus we can (and will) think of a minimum (or optimal) solution to the deletion problem to be a minimum set F whose deletion produces a matroid that satisfies π .

In Theorem 2.2.6 we will prove that the restricted $\text{GF}(q)$ π deletion problem is NP-hard for an infinite family of properties. The proof of this uses the techniques used by Yannakakis to show that the vertex deletion problem on graphs is NP-hard [80]. With this, we will then present a reduction from the restricted $\text{GF}(q)$ π deletion problem to the general $\text{GF}(q)$ π deletion problem in Theorem 2.2.8.

Before we prove the NP-hardness result, we will show that for the class of properties π we are interested in, a free matroid, denoted $U_{n,n}$, satisfies π . ($U_{r,n}$ is the matroid with n elements and any r element subset is independent). This result will be useful in the proof of Lemma 2.2.2 which in turn will be used in the proof of Theorem 2.2.6.

Lemma 2.2.1. *If a non-trivial property π is hereditary and completely closed under series extensions, then π is satisfied by the free matroid $U_{n,n}$.*

Proof. Because π is non-trivial, the matroid $U_{1,1}$ satisfies π . Any series extension of $U_{n-1,n-1}$ is isomorphic to $U_{n,n}$. Because π is completely closed under series extensions, this will satisfy π and the result follows by induction. $\square$

Let $A = [I|C]$ be a representation over the field $\text{GF}(q)$ of the matroid M with basis B , where B is the set of labels of the columns of I . Note that there is a

correspondence between the rows of the matrix A and the columns of I . We could remove I and label the rows with the set of labels B . We can transform this into an edge weighted bipartite graph $G = (\mathcal{E}(G), w, B, E(M) - B)$ with vertex sets B and $E(M) - B$ and weighting function w by adding in an edge between $b \in B$, $c \in E(M) - B$ if and only if the entry in column c , row b is non-zero. For the edge e , we let $w(e)$ be the weighting given by the non-zero entry. When the graph is constructed from a matroid using this construction, denote it by $G(A, B)$ for the $GF(q)$ representation A of the matroid $M[A]$ and basis B . For each $GF(q)$ representation of a matroid with basis B , there exists a unique edge weighted bipartite graph with vertex sets B and $E(M) - B$. For any $GF(q)$ edge weighted bipartite graph with vertex sets B and $E(M) - B$ there exists a unique $GF(q)$ representable matroid with basis B . Due to the direct correspondence between the vertices of $G(A, B)$ and elements of the matroid M , we will often refer to vertices as if they are elements of the matroid and vice versa. In particular, take the edge weighted bipartite graph $G = (\mathcal{E}(G), w, U, V)$ and its corresponding matroid with basis B corresponding to the vertices U . We will often refer to U as B and V as $E(M) - B$. We will also sometimes refer to U as the basis side of the bipartition of G and V as the non-basis side of the bipartition of G . Note that for all $e \in E(M) - B$, if $G(A, B)$ is the edge weighted bipartite graph associated with M , $G(A, B) \setminus e$ corresponds to the matroid $M \setminus e$. The same holds if e is a coloop. For the $GF(q)$ edge weighted bipartite graph $G = (\mathcal{E}(G), w, U, V)$, we will denote its corresponding matroid with basis elements U by $M(G, U)$.

Before we prove the NP-hardness of the restricted $GF(q)$ π deletion problem, we require a few specific definitions and constructions. Take two vectors $A = (a_1, a_2, \dots, a_r)$ and $B = (b_1, b_2, \dots, b_t)$. Then $A <_l B$ (lexicographically smaller than) if $a_1 < b_1$ or $a_1 = b_1, \dots, a_i = b_i$ and $a_{i+1} < b_{i+1}$. If $r < t$ and $a_i = b_i$ for all $i \in \{1, \dots, r\}$ then $A <_l B$.

Let G be a connected graph, v be a vertex of G and let $D_1, \dots, D_{k(v)}$ be the connected components of $G \setminus v$ where $|V(D_k)| \geq |V(D_l)|$ for all $k < l$. Let H_j be the vertex induced subgraph of $V(D_j) \cup \{v\}$. The graphs H_j will be referred to as the components of G with respect to v . Now define

$$\alpha_v(G) = (|V(H_1)|, \dots, |V(H_{k(v)})|).$$

Define $c(G)$ to be the set of vertices v that give the lexicographically smallest $\alpha_v(G)$. Note that if G is 2-connected, then $c(G) = V(G)$. Also note that if G has

a cut vertex, then every vertex in $c(G)$ is a cut vertex. Finally define

$$\alpha(G) = \alpha_v(G),$$

where $v \in c(G)$. That is, $\alpha(G)$ is the lexicographically minimum sequence $\alpha_v(G)$ for the graph G . This gives an ordering on graphs represented by $<_\alpha$ where $G <_\alpha G'$ if $\alpha(G) <_l \alpha(G')$.

Let G be a graph. We will refer to the operation of creating a direct sum of copies of G as *repeating* G . Define $\oplus_k G$ to be the graph made from repeating G k times. We will show that there exists a graph N with certain properties that represents a $GF(q)$ matroid such that $\oplus_k N$ violates π for some k . This graph N will play an important part in our reduction in Theorem 2.2.6.

A connected bipartite graph $G = (\mathcal{E}(G), U, V)$ is a *star graph* if $|U| = 1$ or $|V| = 1$ (or both).

Lemma 2.2.2. *Let π be a hereditary non-trivial $GF(q)$ matroid property that is completely closed under parallel and series extensions. Then there exists an edge weighted bipartite graph $N = (\mathcal{E}(N), w, B, E(M) - B)$ such that*

1. N is connected;
2. N is not a star graph;
3. $c(N) \cap (E(M) - B) \neq \emptyset$;
4. there exists k such that $M(\oplus_k N, B_k)$ is not in π , where B_k is k copies of the vertex set B ; and
5. if $G <_\alpha N$ then G does not satisfy all of 1 – 4.

Proof. Because π is non-trivial there exists a graph N that violates π and therefore there exists a graph N and integer k such that $\oplus_k N$ violates π . Moreover, by Lemma 2.2.1 we know that any free matroid satisfies π . Any graph that has no edges and is associated with a $GF(q)$ representable matroid represents a free matroid possibly with additional loops. Because π is completely closed under series extensions, we can extend each loop in series. This will give a parallel class. We can then remove an element from each parallel class created from a loop. Because π is also completely closed under parallel extensions, these operations will not change whether or not the graph corresponds to a matroid that satisfies π . However, the

resulting matroid will correspond to a free matroid and therefore satisfy π when repeated any number of times. Thus all bipartite graphs that represent a matroid that violates π must have at least one edge. Note that because we can change any loop into a coloop using series extensions and then removing parallel elements, we can assume that any isolated vertex in N is in B . This will not change $\alpha(N)$ as we are just moving isolated vertices from one side of the bipartition to the other. Therefore, we can assume that N corresponds to a matroid with no loops.

Now suppose that all connected $GF(q)$ edge weighted bipartite graphs represent matroids that satisfy π . Then take any graph $N = (\mathcal{E}(N), w, B, E(M) - B)$ such that $M(N, B)$ violates π . As N corresponds to a matroid without loops, we can assume that every connected component of N has a vertex in B . Now add in a single vertex v to the set $E(M) - B$ that is adjacent to at least one vertex of B in each connected component. Give the edges created in this way weighting 1. This new graph will now be connected and its corresponding matroid will therefore satisfy π . We can obtain $M(N, B)$ from this new matroid by deleting v . This is equivalent to deleting the element v from the matroid as $v \in E(M) - B$. Because π is hereditary, $M(N, B)$ must also satisfy π . This is a contradiction and therefore there exists an integer k and connected graph N such that $M(\oplus_k N, B_k)$ violates π .

Now suppose that for all connected graphs N that represent matroids that violate π when repeated k or more times for some $k \in \mathbb{Z}$, $c(N) \cap (E(M) - B) = \emptyset$. Then take a connected graph $N = (\mathcal{E}(N), w, B, E(M) - B)$ such that $M(N, B)$ violates π when repeated k or more times for some integer k . Add in a single vertex v to $E(M) - B$ that is adjacent to every vertex in B . Then add $|E(M)| + 1$ vertices to B that are only adjacent to v . Give all the new edges created weighting 1. We claim that this will produce a graph $N' = (\mathcal{E}(N'), w', B', E(M') - B')$ such that $c(N') = \{v\}$. This is because for vertices $u \neq v$, $\alpha_u(N') >_\alpha (|E(M)| + 1)$ because there will be a component with respect to u that contains v and the $|E(M)| + 1$ vertices added to B in N . However $\alpha_v(N') \leq_\alpha (|E(M)| + 1)$ as deleting v will turn each vertex added to B into a separate component containing just the single vertex. The graph N' will satisfy π because $c(N') \cap (E(M') - B') \neq \emptyset$. However N can be obtained from N' by deleting v and then all vertices added to B' (which represent coloops after deleting v). Because of this, $M(N, B)$ can be obtained from the matroid corresponding to N' by deleting elements. Recall that N' satisfies π while N violates π . Because π is hereditary this is a contradiction. Therefore

there will always exist a connected graph N such that $c(N) \cap (E(M) - B) \neq \emptyset$ and $M(\oplus_k N, B_k)$ violates π .

Finally, from Lemma 2.2.1 we know that any free matroid satisfies π . Any graph made from the direct sum of star graphs corresponds to a matroid such that every connected component is a circuit or a parallel class. Such a matroid can be obtained from the free matroid by adding parallel elements and series extensions and therefore satisfies π . Thus N cannot be a star graph.

Therefore there exist graphs that satisfy 1 – 4. We can simply choose N to be the minimum graph under the ordering $<_\alpha$ that satisfies properties 1 – 4. $\square$

Let $N = (\mathcal{E}(N), w, B, E(M) - B)$ be the graph provided by Lemma 2.2.2. Fix some $s \in c(N) \cap (E(M) - B)$. Recall that H is a component of N with respect to s if $s \in H$, H is a vertex induced subgraph of N and $H \setminus s$ is a connected component of $N \setminus s$. Let $N_0 = (\mathcal{E}(N_0), w_0, B_0, E(M_0) - B_0)$ be a fixed largest component of N with respect to s . Define the graph $N' = N \setminus (N_0 \setminus s)$. Note that if N is 2-connected, then $N' = s$ and $N_0 = N$. For example, the graphs N_0 and N' for a graph N can be seen in Figure 2.1. Note that identifying s in N_0 with s in N' gives the graph N .

Because N is connected and has at least two vertices in $E(M) - B$ (because N is not a star graph), there will exist a component of N with respect to s that includes one of these vertices. Furthermore, s is in every component of N with respect to s and so will be in the same component as this vertex. Any component of N with respect to s that does not contain a vertex in $E(M) - B$ other than s can only have a maximum of two vertices. However, if a component has a vertex in $E - B$ other than s , then it must contain at least 3 vertices. Otherwise the component would not be connected in $N \setminus \{s\}$. Therefore a maximum component with respect to s must have at least two vertices from $E(M) - B$. Therefore there will always exist such a vertex in N_0 . Fix some vertex $d \neq s$ of N_0 that is in $E(M_0) - B_0$.

For any graph G , we will construct a graph $G' = (\mathcal{E}(G'), w', B', E(M') - B')$ by performing the following operations on G .

1. Take each vertex v of G and replace it with a copy of N' by identifying the vertex v with $s \in V(N')$.
2. Then replace each edge $\{u, v\}$ of G with a copy of N_0 by arbitrarily identifying one of u and v with $s \in V(N_0)$ and the other with $d \in V(N_0)$.

Figure 2.1: Examples of the graphs N_0 and N' for a graph N .

For an illustration of this see Figure 2.2. Call this graph G' . Note that G' is bipartite with vertex sets B' and $E(M') - B'$. Note also that all vertices of G were identified with a vertex in $E(M_0) - B_0$. Thus they are all in the same side of the bipartition of G' . That is, they are all in $E(M') - B'$.

Figure 2.2: Replacing an edge with copies of N_0 and N' .

Let $G_N = \oplus_{nk} G' = (\mathcal{E}(G_N), w_N, B_N, E(M_N) - B_N)$ where $n = |V(G)|$ and $B' \subseteq B_N$. Remember that $M(\oplus_k N, B_k)$ violates π . Therefore so does $M(G_N, B_N)$ as we can obtain $M(\oplus_k N, B_k)$ from $M(G_N, B_N)$ by deleting non-basis elements and coloops and then removing series extensions. From our construction of G_N , B_N consists of many copies of the elements in B' . Note that G_N is bipartite with $E(M_N) - B_N$ containing all vertices of G .

Let V_G be a vertex cover for G . Delete the vertices that were identified with the vertices in V_G from each of the nk copies of G' in G_N . Denote the graph created as $G_N - V_G$. As every edge of G will be incident with a vertex in V_G , every copy of N_0 that replaced an edge to make G_N will include a vertex in V_G . When we delete these vertices, every copy of N_0 will have had at least one vertex s and/or d deleted from it (because the vertices of G only correspond to the vertices s or d). We will construct a graph G_j that satisfies π when repeated any number of times such that $G_N - V_G$ is a subgraph of a graph that can be obtained by repeating G_j a number of times. This will show that $G_N - V_G$ satisfies π . Suppose that G has maximum vertex degree j . Construct the graph $L_j = (\mathcal{E}(L_j), w_{L_j}, B_{L_j}, E(M_{L_j}) - B_{L_j})$ by the

following method.

1. Take a copy of N' and at s attach j copies of $N_0 \setminus s$ by identifying d from $N_0 \setminus s$ with s from N' .
2. Then attach j copies of $N_0 \setminus d$ by identifying s in $N_0 \setminus d$ with s in N' .

For example see Figure 2.3.

Figure 2.3: The graph L_2 .

It is possible that L_j is not connected. This will happen if $N_0 \setminus d$ has more than one component. Denote the component of L_j that contains N' as $G_j = (\mathcal{E}(G_j), w_j, B_j, E(M_j) - B_j)$. Note that if $N_0 \setminus d$ is connected, then $L_j = G_j$.

Lemma 2.2.3. *Let G be some graph and let G_j be constructed from G by the above construction. Then we can create a graph G_C that is isomorphic to any component of L_j by deleting vertices from $E(M_j) - B_j$ and then vertices that correspond to coloops from G_j .*

Proof. Take some graph G_C that is isomorphic to a connected component C of L_j that is not G_j in the construction above. If there is no such component, then we are done. This component C cannot contain s or d because if it did, it would be connected to N' and therefore be G_j . Take G_j and delete the vertex s in N' that has been identified with each copy of s and d . Then one of the resulting components will be isomorphic to C . We can then delete all vertices in $E(M_{L_j}) - B_{L_j}$ from the other components which will leave G_C and a number of vertices representing coloops. We can then delete all these vertices and we are left with G_C . $\square$

Now, take G_j and attach $|V(N_0)|$ vertices in B_j to s by adding $|V(N_0)|$ new vertices that are joined to s by a single edge. Give the added edges a weighting of 1. Call this new graph $G'_j = (\mathcal{E}(G'_j), w'_j, B'_j, E(M'_j) - B'_j)$. We will use G'_j to show that G_j can be repeated any number of times without violating π .

Lemma 2.2.4. $G'_j <_\alpha N$

Proof. Consider $G'_j \setminus v$ where $v \neq s$. The graph $G'_j \setminus v$ will have a connected component of size at least $|V(N_0)| + 1$. This will be the component containing s and the $|V(N_0)|$ added vertices. Now consider $G'_j \setminus s$. This will consist of the graph $G_j \setminus s$ plus $|V(N_0)|$ isolated vertices. Note that $G'_j \setminus s$ will have no components of size greater than $|V(N_0)| - 1$. If $|V(N')| < |V(N_0)|$ then $G'_j \setminus s$ will have no component of size greater than $|V(N_0)| - 2$. Thus attaching the extra vertices and edges to G_j will make $c(G'_j) = \{s\}$. If we cut the graph G'_j at s then we will get the connected components of N' with respect to s plus $2j$ components of size no greater than $|V(N_0)| - 1$. Note that N' has one fewer maximum sized connected component with respect to s than N because N' is created by removing such a maximum sized connected component. Thus when compared to $N \setminus s$, $G'_j \setminus s$ will have either a smaller maximum sized connected component with respect to s or one fewer maximum sized connected components with respect to s . Either way, it shows that $G'_j <_\alpha N$. □

Note that it is likely that N is 2-connected. The previous proof gives the illusion that we assume that N is not 2-connected. This is not the case. We will show that if N is 2-connected then this still holds. Suppose N is 2-connected. Then $L_j = G_j$ and will consist of j copies of $N_0 \setminus s$ and j copies of $N_0 \setminus d$ joined at a single vertex by identifying the vertices s in the copies of $N_0 \setminus d$ and identifying d from each of the copies of $N_0 \setminus s$. If we delete the vertex that the copies of $N_0 \setminus s$ and $N_0 \setminus d$ are joined by, then we will be left with disjoint graphs with no more than $|V(N_0)| - 2 = |V(N)| - 2$ vertices. However, if N is 2-connected, then $\alpha(N) = |V(N)|$. This shows that even if N is 2-connected, $G'_j <_\alpha N$.

Lemma 2.2.5. *The graph G_j can be repeated an arbitrary number of times without violating π .*

Proof. Note that G'_j satisfies 1 – 3 from Lemma 2.2.2 and by Lemma 2.2.4 $G'_j <_\alpha N$. Therefore G'_j cannot satisfy 4 from Lemma 2.2.2. Because of this, G'_j can

be repeated an arbitrary number of times without violating π . Furthermore as $M(G'_j, B'_j)$ can be obtained by a series extension of $M(G_j, B_j)$ and π is completely closed under series extensions, G_j can be repeated an arbitrary number of times without violating π . $\square$

Let $\gamma_\pi^R(M, B)$ be the minimum number of elements deleted in a solution to the restricted $GF(q)$ π deletion problem on the matroid M with distinguished basis B . Similarly, let $\gamma_\pi(M)$ be the minimum number of elements deleted in a solution to the general $GF(q)$ π deletion problem.

Theorem 2.2.6. *Let π be a hereditary non-trivial $GF(q)$ matroid property that is completely closed under parallel and series extensions. Then the restricted $GF(q)$ π deletion problem is NP-hard.*

Proof. This will be shown by a reduction from the vertex cover problem. Let G be a graph for which we want a vertex cover. Let V_G be a minimum vertex cover for G . Furthermore, let the graphs N , G' , G_N and G_j be constructed from G and π by the above constructions. We will show that $|V_G| \leq l$ if and only if $\gamma_\pi^R(M(G_N, B_N), B_N) \leq nkl$.

First suppose that $|V_G| \leq l$. Recall that the vertices of G are not in B_N . Due to this, deleting them from the graph is the same as deleting them from the corresponding matroid.

Consider the connected components of $G_N - V_G$. These will consist of N' with a number of copies (equal to or less than the maximum vertex degree of G) of $N_0 \setminus s$ or $N_0 \setminus d$ joined at the vertex s in N' , and possibly components of $N_0 \setminus d$ that do not contain s . Consider a component K . As every copy of N_0 contains a deleted vertex (either s or d), by Lemma 2.2.3 K will be a subgraph of G_j . Therefore any component of $G_N - V_G$ will be a subgraph of G_j . It follows that $G_N - V_G$ is a subgraph of $\oplus_m G_j$ for some $m \in \mathbb{Z}$. Recall that by Lemma 2.2.5, G_j satisfies π when repeated any number of times. Suppose $G_N - V_G$ has m connected components $K_1, \dots, K_m$. To obtain $G_N - V_G$ from $\oplus_m G_j$, we can take the i th copy of G_j and delete all vertices in $E(M_j) - B_j$ that do not correspond to a vertex in K_i . We can then delete any remaining vertices in B_j that are adjacent to no vertex (corresponding to coloops in the resulting matroid). Do this for each connected component of $G_N - V_G$. Call the resulting graph G'' . Each connected component of G'' will consist of the component K_i with a number of degree one vertices joined to s by an edge. Now all the remaining vertices in $G'' - (G_N - V_G)$ correspond

to a series extension of the element $s \in c(G_j)$ in a subgraph of G_j . Because π is hereditary and G_j satisfies π , G'' will still correspond to a matroid that satisfies π as we have only deleted non-basis elements and coloops. As π is completely closed under series extensions, is hereditary and G_j satisfies π when repeated any number of times, $G_N - V_G$ must also satisfy π . Thus $\gamma_\pi^R(M(G_N, B'), B') \leq nkl$.

Now suppose that $|V_G| \geq l+1$. Recall that G' is the graph made from replacing the vertices and edges of G with copies of N' and N_0 respectively. Let F be a solution to the restricted $GF(q)$ π deletion problem on the matroid $M(G_N, B_N)$. Consider a copy of G' and choose a copy of N_0 that replaced an edge and one of the copies of N' that replaced a vertex by identifying $s \in N_0$ with $s \in N'$. Restricting to these gives the graph N . From G' , delete all vertices in $E(M') - B'$ and any vertices that now correspond to coloops that are not in the graph N that corresponds to this edge and vertex. The resulting graph will correspond to a matroid that can be obtained from N by series extensions of s and/or d . Do this for each copy of G' in G_N . This will result in a graph whose corresponding matroid can be obtained by series extensions to the matroid $M(\oplus_{nk} N, B_{nk})$ where B_{nk} is nk copies of the vertices B (recall that B is the basis side of the bipartition of N). Thus this graph will correspond to a matroid that violates π because π is completely closed under series extensions and k or more copies of N violate π . Because of this, to satisfy π we know that no more than $k-1$ copies of G' can still have N as a subgraph. We have nk copies of G' so at least $nk - (k-1) = (n-1)k+1$ cannot contain N as a subgraph. Consider such a copy of G' , denoted G'' , that has had deletions so that it contains no subgraph isomorphic to N . Then in every subgraph isomorphic to N we need to delete at least one vertex.

We will show that without loss of generality, we can assume that all vertices deleted correspond to vertices of the original graph G . Suppose we have a solution that to create G'' , removes vertices that do not correspond to vertices of G (the vertices s and d in G'). If we add such a vertex back in to each copy of N and instead remove s from the same copy of N , we will still have a matroid whose graph does not have N as a subgraph. Denote this new set of elements deleted F' and the graph created by deleting these elements from G' as G''' . Note that $|F'| \leq |F|$.

By our construction every copy of s corresponds to a vertex in G . Suppose that there is still an edge in G after deleting the vertices that are in F' . This will

correspond to a copy of N_0 with a copy of N' attached to each of s and d in G' that has no vertices in F' . When we delete the elements in F' from G' , the resulting graph G''' will still have N as a subgraph. But then F would not have removed all copies of N from this copy of G' . Therefore F would not be a solution to the restricted $GF(q)$ π deletion problem and hence a contradiction. Thus F' contains a vertex cover for G and therefore there are at least $l + 1$ elements in F' that were deleted from a copy of G' to create G''' . This implies that there must have been at least $l + 1$ elements in F deleted to make G'' as $|F| \geq |F'|$. We must delete these elements from $(n - 1)k + 1$ copies of G' showing that

$$\gamma_\pi^R(M(G_N, B'), B') \geq ((n - 1)k + 1)(l + 1) = nkl + l + 1 + k(n - l - 1).$$

Because V_G is a minimum vertex cover, $n \geq l + 1$. It then follows that

$$\gamma_\pi^R(M(G_N, B'), B') > nkl.$$

□

We have shown that the restricted $GF(q)$ π deletion problem is NP-hard for $GF(q)$ representable matroids when given a $GF(q)$ representation of the matroid. As there is a close relationship between matroids and graphs (and in particular, the elements of a matroid and the edges of a graph) this leads to the question: can this be used to show that the π edge deletion problem for graphs is NP-hard for an infinite family of properties π ? That is, if we restrict to graphic matroids, does the reduction hold? It was asked by Yannakakis if there exists a well-defined natural infinite family of properties for which the π edge deletion problem is NP-hard [80]. Lamentably when we restrict to graphic matroids, the reduction in Theorem 2.2.6 does not work. Unless the original graph G for which we are looking for a vertex cover is a forest, the resulting graph G' seems to never represent a graphic matroid (if G is a forest, then G' always represents a graphic matroid). Whether or not G' will always be non-graphic is unknown but it is definitely not the case that G' is always graphic. However, using similar methods to this and those used by Yannakakis, we can obtain a similar result for edge deletion problems and thus graphic matroids. This will be done in Chapter 3.

Now that we have shown that the restricted $GF(q)$ π deletion problem on $GF(q)$ representable matroids is NP-hard, we can move on to the general $GF(q)$ π deletion problem. We will make use of the following lemma in our reduction from

the restricted $GF(q)$ π deletion problem to the general $GF(q)$ π deletion problem.

Lemma 2.2.7. *Let π be a hereditary non-trivial $GF(q)$ matroid property that is completely closed under parallel extensions. Suppose that $A = [I|C]$ is a $GF(q)$ representation of a matroid M with the columns of I labelled by some basis B and let M' be the matroid created by adding a parallel element to each element of B . Then $\gamma_\pi^R(M, B) = \gamma_\pi^R(M', B)$.*

Proof. Note that we can construct A' such that $M' = M[A']$ in polynomial time. Suppose that F is a solution to the restricted $GF(q)$ π deletion problem on M . Let $A' = [I|I'C]$ where $I = I'$, the columns of I are labelled by B and the columns of I' are labelled by B' . The solution F will only involve deleting elements that are labels of columns from C . The resulting matroid, denoted $M[I|(C - F)]$, will satisfy π . Because we can add parallel elements to elements in B without violating π , we can construct the representation $[I|I'(C - F)]$ without violating π . This shows that F is a solution to the restricted $GF(q)$ π deletion problem on M' . Therefore $\gamma_\pi^R(M, B) \geq \gamma_\pi^R(M', B)$.

Now suppose F is a solution to the restricted $GF(q)$ π deletion problem for M' . If an element from B' was deleted so that the resulting matroid satisfies π , then we can add it back in (it is parallel to an element in the basis) without violating π . Therefore without loss of generality we can assume that the solution for M' will only remove elements from C . This will give the matroid represented by $[I|I'(C - F)]$. Because π is hereditary, we can delete all the elements that are labels of I' to obtain the matroid $M[I|(C - F)]$. Thus F is a solution to the restricted $GF(q)$ π deletion property on M with respect to basis B , completing the proof. $\square$

Theorem 2.2.8. *Let π be a hereditary non-trivial $GF(q)$ representable matroid property that is completely closed under parallel and series extensions. Then the general $GF(q)$ π deletion problem is NP-hard.*

Proof. This will be shown by reducing the restricted $GF(q)$ π deletion problem to the general $GF(q)$ π deletion problem. Suppose M is an instance of the restricted $GF(q)$ π deletion problem. Let M be represented by the $GF(q)$ matrix $A = [I|C]$ with the columns of I labelled by some basis B . Also, let M' be the matroid whose representation is obtained by adding $m = |E - B|$ parallel elements to each element in the basis B . Note that we can construct M' in polynomial time. Furthermore, any solution to the restricted problem on M' is a solution to the

general problem on M' . By Lemma 2.2.7, $\gamma_\pi^R(M, B) = \gamma_\pi^R(M', B)$. This shows that a solution to the restricted $GF(q)$ π deletion problem on M with respect to B is also a solution to the general $GF(q)$ π deletion problem on M' . It follows that $\gamma_\pi^R(M, B) \geq \gamma_\pi(M')$.

Let M' be as above and suppose that we have a minimal solution F to the general problem on M' and that this solution involves deleting one of the elements in the parallel classes containing an element from B . Because adding in parallel elements does not violate π , this solution would need to delete all $m + 1$ elements in the parallel class. Otherwise they could be added back in without violating π . However, if we delete all m elements in $E(M) - B$, then we will be left with a matroid isomorphic to the free matroid with each element replaced by $m + 1$ parallel elements. By Lemma 2.2.1, the free matroid satisfies π . We also know that adding parallel elements does not violate π . Thus replacing the elements of the free matroid with a parallel class gives a matroid that satisfies π . Therefore the minimum solution to the general problem on M' deletes only elements from $E(M) - B$ and is therefore a solution to the restricted problem on M' . By Lemma 2.2.7 again we see that this is also a solution to the restricted problem on M . This shows that $\gamma_\pi^R(M, B) \leq \gamma_\pi(M')$ and it follows that $\gamma_\pi^R(M, B) = \gamma_\pi(M')$. Therefore because the restricted $GF(q)$ π deletion problem is NP-hard, the general $GF(q)$ π deletion problem is NP-hard. $\square$

Let M be a matroid with a $GF(q)$ representation $A = [I|C]$ with the columns of I labelled by some basis B . Recall that there is a direct correspondence between B and the rows of C . Suppose now that in the search for a matroid that satisfies property π we are allowed to delete columns and rows of C . Note that deleting column $c \in C$ still produces a representation of $M \setminus c$ while deleting row b produces a representation of M/b . We will refer to the problem of finding the minimum number of deletions of rows and columns to obtain a matroid that satisfies π as the $GF(q)$ π deletion-contraction problem.

THE $GF(q)$ π DELETION-CONTRACTION PROBLEM

INSTANCE: A $GF(q)$ representation $A = [I|C]$ of a matroid M and an integer k .
 QUESTION: Can we delete at most k columns from C and rows from A to obtain a matrix representation of a matroid M' that satisfies π ?

Denote the size of a minimum solution by $\gamma_\pi^C(M)$ for the matroid M .

Theorem 2.2.9. *Let π be a hereditary non-trivial $GF(q)$ matroid property that is completely closed under parallel and series extensions. Then the $GF(q)$ π deletion-contraction problem is NP-hard.*

Proof. Let G be any graph, $N = (\mathcal{E}(N), w, B, E(M) - B)$ be the graph provided by Lemma 2.2.2 and $s \in c(N)$ be in the vertex set $E(M) - B$ of N . Also, let N_0 be a fixed maximum sized connected component of N with respect to s and G' be the graph created by replacing each edge and vertex in G by N_0 and N' respectively using the construction from pp12-13. As N is not a star graph, there is at least one other vertex d in $E(M) - B_0$ so this construction is valid. Recall that $\oplus_k N$ violates π . Now let G_N be the graph constructed from nk disjoint copies of G' where $n = |V(G)|$. Let F be an optimal solution to the $GF(q)$ π deletion-contraction problem for $M(G_N, B_N)$ where B_N is the set made from the bases of all the copies of G' . In at least $(n-1)k+1$ copies of G' every copy of the minimum graph N that corresponds to a matroid that violates π is going to need a deletion or a contraction. Otherwise the resulting graph $G_N - F$ will have at least k copies of N as a subgraph. Because π is hereditary, this would correspond to a $GF(q)$ matroid that violates π . Suppose that a copy of N has had a contraction. This will correspond to deleting an element from B_N (the side of the bipartition that does not include the vertices from G). By the construction of G_N , this will only eliminate a single copy of N . If instead we delete one of the vertices from G which correspond to s or d in the same copy of N , we will still have a graph that does not contain a subgraph isomorphic to N . Let F' be the set F with deletion or contraction of elements not corresponding to s or d replaced with either deleting s or d from the same copy of N . Then $|F'| \leq |F|$.

It now follows from an argument similar to the one used in Theorem 2.2.6 that $\gamma_\pi^C(M(G_N, B')) \leq nkl$ if and only if $|V_G| \leq l$ where V_G is a minimum vertex cover of G . Thus the $GF(q)$ π deletion-contraction problem is NP-hard. $\square$

2.3 Other Matroid Inputs

We have seen that given a matrix over $GF(q)$, finding the minimum number of elements we can delete to obtain a $GF(q)$ representable matroid that satisfies a non-trivial hereditary property π is NP-hard. What if we are not given a $GF(q)$ representation of a matroid but some other input? This is not uncommon as there are many different ways of describing matroids. In

this section we consider a related problem where the input is a list of the independent sets or some similar family of subsets of the matroid. Here the size of the description is the size of the listed subsets. This gives the following problem.

THE π DELETION PROBLEM WITH INDEPENDENT SETS

INSTANCE: An integer k and a matroid M described by listing the independent subsets of the matroid's ground set.

QUESTION: Can we delete at most k elements from M to obtain a matroid M' that satisfies π ?

If we can show that this problem is NP-hard, then we can use the input hierarchy created by Mayhew (and extended in Chapter 5) to extend these results to a whole range of methods for describing matroids by listing a family of subsets [55]. To show this problem is NP-hard we will construct a specific class of matroids.

We will associate a rank 3 matroid M_G with a graph G as follows. Take a positive integer edge weighted graph G with no parallel edges and any number of loops. Suppose G has vertex set $\{v_1, \dots, v_n\}$ and edge set $\{e_1, \dots, e_m\}$ with weights $\{w(e_1), \dots, w(e_m)\}$ respectively. The ground set of the matroid M_G will be

$$E(M_G) = \{v_1, \dots, v_n, e_1^1, \dots, e_1^{w(e_1)}, \dots, e_m^1, \dots, e_m^{w(e_m)}\}.$$

The set of circuits of M_G will be $\mathcal{C} = C_1 \cup C_2 \cup C_3$ where

1. $C_1 = \{v_i, v_j, e_k^l\}$ for $k \in \{1, \dots, m\}$, $l \in \{1, \dots, w(e_k)\}$, $i \neq j$, $i, j \in \{1, \dots, n\}$ where edge e_k is incident with vertices v_i and v_j ,
2. $C_2 = \{e_k^f, e_k^g\}$ where e_k is an edge of G and $f \neq g$, $f, g \in \{1, \dots, w(e_k)\}$,
3. C_3 is any set of four elements that do not contain a set in either C_1 or C_2 .

This creates a unique rank 3 matroid M_G for each positive integer edge weighted graph G . We will call the class of matroids that can be represented by a graph in this fashion *supine* and denote the graph associated with matroid M_G by G_M . We can create a geometric representation for M_G from G_M by the following procedure.

1. Replace each non-loop edge with a rank 2 flat by labelling the vertices incident with this edge as elements of the ground set.
2. For each non-loop edge e (that has been transformed into a rank 2 flat), replace it with the parallel class $\{e_1, \dots, e_{w(e)}\}$. Place this class on the line

so that it is collinear with the elements created in step 1 from the vertices incident with e .

3. For each loop l add in a parallel class of $w(l)$ elements freely in the plane.
4. Finally, throw away any vertices that are only incident with loops.

For an example see Figure 2.4. Note that there is not always a unique graph for each matroid in the class. However, minus loops (in the graph) every matroid in the class has a unique graph.

Figure 2.4: An edge weighted graph and its corresponding supine matroid.

The class of supine matroids is hereditary. Delete any element of a matroid in this class and there will still be a graph unique up to placement of loops that will give the geometric representation for the resulting matroid. Note that when we delete a vertex v that is incident with a non-loop edge, to represent the resulting matroid with a graph, we need to add in a loop with the same weight for each edge incident with v . Furthermore, $M \setminus e_k^l$ corresponds to G_M with $w(e_k) = w(e_k) - 1$ if e_k is an edge of G_M and $w(e_k) \neq 1$. If $w(e_k) = 1$, then we simply delete the edge e_k . We will denote the size of a minimal solution to the π deletion problem with listed subsets on the matroid M by $\gamma_\pi^l(M)$.

Lemma 2.3.1. *Let π be a hereditary non-trivial supine matroid property that is satisfied by $U_{3,n}$ for all n and completely closed under adding parallel elements. Then the π deletion problem with independent sets is NP-hard.*

Proof. This will be done using a reduction from the vertex cover problem similar to that of Theorem 2.2.6. Recall that for a graph N , $c(N)$ is the set of vertices whose removal produces graphs that are minimum under the order $<_\alpha$. For any

supine matroid M , placement of the loops will not affect $\alpha(G_M)$ as long as every loop of G_M is incident with a vertex that is incident with a non-loop edge. We want to find a graph N that is the lexicographically smallest connected graph G_M , such that repeating N k or more times corresponds to a matroid that violates π . Suppose that no connected G_M violates π . Then take any G_M that corresponds to a matroid that violates π and add rank 2 flats containing parallel classes until its corresponding graph is connected. This will satisfy π and because π is hereditary our initial matroid that violated π must satisfy π . Hence a contradiction is obtained and we can find a minimum connected graph N such that repeating N k or more times produces a graph whose corresponding matroid violates π . Because π is satisfied by $U_{3,n}$ and is completely closed under adding parallel elements, N must have at least two vertices joined with an edge. Because of this we can fix some $s \in c(N)$. We can also place all loops on s so they will not affect $\alpha(N)$. Furthermore we can always choose a vertex $d \neq s$ that is in the largest component of N with respect to s . Note that because N need not be bipartite, we do not have the restriction that d is in the same side of the bipartition as s . Let N_0 and N' be as in Theorem 2.2.6, G be any graph and construct the graph M_G from G as in Theorem 2.2.6 by replacing edges of G with N_0 , vertices with N' and then creating nk disjoint copies. Let V_G be a minimum vertex cover for G . We will show that $|V_G| \leq l$ if and only if $\gamma_\pi^l(M_G) \leq nkl$.

Let V_G be a vertex cover for G such that $|V_G| \leq l$. Delete these vertices from each copy of G' to obtain the graph M'_G . Construct the graph G_j as in Theorem 2.2.6 with the addition of j loops attached to the vertex s . Give these loops weighting m where m is the maximum weighting of an edge in N . Every connected component of M'_G will correspond to a submatroid of the matroid represented by G_j . Because adding any number of loops doesn't change the order $\leq_\alpha$, G_j is lexicographically smaller than N . Therefore by our hypothesis, G_j can be repeated arbitrarily many times without violating π . Because π is also hereditary, M'_G must satisfy π and therefore $\gamma_\pi(M_G) \leq nkl$.

Suppose $|V_G| \geq l+1$ and let F be a solution to $\gamma_\pi(M_G)$. In the graph M'_G there can be no more than $k-1$ copies of N . Therefore at least $nk - (k-1) = (n-1)k+1$ of the copies of G' in M'_G cannot have N as a subgraph. Let G'' be a copy of G' without N as a subgraph. For each element in F that does not correspond to s or d from G'' , replace it with either s or d to obtain the set F' . Observe that $|F'| \leq |F|$. Moreover, deleting F' from G' still gives a graph that does not contain

N as a subgraph. Because there is no copy of N as a subgraph in G'' , if we delete these vertices from G we will have a collection of isolated vertices. Therefore these correspond to a vertex cover of G . It follows that there are at least $l + 1$ elements in F' that are from G'' . From this we see that

$$\gamma_{\pi}^l(M_G) \geq ((n - 1)k + 1)(l + 1) = nkl + l + 1 + k(n - l - 1)$$

and therefore $\gamma_{\pi}^l(M_G) \geq nkl$. For any supine matroid, we can generate a list of its independent sets in polynomial time from a description of the corresponding graph. This is because there will be no more than $|E(M_G)|^3$ independent sets and each independent set will have no more than 3 elements. Therefore the π deletion problem with listed subsets is NP-hard when listing independent sets. $\square$

Note that if our property π is not satisfied by $U_{3,n}$ for all n , then minus parallel elements, there are only a finite number of supine matroids that satisfy π . Because π is completely closed under parallel extensions, we can essentially ignore these parallel elements. When we do this, there will no longer be an infinite number of matroids that satisfy π . This means that the π deletion problem could be solved in polynomial time.

We have shown that the π deletion problem with listed subsets is NP-hard when given the list of independent sets. However, there are many alternative ways to describe a matroid by listing subsets. For example, listing its bases or its circuits. This means that there are many different methods of input we can use to describe a matroid. However, as Theorem 2.3.2 shows, the π deletion problem with listed subsets is NP-hard for many natural families of subsets.

Theorem 2.3.2. *Suppose we are given a matroid described by any of the following methods: independent sets, flats, bases, circuits, hyperplanes, nonspanning circuits, cyclic flats, dependent hyperplanes, connected cyclic flats and circuit closures. Then if π is a non-trivial hereditary matroid property that is satisfied by $U_{3,n}$ for all n and completely closed under adding parallel elements, then the π deletion problem with listed subsets is NP-hard.*

Proof. From Lemma 2.3.1 we see that when given independent sets the π deletion problem with listed subsets is NP-hard. Using the ordering of inputs in [55] it follows that it is NP-hard for inputs flats, bases, circuits, hyperplanes, nonspanning circuits, cyclic flats and dependent hyperplanes.

Now the results in Chapter 5 show that it is also NP-hard when the matroid is described by listing its connected cyclic flats or circuit closures. $\square$

2.4 Oracle Methods

The final type of matroid complexity we will consider in this chapter is that of a Turing machine equipped with an oracle. In particular, we assume that our Turing machine has a black box subroutine that returns in unit time a yes or no answer or a piece of information about a subset. In the case of matroids the routine will often return whether a set is independent or the rank of a set. In this section our oracle will be an independence oracle which will return YES if a given set is independent and NO if the given set is dependent. We will define our Turing machine with oracle to be a Turing machine with an additional tape and oracle. If the Turing machine wants to question the oracle about a subset, it just writes this set onto the oracle tape. The oracle will read this tape and reply in unit time whether the set is independent or not. If we are describing our object with an oracle, then we cannot obtain NP-hardness results. This is because when we describe our object with an oracle, we have not input size. However, describing our objects with oracles can be used to give a lower bound on the worst case running times for Turing machines that take the same matroid input. Because of this, we will define a *polynomial oracle algorithm* to be an algorithm that runs on an oracle Turing machine in polynomial time. The Turing machine we will consider will take as input a string of n ones where $|E(M)| = n$.

Let P_1 and P_2 be two problems. We will say that P_1 is polynomially reducible to P_2 if given an oracle for P_2 , there exists a polynomial oracle algorithm to solve P_1 . We will show that for a large class of matroids and properties π , if the π deletion problem is not polynomially reducible to deciding if a matroid satisfies π , then there does not exist a polynomial oracle algorithm for solving the π deletion problem with an independence oracle. We will say an oracle $\mathcal{O}$ *simulates* oracle $\mathcal{O}'$ if we can calculate the answer given by oracle $\mathcal{O}'$ in polynomial time using a polynomial number of calls to oracle $\mathcal{O}$. Independence oracles can simulate most oracles for matroids [63]. Because of this, the results in this section can be easily extended to most oracles for matroids.

Lemma 2.4.1. *If a Turing machine with independence oracle can solve a problem in polynomial time, then so can a deterministic Turing machine when given the*

list of independent sets of a matroid as input.

Proof. Let $\mathcal{T}$ be an oracle Turing machine that can solve the problem in question. Take $\mathcal{T}$ and replace any call to the oracle with a search through the list of independent sets given to $\mathcal{T}$ as input. Each search can be done in time that is polynomial in the size of the list of independent sets. As the rest of the algorithm runs in polynomial time, we can solve the matroid problem in polynomial time when the matroid is described by a list of its independent sets. $\square$

From the previous section and Lemma 2.4.1, it follows that there *probably* is no polynomial oracle algorithm with independence oracle for the π deletion problem. However, this relies on the assumption that $P \neq NP$. Deterministic Turing machines with the list of independent sets as input are at least as powerful as Turing machines with independence oracles. Thus, it does not necessarily follow that if $P = NP$, then there exists a polynomial oracle algorithm with independence oracle to solve the π deletion problem.

Let π be a matroid property such that whenever the matroids $M_1, \dots, M_m$ satisfy π , then so does the matroid $M = M_1 \oplus \dots \oplus M_m$. Then we say that π is *defined by its connected components*. Note that if a property is hereditary and defined by its connected components, then $M = M_1 \oplus \dots \oplus M_m$ satisfies π if and only if all of $M_1, \dots, M_m$ satisfy π .

A matroid is a *sparse paving matroid* if all its non-spanning circuits are hyperplanes. There is evidence that the majority of matroids are sparse paving matroids [56]. Due to this, it is not uncommon to focus on sparse paving matroids. For example, see [43]. Given a sparse paving matroid M we can perform a *circuit-hyperplane relaxation* by changing the circuit hyperplane C to a basis. Similarly if for a basis B , $B \cup e$ is a circuit for all $e \notin B$ then we can perform a *basis tightening* of B which will change B to a circuit. A circuit-hyperplane relaxation or a basis tightening will produce a new sparse paving matroid. We will show that for hereditary properties that are defined by their connected components and non-trivial on sparse paving matroids there does not exist a polynomial oracle algorithm with independence oracle for solving the π deletion problem. Alarm bells might be going off for the careful reader as the property of being a sparse paving matroid is not defined by its connected components. However, the reduction in Theorem 2.4.4 does not require that we restrict to only sparse paving matroids. We only require that the property π is non-trivial on sparse paving matroids.

The π recognition problem is given a property π and matroid M , decide whether or not M satisfies π . We note that the π deletion problem is at least as hard as the π recognition problem. To see this, assume we have an algorithm $\mathcal{A}$ that can solve the π deletion problem. Then given an instance of the π recognition problem, we can simply apply $\mathcal{A}$ to it with $k = 0$, where k is the number of allowed deletions. So if there is no polynomial oracle Turing machine that can solve the π recognition problem, then it follows that there is no polynomial oracle Turing machine that can solve the π deletion problem. It has been shown that for a number of non-trivial hereditary properties π there does not exist a polynomial Turing machine with independence oracle that can determine whether a matroid M satisfies π . Some examples of these are being binary [64], representable [65] or transversal [75]. Thus it follows that there is no polynomial Turing machine with independence oracle that can solve the π deletion problem for the properties binary, representable or transversal.

If the π deletion problem is only as hard as the π recognition problem, then the π deletion problem is not terribly interesting (and possibly easy to solve). As Lemma 2.4.2 shows, if π is completely closed under circuit-hyperplane relaxations, then the π deletion problem is not very interesting on sparse paving matroids.

Lemma 2.4.2. *Let π be a hereditary matroid property that is non-trivial on the class of sparse paving matroids and is defined by its connected components. If π is completely closed under circuit-hyperplane relaxations, then the π deletion problem on sparse paving matroids is polynomially reducible to the π recognition problem for the matroids $U_{\min(r, n-k), n-k}$, where r is the rank of the given matroid, n is the size of the given matroid's ground set, k is the number of allowed deletions and the uniform matroid is described by an independence oracle.*

Proof. Because π is defined by its connected components, we need only consider connected matroids. To prove this, we will show that if π is completely closed under circuit-hyperplane relaxations, then a sparse paving matroid with ground set $|E(M)| = n$ and rank r satisfies π if and only if $U_{r, n}$ satisfies π . To see this, take any sparse paving matroid and perform circuit-hyperplane relaxations until we have the uniform matroid $U_{r, n}$. Because π is completely closed under circuit-hyperplane relaxations, this will satisfy π if and only if the original matroid satisfied π . Now to see if a matroid M is k or fewer deletions away from satisfying π , we need only check whether or not the uniform matroid $U_{\min(r, n-k), n-k}$ satisfies π . This is because any matroid that is k deletions away from M can be transformed

into the uniform matroid $U_{\min(r, n-k), n-k}$ using circuit-hyperplane relaxations. $\square$

Let M and M' be two sparse paving matroids. We will say that M is *linked to* M' if we can obtain M' from M via a sequence of circuit-hyperplane relaxations or basis tightenings. It is straightforward to show that the property of being linked is an equivalence relation. Furthermore, let M and M' be two linked matroids that differ by only a single circuit-hyperplane relaxation such that one of M and M' satisfies π and the other violates π . Then we will call (M, M') a *linked pair* with respect to π . We will usually drop ‘with respect to π ’ when it does not result in any confusion. For consistency, when we talk about a linked pair (M, M') , the matroid M will always violate π while the matroid M' will satisfy π . We note an important property of linked pairs. Let π be a non-trivial on sparse paving matroids property that is hereditary and not completely closed under circuit-hyperplane relaxations. Let $\mathcal{M}$ be the subset of sparse paving matroids that violate π and are linked to matroids that satisfy π . Furthermore, let $\mathcal{M}'$ be the set of matroids that satisfy π and are linked to matroids in $\mathcal{M}$. Then it is straight forward to see that $|\mathcal{M}|$ is infinite if and only if $|\mathcal{M}'|$ is infinite. More importantly, if $|\mathcal{M}|$ is infinite, then there exist an infinite number of linked pairs.

Lemma 2.4.3. *Let $\mathcal{M}$ be a class of sparse paving matroids and π be a hereditary property that is non-trivial on $\mathcal{M}$ and is defined by its connected components. Furthermore, let $\mathcal{M}' \subseteq \mathcal{M}$ be the (possibly empty) set of matroids in $\mathcal{M}$ that are linked to a matroid in a linked pair. Then either*

- (i) *the π deletion problem on $\mathcal{M}$ is polynomially reducible to the π recognition problem, or*
- (ii) *for each integer c , there exists an infinite number of matroids $M \in \mathcal{M}'$ such that $|E(M)| - r(M) > c$.*

Proof. Suppose that (ii) does not hold. Then there exists a c such that $|E(M)| - r(M) \leq c$ for all $M \in \mathcal{M}'$. Because π is non-trivial and defined by its connected components it is satisfied by any free matroid $U_{n,n}$. Suppose we are given a matroid $M \in \mathcal{M}'$. Then deleting c elements from M will give a free matroid which will therefore satisfy π . We can check all possible sets of $\min\{k, c\}$ elements that we could delete from M . One of these will produce a matroid that satisfies π if and only if M is no more than k deletions away from satisfying π . As c is fixed this will require only a polynomial number of checks. Therefore the π deletion problem for matroids in $\mathcal{M}'$ is polynomially reducible to deciding whether or not a matroid

satisfies π . Now consider the class $\mathcal{M} \setminus \mathcal{M}'$. Circuit-hyperplane relaxations will not affect whether or not a matroid in this class satisfies π . From Lemma 2.4.2 we know that the π deletion problem on $\mathcal{M} \setminus \mathcal{M}'$ is polynomially reducible to the π recognition problem for matroids in $\mathcal{M} \setminus \mathcal{M}'$. Thus the π deletion problem on $\mathcal{M}$ is polynomially reducible to the π recognition problem if (ii) does not hold. $\square$

Theorem 2.4.4. *Let π be a hereditary matroid property that is non-trivial on the class of sparse paving matroids and is defined by its connected components. Then either (i) the π deletion problem on sparse paving matroids is polynomially reducible to the π recognition problem or (ii) there does not exist a Turing machine with a polynomial oracle algorithm with independence oracle for solving the π deletion problem.*

Proof. Note that if we relax the circuit hyperplane C for a matroid M and then delete $c \in C$ we obtain the matroid $M \setminus c$ [58], Prop 3.3.5. It follows that because π is hereditary, if a sparse paving matroid is one circuit-hyperplane relaxation away from satisfying π , it is also a single deletion away from satisfying π . Similarly, if we tighten a basis B and then delete $e \in B$, we obtain the matroid $M \setminus e$. Thus, if M is one basis tightening away from satisfying π , it is also one deletion away from satisfying π .

Throughout this proof, k will be some fixed integer that we are given and asked to decide if a matroid is k or less deletions away from satisfying π .

Assume we have a property π such that the π deletion problem is not polynomially reducible to the π recognition problem. Then by Lemmas 2.4.2 and 2.4.3 there exists an infinite number of linked pairs of sparse paving matroids (M, M') such that M violates π , M' satisfies π and there is no bound for their corank $(|E(M)| - r(M))$. Let $\mathcal{P}$ be the class of all sparse paving matroids in such a linked pair. As $\mathcal{P}$ is infinite, either there exists an infinite number of matroids in $\mathcal{P}$ that are one circuit-hyperplane relaxation away from satisfying π or an infinite number of matroids in $\mathcal{P}$ that are one basis tightening away from satisfying π . We will first deal with the case where there exists an infinite number of matroids $M \in \mathcal{P}$ that violate π and are one circuit-hyperplane relaxation away from satisfying π . Let $\mathcal{N}$ be the set of all matroids in $\mathcal{P}$ that are one circuit-hyperplane relaxation away from satisfying π or are in a linked pair with a matroid that is one circuit-hyperplane relaxation away from satisfying π .

To begin with, suppose that the number of circuit hyperplanes of matroids in $\mathcal{N}$ is bounded by no polynomial in the size of the ground set $E(M)$. For every

linked pair (M, M') such that $M, M' \in \mathcal{N}$, M violates π and M' satisfies π , let M_1 be the matroid made from the direct sum of $k+1$ copies of M and let M_2 be made from the direct sum of k copies of M and one copy of M' . Let $\mathcal{N}_k$ be the class of all matroids M_1 and M_2 created from linked pairs $(M, M') \in \mathcal{N}$. Note that because π is hereditary and defined by its connected components, a matroid M_1 is $k+1$ deletions away from satisfying π and M_2 is only k deletions away. To know which matroid we have, we need to be able to tell the difference between M and M' . Now suppose we are given a matroid $M \in \mathcal{N}_k$ and asked if this is k deletions away from satisfying π . To answer this, we would need to be able to tell if the matroid in question is isomorphic to some M_1 or M_2 . To do this, we will have to question the oracle about every circuit hyperplane in each connected component of M_1 and see if it has been relaxed. As the number of circuit hyperplanes of matroids in $\mathcal{N}$ is bounded by no polynomial, in the worst case we require a non-polynomial number of calls to the oracle and therefore this cannot be done in polynomial time.

Now suppose that there is a polynomial p such that for every $M \in \mathcal{N}$, the number of circuit hyperplanes of M is at most $p(|E(M)|)$. Recall that for the linked pair (M, M') , the matroid M' satisfies π . From each matroid $M' \in \mathcal{N}$, we will create the matroid M'' by doing the following. Take M' and repeatedly tighten arbitrary bases as long as the resulting matroid satisfies π . Keep choosing an arbitrary basis and tighten it until there are no more bases left that can be tightened without violating π . This can be done in any order as all we require is that the resulting matroid M'' has no basis that can be tightened without violating π . Now let M''' be a matroid obtained from M'' by a single basis tightening. Note that each matroid M''' violates π , $|E(M''')| = |E(M'')| = |E(M)|$ and there is no bound on the size of $|E(M'')| - r(M'')$ for matroids M'' as rank cannot decrease as a result of a basis tightening or circuit-hyperplane relaxation. Let $\mathcal{N}' \subseteq \mathcal{N}$ be the class of matroids containing the pairs (M''', M'') made from matroids $M, M' \in \mathcal{N}$ such that (M, M') is a linked pair. Note that the number of circuit hyperplanes of matroids in the class $\mathcal{N}'$ is bounded by some polynomial $p(|E(M)|)$. Otherwise there would be no polynomial bound on the number of circuit hyperplanes of matroids in $\mathcal{N}$.

For each matroid $M'' \in \mathcal{N}'$ there are $\binom{|E(M'')|}{r(M'')}$ sets of size $r(M'')$ and at most $p(|E(M'')|)$ of them are circuit-hyperplanes. If for some basis B , $B \cup e$ is not a circuit, then $B \cup e$ must contain a circuit. This is not possible if $|B \cap C| \leq |C| - 2 = r(M'') - 2$ for all circuit hyperplanes C . Thus, a basis B can be tightened if and only

if $|B \cap C| \leq r(M'') - 2$ for all circuit hyperplanes C . For each circuit hyperplane C , there are at most

$$r(M'') \cdot (|E(M)| - r(M'')) + 1 < r(M'') \cdot |E(M)|$$

sets X of size $r(M'')$ such that $|X \cap C| > r(M'') - 2$. Therefore in each matroid M'' , there are at least

$$\binom{|E(M'')|}{r(M'')} - r(M'') \cdot |E(M'')| \cdot p(|E(M'')|)$$

bases that can be tightened. As there is no bound on the corank of matroids in $\mathcal{N}'$, there is no polynomial that bounds the above function. So suppose we are given matroids $M'', M''' \in \mathcal{N}'$ that were created from a linked pair $(M, M') \in \mathcal{N}$. Then to tell the difference between M'' and M''' , an independence oracle will need to be questioned about all of the bases. As the number of bases that can be tightened in matroids in $\mathcal{N}'$ grows non-polynomially as the ground set of the matroids grows, this cannot be done in polynomial time. Now let M_1 be made from the direct sum of $k+1$ copies of M''' and M_2 be made from the direct sum of k copies of M''' and one copy of M'' . Note that M_1 is $k+1$ deletions away from satisfying π while M_2 is only k deletions away. Let $\mathcal{N}_k$ be the class of matroids M_1 and M_2 created from the matroids in $\mathcal{N}'$. When given a matroid from $\mathcal{N}_k$ and asked if it is k or less deletions away from satisfying π , to decide this we will have to be able to decide if it is of the form M_1 or M_2 . To do this, we would need to check every basis of each connected component and see if it has been tightened. However, as we have shown above, there is no polynomial bound on the number of bases that can be tightened in matroids in $\mathcal{N}_k$. Thus we cannot tell the difference between M_1 and M_2 in polynomial time. Therefore in this case, there does not exist a polynomial oracle algorithm with independence oracle for solving the π deletion problem.

Now suppose we only have an infinite family $\mathcal{N}$ of linked pairs (M', M) such that M is one circuit-hyperplane relaxation away from violating π and M' violates π . Recall that this implies that M is one deletion away from satisfying π . If there is no polynomial bound on the number of circuit hyperplanes of the matroids in $\mathcal{N}$, then it follows from above that there does not exist a polynomial oracle algorithm with independence oracle that can tell the difference between M and M' .

Alternatively, suppose that there is a polynomial bound on the number of circuit hyperplanes of matroids in $\mathcal{N}$. Then by the above argument, for each

linked pair (M', M) we can construct matroids M''' and M'' such that M'' violates π , M''' satisfies π and to tell the difference between the two could require a non-polynomial number of calls to an independence oracle. Again we see that in this case there does not exist a polynomial oracle algorithm with independence oracle for solving the π deletion problem. $\square$

Corollary 2.4.5. *Let π be a hereditary matroid property that is non-trivial on the class of sparse paving matroids and is defined by its connected components. Furthermore, suppose that the π deletion problem on sparse paving matroids is not polynomially reducible to the π recognition problem. Then there is no polynomial oracle algorithm with independence oracle for solving the π recognition problem.*

An example of a property that is hard to decide on sparse paving matroids is the property of being the direct sum of uniform matroids. Note that this property satisfies all the conditions of Theorem 2.4.4. This can be easily seen to be hard by taking the direct sum of some uniform matroids and tightening a basis in one of the uniform matroids. For a Turing machine with independence oracle to tell the difference, it would need to ask the oracle about every basis of each uniform matroid in the direct sum. This cannot be done in polynomial time. Thus the property of being a direct sum of uniform matroids is hard for sparse paving matroids.

Chapter 3

Edge Deletion

3.1 Introduction

There is a direct correspondence between edges of a graph and elements of a matroid. So given that the matroid deletion problem is NP-hard, it is only natural to ask if the same holds for the edge deletion problem. However, as was mentioned in Chapter 2, the reduction used to show that the matroid deletion problem is NP-hard does not work when restricted to graphic matroids. This may not come as a surprise as the techniques used on the matroid deletion problem were very similar to those used by Yannakakis to show that the vertex deletion problem is NP-hard. While these techniques worked very well for the vertex deletion problem, they appeared not to gain any traction with edge deletion problems. Yannakakis asked whether or not there are any well-defined natural infinite families of graph properties for which the π edge deletion problem is NP-hard. Since then most of the results have been for specific edge deletion problems and not infinite families of problems. For example see [17, 37, 57]. However, there have been a few results on the complexity of edge deletion problems for families of non-trivial hereditary properties. The following two results are known.

Theorem 3.1.1. *If π is non-trivial, hereditary and satisfied by all bipartite graphs, then for any $\delta > 0$, it is NP-hard to approximate the edge deletion problem to within an additive error of $n^{2-\delta}$ (see [3]).*

Theorem 3.1.2. *If π is non-trivial, hereditary and defined by its 3-connected components, then the π edge deletion problem is NP-hard (see [8, 9]).*

Theorem 3.1.2 was a strengthening of an earlier result where π was finitely

characterizable by 3-connected graphs [77, 78].

Apart from these results, little more is known. We present a new less restrictive infinite family of graph properties for which the π edge deletion problem is NP-hard. Moreover, the techniques used to show the NP-hardness of this family are very similar to the original techniques used by Yannakakis for vertex deletion problems, showing just how powerful these techniques are. Note that if the graph property π we are interested in can be verified in polynomial time, then its corresponding π edge deletion problem is NP-complete as opposed to NP-hard.

THE π EDGE DELETION PROBLEM

INSTANCE: A graph G and integer k .

QUESTION: Can we delete at most k edges from $\mathcal{E}(G)$ to obtain a graph that satisfies π ?

It will be shown that this problem is NP-hard for an infinite class of properties π . To do this, we will first show that a similar problem is NP-hard. Suppose that when we are deleting edges, we are only allowed to delete certain edges. This gives the following problem.

THE π EDGE DELETION PROBLEM WITH SELECTED EDGES

INSTANCE: A graph G , an integer k and a set of edges $\mathcal{E}' \subseteq \mathcal{E}(G)$.

QUESTION: Can we delete at most k edges from $\mathcal{E}'$ to obtain a graph that satisfies π ?

We begin with some preliminary definitions. Note that the proofs in this chapter will use similar ideas to those in Chapter 2. However, in Chapter 2 we were interested in matroids, while we are now interested in graphs. Because of this, a number of the key definitions will seem very similar, but will be slightly different as we are now working with different objects. So a sense of déjà vu may be expected. We define a *graph property* π to be a class of graphs that is closed under isomorphism. We will often refer to a graph in the class π as satisfying π and a graph not in the class as violating π . Such a property is *hereditary* if for any graph G that satisfies π , any subgraph $G' \subseteq G$ also satisfies π . We will define a property π to be *non-trivial* if (i) it is satisfied by any collection of isolated vertices and (ii) there exist graphs that violate π . Note that property (i) implies that there is an infinite number of graphs that satisfy π . If a graph property π is hereditary and

only satisfied by a finite number of graphs, then the π edge deletion problem is trivially in P. This is because there will be a maximum sized graph that satisfies π . To see if a graph is k or less deletions away from satisfying π , we need only check all subgraphs of a fixed size. Therefore to get any meaningful complexity results, we require π to be non-trivial.

Let G be a graph with two or more connected components. Choose two connected components and join them with a single edge. We say the resulting graph has been obtained from G by a *series composition*. Suppose for a pair of edges e and e' , e is in a cycle if and only if e' is in the cycle. Then e and e' are *in series*. A *series extension* of an edge e is adding an edge e' in series with e . Note that our definition of a series extension is slightly different to the norm. We take our definition of a series extension from matroid theory. So if an edge e is in a cycle if and only if another edge e' is in the cycle then e' is a series extension of e . That is, e' is a series extension of e if and only if every cycle that contains e also contains e' and vice versa. This definition includes the usual definition of series extension but allows for more variety. Performing our version of a series extension on a graph will give exactly the same cycles as performing the usual version of a series extension on the same graph. For an example of the difference in definitions of series extensions, see Figure 3.1. By our definition, the edges e_2 and e_3 are both series extensions of e_1 . However, by the usual definition, only the edge e_2 is a series extension of e_1 .

Figure 3.1: Example of the difference in series extensions.

Two edges $e = \{u, v\}$ and $e' = \{w, x\}$ are *parallel* if $\{e, e'\}$ is a cycle. A *parallel extension* of e is adding an edge e' in parallel with e . Let m be some edge modification. For example m could be adding parallel edges, adding edges in series or a series composition. We say that π is *completely closed under m* if, whether or not a graph satisfies π is not changed by performing edge modification m . Define a graph property π to be *malleable* if it is completely closed under series and parallel extensions and series compositions. For an example, the graph property of being

planar is malleable.

The family of properties we are interested in will be all properties that are non-trivial, hereditary and malleable. We note that this gives us the following additional property for free.

Lemma 3.1.3. *Let π be a non-trivial hereditary graph property that is completely closed under series and parallel extensions. Then π is completely closed under addition/deletion of isolated vertices.*

Proof. Suppose that this is not the case. Then there exists graphs G and $G' = G \cup v$ where v is an isolated vertex such that one of G and G' satisfies π and the other one violates π . Because π is hereditary, if G' satisfies π then so does G . So assume that G satisfies π and G' violates π . Because π is non-trivial, any collection of isolated vertices satisfies π . Thus we can assume that G (and G') have at least one edge e . We can obtain the graph G' from G using the following edge modifications: (i) add an edge e' in parallel with e , (ii) then subdivide the edge e' to add an edge e'' in series with e' and (iii) delete e' and e'' . This will produce a graph isomorphic to G' . As π is hereditary and completely closed under series and parallel extensions, the resulting graph G' must also satisfy π . Hence a contradiction and π is completely closed under deletion of isolated vertices. $\square$

3.2 Preliminary Constructions

This section contains a number of basic constructions and results that are used throughout the remainder of the chapter.

Lemma 3.2.1. *Let G be a graph and let G^1 be made from subdividing each edge of G twice. Then G has a vertex cover of size k if and only if G^1 has a vertex cover of size $k + |\mathcal{E}(G)|$.*

Proof. Let G be a graph and create the graph G^1 by subdividing each edge $e \in \mathcal{E}(G)$ twice. Suppose F is a vertex cover of G of size at most k . Each vertex in G corresponds to a vertex in G^1 . Because F is a vertex cover, each edge in G is incident to a vertex in F . Consider the three edges e_1, e_2 and e_3 in $\mathcal{E}(G^1)$ that were created by subdividing the edge $e \in \mathcal{E}(G)$. As F is a vertex cover of G , e is incident to a vertex of F . This means that at least one of the three edges e_1, e_2 or e_3 are incident to a vertex in F and the other two are incident with a common

vertex v . Without loss of generality, we will assume that e_1 is incident to a vertex in F . To cover the edges e_2 and e_3 we need only choose the vertex v that is incident to both e_2 and e_3 . This gives a vertex cover for G^1 of size no greater than $k + |\mathcal{E}|$.

Now suppose that G^1 has a vertex cover F' of size no greater than $k + |\mathcal{E}(G)|$. We can assume without loss of generality that $e = (v_1, v_2) \in \mathcal{E}(G)$ has been replaced with $e_1 = (v_1, v_3)$, $e_2 = (v_3, v_4)$ and $e_3 = (v_4, v_2)$. Suppose that both the vertices v_3 and v_4 are in F' . Then let $F'' = (F' \setminus \{v_4\}) \cup v_2$. This is still a vertex cover of G^1 . Thus without loss of generality, we can assume that F' contains exactly $|\mathcal{E}(G)|$ vertices that do not correspond to vertices in G . It follows that G has a vertex cover of size no greater than $|F'| - |\mathcal{E}(G)| = k$. $\square$

A connected graph G is *2-connected* if we cannot delete a single vertex to disconnect G . That is, $G \setminus \{v\}$ is connected for all $v \in V(G)$. A maximal 2-connected component of a graph G is a maximal vertex induced subgraph of G that is 2-connected.

Take two vectors $A = (a_1, a_2, \dots, a_r)$ and $B = (b_1, b_2, \dots, b_t)$. Recall from Chapter 2 that $A <_l B$ (lexicographically smaller than) if (i) $a_1 < b_1$ or (ii) $a_1 = b_1, \dots, a_i = b_i$ and $a_{i+1} < b_{i+1}$ or (iii) $r < t$ and $a_i = b_i$ for all $i \in \{1, \dots, r\}$. Also recall that $\oplus_k G$ is the graph made from k disjoint copies of G .

Let G be a graph and let $e \in \mathcal{E}(G)$. Define $\alpha_e(G) = (|\mathcal{E}(G_1)|, \dots, |\mathcal{E}(G_n)|)$ where G_i is a maximal 2-connected component of $G \setminus \{e\}$ and $|\mathcal{E}(G_i)| \geq |\mathcal{E}(G_j)|$ for $i < j$. Define $c(G)$ to be the set of edges of G that give the lexicographic minimal sequence and let $\alpha(G) = \alpha_e(G)$ where $e \in c(G)$. This gives a total ordering on all graphs represented by $<_\alpha$ where $G_1 <_\alpha G_2$ if $\alpha(G_1) <_l \alpha(G_2)$. Note that this is a very similar idea to that of $\alpha(G)$ from Chapter 2. However, the definition is slightly different as this time we are interested in the number of edges as opposed to the number of vertices.

The circuit elimination axiom is: if C_1 and C_2 are circuits of a matroid such that $e \in C_1 \cap C_2$, then there exists a circuit $C_3 \subseteq (C_1 \cup C_2) - \{e\}$. This also applies to cycles in a graph due to the relationship between graphs and matroids.

Lemma 3.2.2. *Let π be a non-trivial hereditary graph property that is malleable. Then there exists a 2-connected graph N such that*

1. N has more than two cycles;
2. there exists some natural number k such that the graph $\oplus_k N$ violates π ; and

3. for any 2-connected graph $G <_\alpha N$ and natural number j , the graph $\oplus_j G$ satisfies π .

Proof. Suppose we have a 2-connected graph G with no more than two cycles. Then G cannot have exactly two cycles. This is because by the circuit elimination axiom, if a graph has two cycles with a common edge, then there must exist a third cycle. So if G has exactly two cycles, then there cannot be any edge in both cycles. But then G would not be 2-connected. Therefore, if G is 2-connected with no more than two cycles, then G must be either a single edge or a single cycle. Either of these can be obtained from the graph consisting of two isolated vertices from a series composition and possibly a parallel extension followed by series extensions. As π is non-trivial and completely closed under these operations, G must satisfy π . Thus every 2-connected graph with no more than two cycles satisfies π . As π is non-trivial, there exist graphs that violate π . Therefore as π is hereditary, there exists graphs that have more than two cycles such that k disjoint copies of them violate π . Take any graph N' such that $\oplus_k N'$ violates π . We can add edges to N' until we have a 2-connected graph. The graph made from k disjoint copies of this graph must also violate π as π is hereditary. This gives a 2-connected graph N such that $\oplus_k N$ violates π . Therefore there exists a 2-connected graph N that is minimal under $\leq_\alpha$ with the property that $\oplus_k N$ violates π for some $k \in \mathbb{N}$. $\square$

From now on we will assume that we have some fixed property π that is hereditary, non-trivial and malleable. Let N be a fixed 2-connected graph given by Lemma 3.2.2. Fix some $e_1 = (v, v') \in c(N)$. Let N^+ be a fixed maximum sized 2-connected component of $N \setminus e_1$ and $N^- = N - \mathcal{E}(N^+)$ minus any isolated vertices. Because N is 2-connected, there must exist vertices v_1 and v_2 that are in both N^+ and N^- . Moreover, as N^+ is a maximal sized 2-connected component of $N \setminus \{e_1\}$, there can be no more than 2 vertices that are in both N^+ and N^- . Note that if N is 3-connected, then N^- is just the edge e_1 and its incident vertices. Before we move on, we cover some basic properties of the graphs N^+ and N^- .

Lemma 3.2.3. *The graph N^+ is not a cycle*

Proof. Suppose that N^+ is a cycle. Recall that there are two vertices $v_1, v_2 \in V(N^+)$ that are also in N^- . Because π is completely closed under series and parallel extensions, in N we could contract this cycle down to a parallel class and then to a single edge $e = (v_1, v_2)$. If N_s is the resulting graph, then $\oplus_k N_s$ would still violate π . This would imply that N is not minimal with the property that $\oplus_k N$

violates π . This is because contracting a series extension cannot create maximal 2-connected components or increase the size of maximal 2-connected components of a graph. As one of the maximal 2-connected components of $N \setminus e_1$ has lost some edges we see that $N_s <_\alpha N$. Hence a contradiction and we can assume that N^+ is not a cycle. $\square$

Using a similar argument as Lemma 3.2.3, we can assume that N has no parallel edges or series extensions. Otherwise we could delete or contract these to obtain a lexicographically smaller 2-connected graph N' such that $\oplus_k N'$ violates π . This will contradict the fact that N is minimal with the property that $\oplus_k N$ violates π . Therefore we can assume that N is simple.

Lemma 3.2.4. *The graph N^+ has at least four vertices.*

Proof. From Lemma 3.2.2 we know that N has at least three cycles. We also know that N is simple. Moreover, by the circuit elimination axiom we know that there cannot be an edge that is in all cycles. Thus we cannot remove all cycles from N by deleting a single edge. Thus in $N \setminus e_1$, there exists a cycle. Any simple 2-connected graph without a cycle must just be a single edge or an isolated vertex. Therefore as N^+ is a maximal 2-connected component of $N \setminus e_1$ and $N \setminus e_1$ contains a cycle, N^+ must contain a cycle. Any simple 2-connected graph that contains a cycle must have at least three vertices. However, any simple graph that contains a cycle and only has three vertices is itself a cycle. By Lemma 3.2.3 N^+ is not just a cycle. Thus it must have another vertex. Therefore N^+ has at least four vertices. $\square$

Lemma 3.2.5. *There is no cycle in N^- that contains the edge e_1 .*

Proof. Suppose there is a cycle C^- in N^- that contains e_1 . Choose some edge $e^+ \in \mathcal{E}(N^+)$. As N is 2-connected, there exists a cycle C^+ containing e^+ and e_1 . Now by the circuit elimination axiom, there exists a cycle $C \subseteq (C^+ \cup C^-) - \{e_1\}$. Moreover, C must contain at least one edge from $\mathcal{E}(N^+)$ as C^- cannot contain a cycle. However, if this is the case, then the graph with edges $\mathcal{E}(N^+) \cup C$ would be 2-connected. As $e_1 \notin C$, this is a contradiction to the fact that N^+ is a maximal 2-connected component of $N \setminus \{e_1\}$. Therefore C^- cannot exist. $\square$

Lemma 3.2.6. *There exists an edge $e_2 = (v_3, v_4)$ of N^+ such that $v_3, v_4 \notin \{v_1, v_2\}$.*

Proof. From Lemma 3.2.4 we know that N^+ has at least four vertices. We also know that N^+ contains a cycle. If there is no such edge e_2 , then every edge of N^+ must be incident with v_1 or v_2 or both. There are at least two other vertices in N^+ and because N^+ is 2-connected, these must be adjacent to both v_1 and v_2 . As every edge is incident with v_1 or v_2 , all other vertices must have degree 2. But then N^+ and hence N have two edges in series. This is a contradiction as we know that N has no edges in series. Thus an edge $e_2 = (v_3, v_4)$ such that $v_3, v_4 \notin \{v_1, v_2\}$ exists. $\square$

Much like in Chapter 2, we wish to replace vertices and edges of given graphs with graphs derived from the graph N . We will show that the π edge deletion problem with selected edges is NP-hard with a reduction from the vertex cover problem on simple planar graphs with maximum vertex degree 3. This problem was shown to be NP-complete by Garey and Johnson [29]. Due to this, the majority of the following results will assume that we are replacing vertices and edges from a graph G that is a simple planar graph with maximum vertex degree 3. With this in mind, we will construct the following graphs.

Choose some fixed edge $e_2 = (v_3, v_4)$ of N^+ such that $v_3, v_4 \notin \{v_1, v_2\}$. Such an edge will always exist by Lemma 3.2.6. We will use this edge to construct the graph N' from N^+ by the following construction. Recall that v_1 and v_2 are the two vertices that are in both of $V(N^+)$ and $V(N^-)$. First attach a vertex u_1 by an edge to v_1 and attach another vertex u_2 by an edge to v_2 . Then delete the edge $e_2 = (v_3, v_4)$ and for each of v_3 and v_4 , attach the vertices u_3 and u_4 via an edge to v_3 and v_4 respectively. Note that as N^+ is 2-connected, the graph N' is connected. An example of such a graph N' is shown in Figure 3.2.

Figure 3.2: The graph N' .

Now for a vertex v with degree 2 we will create the graph N_2 by taking N^- and attaching a path of length 5 between the vertices v_1 and v_2 . We will label the added edges $f_1, \dots, f_5$ and the vertices $w_1, \dots, w_4$ so that $f_1 = \{v_1, w_1\}$, $f_2 =$

$\{w_1, w_2\}, \dots, f_5 = \{w_4, v_2\}$. For a vertex with degree 3 we will create the graph N_3 by taking N^- and attaching a similar path of length 7 from v_1 to v_2 . We will use the same convention for labelling the added vertices and edges. We will denote the copy of N_2 or N_3 created for vertex v as N^v . We will define the edges f_{2j+1} as *deletable edges*. These will be the edges we are allowed to delete in our instance of the π edge deletion problem with selected edges used in the reduction at the end of this chapter. For an example of the graphs N_2 and N_3 , see Figure 3.3 where the deletable edges are represented by dashed lines.

Figure 3.3: The graphs N_2 and N_3 .

Suppose we are given a planar graph G with no vertex degree exceeding 3 and asked for a vertex cover. Without loss of generality, we can assume that G has no vertex with degree 1 as such a vertex will have no effect on the complexity of finding a vertex cover of G . Create the graph G^1 by subdividing each edge of $\mathcal{E}(G)$ twice. This has the effect of adding two edges in series with each edge of G . Note that G^1 remains planar. Note also that due to Lemma 3.2.1, finding a vertex cover of G^1 will give a vertex cover for the original graph G . When we create G^1 we will create a path of three edges from each edge e in $\mathcal{E}(G)$. We will denote the middle edge of each path as *the internal edge with respect to e* . Now, given G^1 we will create the graph G^2 by replacing vertices and edges with the graphs N^v and N' respectively in the following manner.

1. For each vertex $v \in V(G^1)$ with degree 2, create a copy of the graph N_2 . For each vertex v of degree 3 create the graph N_3 .
2. If two vertices v and v' are adjacent in G^1 , join the graphs N^v and $N^{v'}$ with a copy of N' by doing the following. First choose an edge $f_{2i} = (w_{2i-1}, w_{2i}) \in$

$\mathcal{E}(N^v)$ and delete it. Next identify u_1 from N' with w_{2i-1} and u_2 with w_{2i} . Then delete an edge $f_{2j} = (w_{2j-1}, w_{2j})$ from $\mathcal{E}(N^{v'})$ and identify u_3 and u_4 from N' with w_{2j-1} and w_{2j} respectively. Note that an embedding of a planar graph induces an ordering on edges incident to a common vertex (by rotating clockwise around the vertex). When we add copies of N' for edges by attaching them to N_3 , maintain this ordering.

For an example of this construction see Figure 3.4 where the dashed edges are the deletable edges. Note that in Figure 3.4 each edge in G has not been extended in series to make G^1 . The figure is just to illustrate how we construct the graph G^2 from piecing together the graphs N' , N_2 and N_3 .

Figure 3.4: Replacing the edges and vertices of G with N' , N_2 and N_3 .

Lemma 3.2.7. *Suppose G^2 is a graph made from the above construction from a planar graph G with no vertex degree exceeding 3 and let $e = (v, v')$ be some edge of G . Let G' be the subgraph of G^2 induced by the vertices of N^v and $N^{v'}$ and any copies of N' attached to them. Then from G' we can obtain a graph isomorphic to N by deleting edges and isolated vertices and contracting series extensions.*

Proof. Recall that without loss of generality, we can assume that G has no vertex of degree one. Consider a copy of N' attached at either end to a copy of N_2 or N_3 . This corresponds to an edge e incident with the two vertices v and v' . Recall that

the graphs N_2 or N_3 that correspond to the vertices v and v' are denoted as N^v and $N^{v'}$ respectively. Without loss of generality, we can assume that N' is joined to N^v and $N^{v'}$ by identifying u_1, u_2, u_3 and u_4 with w_1, w_2, w'_1 and w'_2 respectively where $w_1, w_2 \in V(N^v)$ and $w'_1, w'_2 \in V(N^{v'})$. By the construction, there will be other copies of N' attached to N^v and $N^{v'}$ (because G has no vertex of degree one). We can assume without loss of generality that there is only one additional copy of N' attached to each of N^v and $N^{v'}$ and that each of these additional copies of N' have been attached to N^v or $N^{v'}$ by identifying the vertices $u_1, u_2 \in V(N')$ with the vertices $w_i \in V(N^v)$ or $w'_j \in V(N^{v'})$. In each of these additional copies of N' , there will be at least one path from u_1 to u_2 . Fix such a path and delete all other vertices and edges from these copies of N' . These remaining paths are series extensions of f_1 or f'_1 and therefore we can contract them to a single edge. Do this for all copies of N' that are joined to one but not both of N^v and $N^{v'}$. This will give the graph shown in Figure 3.5.

Figure 3.5: The graph G^2 after deleting some edges and vertices and contracting some series extensions.

In each copy of N^- in $N^{v'}$, there will be a path from v_1 to v_2 . Delete all other edges from this copy of N^- . This will leave just a path from v_3 to v_4 where $v_3, v_4 \in V(N')$. Because this path will just be a number of series extensions, we can contract it down to a single edge (v_3, v_4) . Now consider N^v . From Lemma 3.2.5 we know that there is no cycle in N^- that contains e_1 . Also recall that v_1 and v_2 are in both $V(N^+)$ and $V(N^-)$. As N is 2-connected, there must be a cycle that contains e_1 and some edge in N^+ . Furthermore, as v_1 and v_2 are the only edges that are in both N^+ and N^- , all cycles that contain e_1 must pass through the vertices v_1 and v_2 . Therefore each edge f_i is a series extension of e_1 that is in the copy of N^- . Contract all these edges f_i . Now finally the edges (u_1, v_1) and (u_2, v_2) in N' will also be a series extension of e_1 and can therefore be contracted. The resulting graph is isomorphic to N . $\square$

Each vertex v of G^1 can now be associated with 3 or 4 edges f_i of G^2 , depending

on the degree of v . These are the dashed edges in figures 3.3 and 3.4. They are the deletable edges in the graph N^v that correspond to each vertex v . Now for each face of G choose two distinct edges e_m and e_n that border this face. Take a copy of N and subdivide $e_1 = (v, v')$ twice. Recall that e_1 is fixed and contained in $c(N)$. Let $e_i = \{x_1, x_2\}$ be the internal edge with respect to e_1 after this subdivision. Now delete e_i and identify x_1 with any vertex $\notin \{u_1, u_2, u_3, u_4\}$ of the copy of N' that corresponds to the internal edge with respect to e_m and identify x_2 with any vertex $\notin \{u_1, u_2, u_3, u_4\}$ of the copy of N' that corresponds to the internal edge with respect to e_n . We will call a copy of N that has been added in this fashion a *spanning N* . Call the graph created G^3 . Figure 3.6 shows a sketch of how G^3 is created from G^2 without including the details of G^2 . The large filled in circles represent copies of N_2 or N_3 while the two lines between them represent copies of N' . In G^3 we will imaginatively call any cycle that contains edges from a spanning N and edges not from the spanning N a *bad cycle*.

Figure 3.6: Creating the graph G^3 .

Lemma 3.2.8. *Let G^2 be a graph made from the above construction from a simple planar graph G with maximum vertex degree 3 and at least two edges. Then we can always add spanning N 's to create G^3 so that every bad cycle of G^3 contains at least one deletable edge. Moreover, deciding where to add the spanning N 's can be done in polynomial time.*

Proof. Take the graph G^2 constructed from a simple planar graph G with maximum vertex degree 3. Assume that some arbitrary configuration of spanning N 's has been attached to make G^3 . Note that if a bad cycle contains no deletable

edge, then there must be a path between the vertices x_1 and x_2 from the spanning N that passes through no deletable edge. Consider a copy of N' in G^3 that replaced an edge from G^1 . Any path from this to another copy of N' must contain a deletable edge. Thus if a bad cycle contains no deletable edge, then both copies of N' that the spanning N is attached to must have two spanning N' 's attached to them. So the bad cycle can only contain edges from the spanning N 's and the copies of N' that they are attached to. Consider the graph G_s where the vertices of G_s correspond to a copies of N' in G^3 (edges of G^1) and two vertices are joined if there is a spanning N that is attached to both of their corresponding copies of N' . For an example, the graph G_s created for the graph G^3 in Figure 3.6 is shown in Figure 3.7.

Figure 3.7: Example of the graph G_s made from G^3 .

A bad cycle that contains no deletable edge corresponds to a cycle in G_s . Each edge in G can be in the border of no more than two faces. Thus each copy of N' that corresponds to an edge of G can have at most two spanning N 's attached to it. It follows that G_s can have maximum vertex degree 2 and is therefore a collection of disjoint cycles and trees. Take a cycle in G_s and choose an arbitrary edge $e = (u, v)$ in this cycle. This edge will correspond to a spanning N , denoted N_e , in G^3 and u and v will correspond to copies of N' in G^2 that in turn correspond to internal edges in G^1 (recall that an internal edge is the middle edge in the path of length 3 that the edges of G were replaced with to make G^1). As G is simple, each face of G has at least three edges in its border. Therefore there will be another copy of N' that N_e could be attached to. This copy of N' corresponds to an internal edge with respect to f where f is some edge of G that borders the

same face as the edges that correspond to u and v . The vertex corresponding to this copy of N' in G_s , denoted w , will be in no cycle and in a different connected component to e . Thus replacing the edge e in G_s with the edge (u, w) will reduce the number of cycles by one. This can be done for all cycles in G_s resulting in a forest. Once we have a forest, we know that all bad cycles contain a deletable edge. This gives a method of attaching spanning N 's to G^2 such that every bad cycle contains at least one deletable edge. Moreover this can be achieved in polynomial time. $\square$

Note that because spanning N 's can only be attached to internal edges, every bad cycle will actually have at least three deletable edges when we follow the construction given in Lemma 3.2.8. From now on we will assume that all graphs G^3 will be constructed so that every bad cycle contains deletable edges. This is necessary because later on we will be attempting to remove bad cycles by only deleting deletable edges.

Lemma 3.2.9. *Let G^3 be a graph made from the above construction. Let G_d be obtained from G^3 by deleting any number (possibly zero) of deletable edges such that G_d has a bad cycle. Then by deleting edges and vertices and contracting series extensions, we can obtain a graph isomorphic to N .*

Proof. Let G_d be such a graph with such a bad cycle. Delete all edges and vertices that are not in the bad cycle or the spanning N . The remaining edges that are not in the spanning N will correspond to a series extension of e_1 and can therefore be contracted down to a single edge. The resulting graph will be isomorphic to N . $\square$

3.3 Masks and the Mask Graph

When we replace edges and vertices of a planar graph with copies of N' and N^v , we may lose planarity. In this section, we define masks and the mask graph. These are used as a means of keeping track of the faces of the original graph. This section also contains results on properties of masks and the mask graph.

Let G be any simple planar graph with maximum vertex degree 3 and G^1 and G^3 be graphs constructed from G by the above construction. We will first define a *mask graph* $M(G^3, F)$ where F is any set of deletable edges of G^3 . A mask graph can be thought of as a representation of what is left of the original structure of G

after the edges in F have been deleted. We begin by modifying the graph G^1 . For any edge e of G^1 , if the copy of N' in G^3 that corresponds to e has a spanning N attached to it, subdivide e . Join the two vertices created by the two subdivisions from each spanning N with an edge. Call this graph G^* . Consider a vertex v of G^* that corresponds to N^v from G^3 . There will be a corresponding graph, N^v , for every vertex v of G^* that wasn't created by subdividing an edge due to a spanning N . To begin with assume that $N^v \simeq N_2$. If F contains two or more of the edges f_1, f_3 and f_5 from this copy of N_2 , then in G^* , delete v .

Now suppose that $N^v \simeq N_3$. In G^* , we will assume that the edges incident with v are $g_1 = (v, x_1)$, $g_2 = (v, x_2)$ and $g_3 = (v, x_3)$. Furthermore, we will assume that in G^3 , N^{x_1} is attached to N^v at vertices w_1 and w_2 , N^{x_2} is attached to N^v at w_3 and w_4 and N^{x_3} is attached to N^v at w_5 and w_6 . Now depending on which of the edges f_1, f_3, f_5 and f_7 are in F we will delete different edges g_1, g_2 and g_3 from G^* .

1. If both f_1 and f_3 are in F or if both f_3 and f_7 are in F then delete g_1 from G^* .
2. If both f_3 and f_5 are in F then delete g_2 from G^* .
3. Finally, if both f_1 and f_5 are in F or both f_5 and f_7 are in F then delete g_3 .

Note that if both f_1 and f_7 are in F , then we do not delete any edges from G^* (unless one or more of the above conditions are also met). If two or more of the edges g_1, g_2 and g_3 have been deleted, delete the vertex corresponding to v . The resulting graph is the mask graph $M(G^3, F)$.

An example of the graph $M(G^3, \emptyset)$ for the graphs in Figure 3.6 is shown in Figure 3.8. Note that up to isomorphism there is a unique mask graph for each G^3 and set of deletable edges F . Also note that the mask graph is always planar as G^1 is planar and series extensions and subdividing faces preserves planarity.

Let G_d be any graph made from taking a graph G^3 constructed by the above method and deleting any number of deletable edges from G^3 . Define a *mask* of G_d to be a face of the mask graph $M(G^3, \mathcal{E}(G^3) - \mathcal{E}(G_d))$. If G has f faces and $F = \emptyset$, then G_d has $2f$ masks. Having more than one mask can be thought of as a certificate that G_d has a bad cycle and therefore $\oplus_k G_d$ violates π for some k . To see this, let x_1 and x_2 be the two vertices of a spanning N that have been identified with vertices of G^2 . If there is more than one mask, then for some spanning N there will be a path from x_1 to x_2 that does not include any other vertex of the

Figure 3.8: Creating the mask graph $M(G^3, \emptyset)$.

spanning N . This is a bad cycle and therefore by Lemma 3.2.9 $\oplus_k G_d$ violates π if it has more than one mask.

Lemma 3.3.1. *Let G^3 be a graph constructed from a simple planar graph with no vertex degree greater than 3. Let G_d be a graph obtained from G^3 by deleting any number of deletable edges and let e be any deletable edge of G_d . Suppose also that G_d has f masks. Then $G_d \setminus e$ has at least $f - 1$ masks.*

Proof. To begin with assume that e is from a copy of N_2 . Then this corresponds to a vertex v of degree at most 2 in the mask graph. Therefore deleting any number of edges incident with v can only reduce the number of faces in the mask graph by one as all edges incident with v border the same two faces.

Now assume that e is deleted from a copy of N_3 . The cases where 0, 1 or 3 deletable edges have already been deleted from this copy of N_3 are all straightforward. Suppose that no deletable edge has been deleted from this copy of N_3 . Then deleting e will not reduce the number of masks. So suppose that a single edge has already been deleted from this copy of N_3 . Then deleting e can only result in deleting a single edge from the mask graph and thus only reduce the number of masks by one. If three edges have already been deleted from this copy of N_3 then its corresponding vertex in the mask graph will have degree at most 2 and therefore all incident edges remaining will border the same two faces. Therefore deleting the final deletable edge from this copy of N_3 will not reduce the number of masks by more than one.

Now suppose that two deletable edges have already been deleted from this copy of N_3 . If these edges are not f_1 and f_7 then the corresponding vertex in the mask

graph will have degree 2 and thus deleting all edges incident with it will only reduce the number of masks by one. Thus we assume that the edges already deleted are f_1 and f_7 . Now suppose that $e = f_3$. Then the pairs of deletable edges that result in an edge being deleted from the mask graph are the pair f_1 and f_3 and the pair f_3 and f_7 (as the pair f_1 and f_7 does not result in an edge being deleted from the mask graph). However, the pair f_1 and f_3 and the pair f_3 and f_7 both result in the same edge being deleted from the mask graph (the edge g_1). As only one edge is deleted from the mask graph, the number of masks can only be reduced by one. So suppose that $e = f_5$. Then the pairs of deletable edges whose deletion results in an edge being deleted from the mask graph are the pair f_1 and f_5 and the pair f_5 and f_7 . However, again we see that both these pairs result in a single edge (this time g_3) being deleted from the mask graph. Therefore the number of masks only decreases by one.

Therefore $G_d \setminus e$ must have at least $f - 1$ masks. $\square$

Lemma 3.3.2. *Let G^3 be constructed by the above construction from a simple planar graph with no vertex degree greater than 3. Let G_u be the graph made from G^3 by deleting a single deletable edge from every copy of N^v that corresponds to a vertex of G^1 plus any number of other deletable edges. Suppose we are only allowed to delete deletable edges from G^3 . Then the minimum number of edges whose deletion removes all bad cycles from G_u is $f - 1$ where G_u has f masks.*

Proof. We will prove this by induction. Let F be the set of edges deleted to make G_u . Suppose G_u has exactly two masks. This is the minimum created by the above construction that still has a bad cycle. As a deletable edge has been deleted from every copy of N_2 and N_3 , we can remove all bad cycles by deleting a single deletable edge e . This is because the mask graph $M(G^3, F \cup e)$ is a forest and therefore $G_u \setminus e$ has no bad cycle. This proves the base case for the induction.

Now assume that this holds for all graphs G_u with no more than k masks that have been obtained by the construction above. Take some G_u with no more than $k + 1$ masks. Delete any deletable edge e . Deleting one such edge will need to be done to remove a bad cycle. If deleting this edge does not reduce the number of masks, then there must still be bad cycles in $G_u \setminus e$. Therefore we need to delete another deletable edge. Keep deleting deletable edges until the number of masks is reduced by one. By Lemma 3.3.1 this is the maximum reduction in masks possible by deleting a single deletable edge. Now by the induction hypothesis it will require at least another $k - 1$ deletions to remove all bad cycles. This gives a total of at

least $k - 1 + 1 = k = (k + 1) - 1$ deletions. Therefore it takes at least $f - 1$ deletions to remove all bad cycles from a graph G_u that has f masks. $\square$

Lemma 3.3.3. *Let G_{vc} be the graph made from taking a simple planar graph with maximum vertex degree 3, constructing the graph G^3 from G by the above method and then deleting a deletable edge from a copy of N^v that corresponds to a vertex of G^1 for each vertex v in a vertex cover of G^1 plus any number (possibly zero) of additional deletable edges. Then by deleting no more than $f - 1$ deletable edges from G_{vc} where G_{vc} has f masks, we can obtain a graph that contains no bad cycles.*

Proof. This will be done by induction. Assume that G_{vc} has two masks. This is the minimal number of masks a graph can have while having a bad cycle. Because each edge in G is replaced by 3 edges in G^1 , there will be a vertex $v \in V(G^1)$ that is in the given vertex cover and borders the two faces of the mask graph $M(G^3, VC)$ where VC is the set of edges in the given vertex cover. As a deletable edge corresponding to v has already been deleted, there will exist at most three deletable edges (and because v borders two faces, at least two deletable edges) that correspond to v remaining in G_{vc} . We can choose one of these deletable edges so that deleting it will also remove an edge in the mask graph which will reduce the number of faces of the mask graph. This will leave a graph with no bad cycles.

Now assume that this holds for all possible graphs G_{vc} that have at most k masks. Take some graph G_{vc} that has at most $k + 1$ masks. Choose some deletable edge that corresponds to a vertex v of G^1 that is in the vertex cover of G^1 and is in the border of two masks in $M(G^3, \mathcal{E}(G^3) - \mathcal{E}(G_{vc}))$. Because each edge of G is replaced by three edges, there will always be such an edge that can be chosen that corresponds to a vertex on such a border. Note that if f_1 or f_7 have been deleted from the copy of N^v , then when we choose another edge to delete we will not choose f_7 or f_1 respectively. Delete the chosen edge. Because an edge corresponding to this vertex has already been deleted (because we have deleted edges corresponding to a vertex cover), this will reduce the number of masks of G_{vc} by 1. Now by the induction hypothesis we require only another $k - 1$ deletions to remove all masks. $\square$

Note that the procedure in Lemma 3.3.3 can be carried out in polynomial time.

3.4 The Reduction

We now have all the tools required for our reduction. We will begin by showing that the π edge deletion problem with selected edges is NP-hard. We will then use a reduction from this to the π edge deletion problem.

Theorem 3.4.1. *Let π be a non-trivial hereditary property that is malleable. Then the π edge deletion problem with selected edges is NP-hard.*

Proof. This will be shown via a reduction from the vertex cover problem on planar graphs with no vertex degree exceeding 3.

Before we launch into the proof, we will give a rough overview of how it is organized. We are going to take an instance of the vertex cover problem on G and transform it into an instance of the π edge deletion problem with selected edges. We will do this by constructing the graph G^3 from G . We will then make a graph G_k from a direct sum of a polynomial number of copies of G^3 . It will be shown that we can delete deletable edges corresponding to vertices in a vertex cover of G plus a polynomial number of other deletable edges that can easily be found from G_k to obtain a graph that satisfies π . We will show this by constructing a graph $N'' <_\alpha N$ which implies that $\oplus_j N''$ satisfies π for any j . We will choose N'' so that the graph we get from deleting our chosen deletable edges from G_k can be obtained from $\oplus_j N''$, for some j , by performing a number of allowable modifications, such as deleting edges and contracting series extensions. These operations are allowable because when performed on a graph that satisfies π , the resulting graph will also satisfy π . We will use N'' to show that the graph we get from deleting edges from G_k satisfies π . This will give an upper bound on the number of deletions required to make G_k satisfy π . We will then show that if we cannot create a vertex cover of a certain size then we can also not delete a related number of deletable edges to obtain a graph that satisfies π .

Let G be an instance of the vertex cover on planar graphs with no vertex degree exceeding 3. Without loss of generality we can assume that G is simple and has no vertices of degree one. Construct G^3 using the construction above. Now let $G_k = \oplus_{3nk} G^3$ where n is the number of vertices in G^1 . Denote F to be a minimum solution to the π edge deletion problem with selected edges on G_k where we are only allowed to delete deletable edges (the edges that correspond to vertices of G^1). Let V_G be a minimum vertex cover of G and V_{G^1} be a minimum vertex cover

for G^1 . We will show that $|V_G| \leq h$ if and only if $|V_{G^1}| \leq h + |\mathcal{E}(G)| = l$ if and only if $|F| \leq 3nk(l + 2f - 1)$ where f is the number of faces of G .

First suppose that $|V_G| \leq h$. Then $|V_{G^1}| \leq l$ by Lemma 3.2.1. In each copy of G^3 delete a deletable edge corresponding to each of the (no more than l) vertices $v \in V_{G^1}$. Then by Lemma 3.3.3, we only need to delete another $f' - 1$ deletable edges to remove all bad cycles from G^3 where G^3 has f' masks. By the construction of G^3 , it will have $2f$ masks where G has f faces. Therefore this is a total of no more than $l + 2f - 1$ edges deleted from each copy of G^3 in G_k and no more than $3nk(l + 2f - 1)$ edges deleted from G_k . Let F' be this set of deleted edges.

If we can show that deleting the edges in F' gives a graph that satisfies π , then we have shown that $|F| \leq 3nk(l + 2f - 1)$ if $|V_G| \leq h$. Let $G'_k = G_k - F'$. Consider the graph N'' shown in Figure 3.9 where N_D is shown in Figure 3.10. The reason for N_D is that given a copy of N^v that corresponds to a vertex, we don't know if the copies of N' corresponding to edges of G^1 are attached via u_1 and u_2 or via u_3 and u_4 . The graph N_D can be used to simulate either orientation.

Figure 3.9: The graph N'' .

We will show that each 2-connected component of G'_k can be obtained from N'' via a series of edge deletions and contractions of series extensions and deletion of isolated vertices. To see this consider a 2-connected component of G'_k . Suppose the 2-connected component contains edges that were from N_v and $N_{v'}$ for distinct vertices v and v' of G^1 . Then by the construction, there would be a copy of N' joined to them and for this to be a 2-connected component either

1. neither of N_v or $N_{v'}$ could have had any deletions; or

Figure 3.10: The graph N_D .

2. the vertices v and v' must be in a cycle in G_1 and all other vertices in this cycle must have edges from their corresponding copy of N^v in this 2-connected component.

In the first case the vertices v and v' would not be in V_G . But then the edges in F' would not correspond to a vertex cover of G^1 which is a contradiction. Now, in the second case there would exist a bad cycle in G'_k which is a contradiction as we have removed all bad cycles. Thus each 2-connected component of G'_k can include edges from at most one copy of N^v corresponding to some vertex of G^1 . Note that in N'' , the copy of N^v has six subgraphs of N' attached to it while in G'_k each copy of N^v can have at most three. The reason for this is we don't know how each copy of N' is attached to each copy of N^v . That is, has it been attached by the vertices u_1 and u_2 or the vertices u_3 and u_4 . The graph N_D accounts for both possible orientations. Each 2-connected component of G'_k will be a subgraph of the graph N_2 or N_3 with copies of N' attached. Thus each 2-connected component of G'_k can be obtained from N'' by contracting series extensions and deleting edges and isolated vertices. Note that as π is completely closed under series compositions, we only need that the 2-connected components of G'_k can be obtained from N'' . Recall that Lemma 3.1.3 states that π is completely closed under deletion of isolated vertices. As π is completely closed under series compositions, this implies that G'_k will satisfy π if the graph $\oplus_j N''$ satisfies π for any j . This will hold if $N'' <_\alpha N$.

Consider the graph N'' . It will be 2-connected because adding an edge $e = (u_1, u_4)$ to N_D will create a 2-connected graph. First suppose that $N \setminus e_1$ only has one maximum sized 2-connected component. If we delete one of the deletable edges from N'' , then the resulting graph will have no 2-connected component of size $|N^+|$ as each copy of N^+ in N'' has had a deletion and there are no other 2-connected

components of size $|N^+|$. In this case, we see that $N'' <_\alpha N$ and therefore $\oplus_j N''$ will satisfy π for any j .

Now suppose that $N \setminus e_1$ has m maximum sized 2-connected components. Then, deleting one of the deletable edges from N'' will give a graph with only $m - 1$ maximum sized 2-connected components. As these will be no larger than the maximum sized 2-connected components of N , we again see that $N'' <_\alpha N$ and therefore $\oplus_j N''$ will satisfy π for any j .

Now consider each 2-connected component of G'_k . This can be obtained from a copy of N'' by deleting edges and vertices and contracting series extensions. Thus we can make the graph G'_k by taking copies of N'' and doing these deletions and contractions until we have all the 2-connected components of G'_k . We can then add isolated vertices and perform series compositions until we have created the graph G'_k . Thus G'_k can be obtained from a graph $\oplus_j N''$, for some j , via edge deletions, contractions of series extensions, addition or deletion of isolated vertices and series compositions. As π is hereditary, malleable and $\oplus_j N''$ satisfies π for all j , G'_k must also satisfy π . Therefore $|F| \leq 3nk(l + 2f - 1)$.

Now suppose that $|V_G| \geq h + 1$ and therefore $|V_{G^1}| \geq l + 1$. We will show that this implies that $|F| > 3nk(l + 2f - 1)$. Recall that F is a minimum set of edges whose deletion from G_k gives a graph that satisfies π . Every deletable edge from G_k corresponds to a vertex of G^1 . From at least $3nk - (k - 1)$ copies of G^3 , there cannot be any graph isomorphic to N that can be obtained by deleting edges and isolated vertices and contracting series extensions. Consider a copy of G^3 such that when the edges in F have been removed we cannot obtain a graph isomorphic to N by deleting edges and isolated vertices and contractions of series extensions. We will denote this copy of G^3 by G^* . Also let F^* be the edges deleted from a copy of G^3 to create G^* . Suppose F^* does not contain edges corresponding to a vertex cover of G^1 . Then there would be an edge incident to two vertices remaining in G^1 after deleting the vertices corresponding to edges in F^* . By Lemma 3.2.7, this would imply that we can get a graph isomorphic to N by deleting edges and vertices and contracting series extensions. Hence F^* must contain edges that correspond to a vertex cover of G^1 . Therefore $|F^*| \geq l + 1$. Now suppose that in G^* we have a bad cycle. Then by Lemma 3.2.9 we know we can obtain a graph isomorphic to N by edge and vertex deletions and contraction of series extensions. Hence another contradiction. Therefore G^* must have no bad cycle. From Lemma 3.3.2 we know that no matter what set of edges corresponding to a vertex cover of G^1 is deleted

from G^3 , we still need at least another $f' - 1$ deletions from G^3 to remove all such bad cycles where G^3 has f' masks. By the construction of G^3 , $f' - 1 = 2f - 1$ and therefore $|F^*| \geq l + 1 + 2f - 1 = l + 2f$. This implies that

$$\begin{aligned}
 |F| \geq (3nk - k + 1)(l + 2f) &= 3nkl + 6nkf \\
 &\quad - kl - 2kf + l + 2f \\
 &= 3nkl + 6nkf + k(-l - 2f) \\
 &\quad + l + 2f \\
 &= 3nk(l + 2f - 1) + k(3n - l - 2f) \\
 &\quad + l + 2f.
 \end{aligned}$$

Since $n > l$ and $n > f$, we have $3n - l - 2f > 0$ and therefore $|F| > 3nk(l + 2f - 1)$. This shows that $|F| \leq 3nk(l + 2f - 1)$ if and only if $|V_G| \leq k$. As the construction that created the graph G_k can be done in polynomial time, this shows that given an algorithm to solve the π edge deletion problem with selected edges, we can use it to solve the vertex cover problem on planar graphs with no vertex degree exceeding 3. Therefore the π edge deletion problem with selected edges is NP-hard. $\square$

We have shown that the π edge deletion problem with selected edges is NP-hard. We now present a simple reduction from this to the π edge deletion problem, showing that the π edge deletion problem is NP-hard.

Theorem 3.4.2. *Let π be a non-trivial hereditary graph property that is malleable. Then the π edge deletion problem is NP-hard.*

Proof. Take an instance G , k and $\mathcal{E}'$ of the π edge deletion problem with selected edges. Add k parallel edges to each edge of $\mathcal{E}(G) - \mathcal{E}'$. Call this graph G' . Now let F be a minimal solution to the π edge deletion problem with selected edges on G and let F' be a minimal solution to the π edge deletion problem on G' . We will show that $|F| \leq k$ if and only if $|F'| \leq k$.

Suppose $|F| \leq k$. Consider the edge induced subgraph $G' - F$. We can obtain this graph from $G - F$ by adding parallel edges to the edges in $\mathcal{E}(G) - F - \mathcal{E}'$. As π is completely closed under parallel extensions, this graph $G' - F$ must also satisfy π . Thus $|F'| \leq k$.

Now suppose $|F| > k$ and consider F' . If $|F'| \leq k$ then F' must consist of only edges that are selected edges in the π edge deletion problem with selected edges.

This is because if F' contains an edge e from G' that is not a selected edge, then as π is closed under parallel extensions, F' must contain all k edges parallel to e . But then $|F'| > k$. This is a contradiction. Therefore $|F| \leq k$ if and only if $|F'| \leq k$. It follows that the π edge deletion problem is NP-hard. $\square$

Chapter 4

Basis and Circuit Counting

4.1 Introduction

In this chapter we are interested in the difficulty of counting problems. Specifically, counting the number of bases or circuits of a given matroid. These problems are not decision problems and therefore are not in the class NP. Instead they are in the complexity class $\#P$. This is the counting version of the class NP of decision problems. For example the decision problem could be: is there a vertex cover of size k ? While the corresponding enumeration problem would be: how many vertex covers of size k are there? The complexity class $\#P$ was introduced by Valiant in 1979 when showing that the problem of calculating the permanent of a matrix is $\#P$ -complete [70]. We begin our definition of $\#P$ with a *counting Turing machine*. This is just a standard non-deterministic Turing machine with additional output that prints the number of accepting paths. The time complexity is that of the longest accepting path. The class $\#P$ is the class of problems that can be solved in polynomial time on a counting Turing machine. Thus we see that a Turing machine with an $\#P$ oracle is at least as powerful as a Turing machine with a NP oracle as any problem that can be verified in polynomial time by a non-deterministic Turing machine can be solved in polynomial time by a counting Turing machine. Despite this, we note that the classes NP and $\#P$ cannot be directly compared as NP is a class of decision problems while $\#P$ is a class of functions.

The notion of NP-completeness carries over to the class $\#P$. Much as NP-complete problems are the hardest problems in NP, $\#P$ -complete problems are the hardest problems in $\#P$. The enumeration version of a number of NP-complete

problems are known to be $\#P$ -complete. However, it is certainly not known that the enumeration versions of all NP-complete problems are $\#P$ -complete. As an example of this, consider the problem of deciding if a graph has a Hamiltonian induced subgraph of size $\geq k$ or more. It follows from the NP-completeness of finding a Hamiltonian circuit that it is NP-complete to find a vertex induced subgraph with a Hamiltonian circuit. This is because given a graph, we can set $k = |V(G)|$. However, the problem of enumerating the number of Hamiltonian vertex induced subgraphs of size $\geq k$ is believed to not be in $\#P$. The reason for this is to know you have a Hamiltonian subgraph you would need to find a vertex induced subgraph and a Hamiltonian circuit of the vertex induced subgraph. However, the number of such pairs is not the number of vertex induced Hamiltonian subgraphs. For more details of this and the class $\#P$ see [79].

The method of showing that a problem is $\#P$ -complete is similar to that of showing that a problem is NP-complete. We take a known $\#P$ -complete problem and show that by performing a polynomial-time reductions to our problem, we can extract the solution to the known $\#P$ -complete problem from the solution of our problem. The major difference here is that we are allowed to perform multiple oracle calls in this reduction.

It is worth pointing out that some decision problems that are in P have corresponding enumeration problems that are $\#P$ -complete. Examples of these include counting forests of a graph or perfect matchings in bipartite graphs ([41] and [71] respectively). Counting forests of a graph will be of particular use to us in this chapter as the proofs of our main basis counting results will be reductions from the problem of counting forests in a graph.

The main focus of this chapter will be considering the difficulty of counting bases of matroids. We note that given a rank or independence oracle or a matrix representation, finding a basis of a matroid is easy. However, in many cases counting the bases of a matroid is $\#P$ -complete. For example, it is $\#P$ -complete to count the number of bases of transversal matroids or bicircular matroids ([21] and [31] respectively).

4.2 Basis Counting in Representable Matroids

Representable matroids are an important class of matroids and it is only natural to consider the difficulty of counting bases for the class of representable matroids.

The focus of this section will be to study the complexity of counting bases for matroids representable over the fields $\text{GF}(q)$ for fixed q . The obvious goal would be a theorem that answers Question 4.2.1.

Question 4.2.1. *Is it $\#P$ -complete to count the number of bases of a representable matroid over any fixed field?*

Vertigan proved this to be $\#P$ -complete in 1991. This result has been referenced in several papers [21, 76]. However, no publication was ever produced. This feels like a substantial hole in the literature and should be remedied. One of the surprising things about this result is the fact that it is easy to count bases of graphic matroids while Vertigan's result implies that it is hard to count the number of bases of binary matroids.

While we do not resolve Question 4.2.1, we provide proofs for the fact that it is $\#P$ -complete to count bases of representable matroids over

- (i) fixed infinite fields and
- (ii) finite fields of a fixed characteristic.

Note that any transversal matroid is representable over any sufficiently large field. Therefore the fact that it is $\#P$ -complete to count bases of transversal matroids implies that it is $\#P$ -complete to count bases of matroids over any infinite field, provided an appropriate construction is given [21]. Furthermore given a bicircular matroid M , we can create a representation of M over a field of any characteristic using transcendentals. Thus it follows that it is $\#P$ -complete to count bases of matroids representable over a fixed characteristic given an appropriate construction [31]. This is the goal of this section. We provide polynomial-time constructions that can be used in proving that it is $\#P$ -complete to count bases of matroids representable over any infinite field or finite fields of a fixed characteristic. While these results are not strictly speaking new, we believe the constructions provided are. We are unaware of any appropriate polynomial-time constructions being previously written down.

We will be using the operation of truncation for several of the reductions that follow. A problem with using truncation on representable matroids is that truncation of a representable matroid does not always produce a matroid representable over the same field. Even if truncating produces a matroid representable over the same field, it may be hard to construct a representation of the resulting matroid. Thus we need to find a way of producing an appropriate representation of a ma-

troid created by truncation. There are ways of getting around this though. The operation of truncation is equivalent to adding an element freely and then contracting it. It is often easier to create a representation of the matroid obtained by adding elements freely than it is to create the representation of a truncated matroid. This is partly due to the fact that to add elements freely, we only need to create a few columns of the matrix while to create the truncated representation we need to create a matrix representation almost from scratch. This is why, if we want to truncate a representable matroid, we will often add elements freely and then contract them. We can do this because contraction preserves representability. Therefore if we can find a representation of the matroid obtained by adding elements freely, we can obtain a representation of the truncated matroid.

Our approach to proving that counting bases is $\#P$ -complete for matroids representable over fixed infinite fields and fields of fixed characteristic will be similar. We will begin by adding elements freely to a given representable matroid. We will then construct a representation over an appropriate field for the matroid obtained and contract the added elements. This will allow us to create representations for truncations of the given matroid. We will then use this construction of the truncated matroid in a reduction from the known $\#P$ -complete problem of counting forests of a graph.

$\#FORESTS$

INSTANCE: A graph G .

QUESTION: How many forests does G have?

Note that we could use reductions from counting bases of transversal or bicircular matroids. These reductions would be very similar to the ones provided. In all of our reductions from $\#Forests$, we will construct a totally unimodular representation of the cycle matroid of the graph G in polynomial time. We can do this in the following fashion ([58], Chapter 5). Take the graph G and arbitrarily direct each edge to form the directed graph $D(G)$. Then the totally unimodular representation of G is the incidence matrix of $D(G)$. For the rest of this chapter, we will assume that all representations of graphic matroids are constructed by this method. Thus they are all totally unimodular.

In the reductions that follow, we will need to be able to add elements freely to the matrices produced by the above method to produce matroids representable

over certain fields. To do so, we will make use of a special type of matrix. An $n \times n$ *Vandermonde matrix* V is a matrix of the following form.

$$V = \begin{bmatrix} 1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^{n-1} \\ 1 & \alpha_2 & \alpha_2^2 & \cdots & \alpha_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_n & \alpha_n^2 & \cdots & \alpha_n^{n-1} \end{bmatrix}$$

For the Vandermonde matrix V , $\det(V) = \prod_{i < j} (\alpha_j - \alpha_i)$. Therefore if $\alpha_1, \dots, \alpha_n$ are all distinct, then $\det(V) \neq 0$. Otherwise, $\det(V) = 0$. In the following arguments we will be using a special matrix that is very similar to a Vandermonde matrix. We will say an $m \times n$ matrix X with entries in $\mathbb{Z}[x]$ is an r -*polynomial Vandermonde matrix* if $X_{i,j} = p_i(x^{k_j})$, where p_i is a monic polynomial such that $\deg(p_1) < \deg(p_2) < \dots < \deg(p_m) \leq r$ and $0 \leq k_1 < k_2 < \dots < k_n \leq r$. Note that if we let $p_i(x) = x^i$ and $k_j = j - 1$, then the r -polynomial Vandermonde matrix is also a Vandermonde matrix with $\alpha_i = x^i$.

Lemma 4.2.2. *Let X be a $n \times n$ r -polynomial Vandermonde matrix. Then $\det(X)$ is a non-zero monic polynomial with degree less than r^3 .*

Proof. The determinant of an $n \times n$ matrix X can be evaluated as $\sum_{\sigma \in S_n} \text{sgn}(\sigma) \prod_{i=1}^n X_{i,\sigma(i)}$ where $\text{sgn}(\sigma) = 1$ if σ is even and -1 otherwise. We will show that for the matrix X , there is a single product of maximum degree in this sum and thus there can be no cancellation of products and therefore the determinant cannot be 0. Moreover, this product of maximum degree will happen when σ is the identity permutation e .

Let P be some product $\prod_{i=1}^n X_{i,\sigma(i)}$ such that $\sigma \neq e$. Then P must contain some element $X_{i,j}$ where $i \neq j$. Take the greatest i such that $\sigma(i) \neq i$. Then P must also contain $X_{l,i}$ for some $l \neq i$ as σ is a permutation. As i has been chosen to be maximum, $i > j$ and $i > l$. Then as X is a r -polynomial Vandermonde matrix, $\deg(X_{i,i}) = \deg(p_i)k_i$, $\deg(X_{l,j}) = \deg(p_l)k_j$, $\deg(X_{i,j}) = \deg(p_i)k_j$ and $\deg(X_{l,i}) = \deg(p_l)k_i$. Let P' be a new product given by

$$P' = \frac{P X_{i,i} X_{l,j}}{X_{i,j} X_{l,i}}.$$

Note that

$$\deg(p_i)k_i + \deg(p_l)k_j - \deg(p_i)k_j - \deg(p_l)k_i = (\deg(p_i) - \deg(p_l))(k_i - k_j) > 0,$$

as $\deg(p_i) > \deg(p_l)$ and $k_i > k_j$. Thus $\deg(P') > \deg(P)$. Therefore by changing σ so that it fixes more elements we increase the degree. Hence we obtain the maximum degree only when σ is the identity permutation. As all $X_{i,j}$ are monic polynomials, any product of them must also be a monic polynomial. Thus the determinant of X is a non-zero monic polynomial. Now consider the product $\prod_{i=1}^n X_{i,i}$. This product will be a subproduct of $\prod_{i=1}^r X'_{i,i}$ which is a degree $\sum_{i=1}^r i^2$ monic polynomial. Thus

$$\deg(\det(X)) \leq \sum_{i=1}^r i^2 < r^3.$$

□

Lemma 4.2.3. *Let $A = [D|X]$ be a square matrix such that D is totally unimodular with non-zero determinant and X is an r -polynomial Vandermonde matrix. Then by row reductions and row swapping on A , we can get the following matrix*

$$A' = \left[\begin{array}{c|c} D' & X_t \\ \hline 0 & X_b \end{array} \right]$$

that has the following properties:

1. $|\det(A)| = |\det(A')|$;
2. D' is a square matrix in upper triangular form with non-zero entries on the diagonal; and
3. the matrix X_b is an r -polynomial Vandermonde matrix.

Proof. This will be proven by induction on the number of columns in D . Suppose D has no columns. Then A is already in the required form.

Now suppose this holds for matrices A where D has no more than k columns and take a matrix such that D has at most $k + 1$ columns. For $i < l$, consider the matrix X'' obtained from X by adding row i to row l α times. Then $X''_{l,j} = p_l(x^{k_j}) + \alpha p_i(x^{k_j})$. As $i < l$, this is a monic polynomial with the same degree as $X_{l,j}$. Thus X'' is also an r -polynomial Vandermonde matrix.

Now the columns of D are all linearly independent, so there must be a non-zero entry in the first column. Choose the first non-zero entry in the first column. Use row reductions so that below this non-zero element, the column is only zeros. Because in the row reduction, rows have only had rows added/subtracted to them from above, we see that the matrix X' obtained by the row operations is still an r -polynomial Vandermonde matrix. Now delete the first column and the row with the non-zero entry. Call the resulting matrix A'' . It now follows from the induction hypothesis that we can create a matrix in the required form from A'' . We can then add the deleted row back in as the first row and put a column of zeros under the non-zero element from the deleted row. This has the same effect as moving the deleted row to the top of the matrix. The resulting matrix will be in the desired form. As the only operations performed are adding or subtracting rows from one another and row swaps, the absolute value of the determinant has not changed. Thus $|\det(A)| = |\det(A')|$. $\square$

We know that the determinant of any r -polynomial Vandermonde matrix is a non-zero monic polynomial. We want a similar result for the determinant of a square submatrix of matrices of the form $[A|X]$ where A is a submatrix of a totally unimodular matrix and X is an r -polynomial Vandermonde matrix. As A is a submatrix of a totally unimodular matrix, we can no longer guarantee that $\det([A|X])$ is a monic polynomial. However, as we will see in Lemma 4.2.4, if $\det([A|X])$ is not a monic polynomial, then its leading coefficient is -1 . In light of this, we will define an *absolutely monic polynomial* to be a polynomial with leading coefficient 1 or -1 .

Lemma 4.2.4. *For $k < r$, let A be a rank k $r \times k$ totally unimodular matrix and X be a $r \times (r-k)$ r -polynomial Vandermonde matrix. Then $\det[A|X]$ is a non-zero absolutely monic polynomial of degree $\leq r^3$ and coefficients of absolute value no greater than $r!m^r$ where m is the value of the largest coefficient in X .*

Proof. By Lemma 4.2.3, we know we can get $[A|X]$ in the form

$$A' = \left[\begin{array}{c|c} D' & X_t \\ \hline 0 & X_b \end{array} \right]$$

where

1. $|\det([A|X])| = |\det(A')|$;

2. D' is a square matrix in upper triangular form with non-zero entries on the diagonal; and
3. the matrix X_b is an r -polynomial Vandermonde matrix.

Note that $\det(A') = \det(D') \cdot \det(X_b)$ and $|\det(D')| = 1$ as A is totally unimodular. Thus $\det(A') = 0$ if and only if $\det(X_b) = 0$. As X_b is an r -polynomial Vandermonde matrix, it follows by Lemma 4.2.2 that $\det(X_b)$ is a non-zero monic polynomial with degree no greater than r^3 . Thus $\det(A')$ is a non-zero absolutely monic polynomial of degree no greater than r^3 .

Now consider the coefficients in the determinant of $[A|X]$. The determinant is a sum of $r!$ products of r polynomials. As m is the maximum size of a coefficient in $[A|X]$, the absolute value of a coefficient in each product can be no greater than m^r . As there are $r!$ products, the maximum size of a coefficient in $\det[A'|X]$ can therefore be no greater than $r!m^r$. $\square$

We now have all we need to move on to specific cases of the basis counting problem. We begin with showing it is $\#P$ -hard to count the number of bases of matroids representable over fields of characteristic 0.

CHAR-0 $\#Bases$

INSTANCE: A matrix representation of a matroid M over a fixed field of characteristic 0.

QUESTION: How many bases does M have?

We need to add one caveat to this as not all fields of characteristic 0 can be worked with in polynomial time by a Turing machine. For example, certain real numbers may require an infinite binary string to represent them and thus cannot be used as input. Moreover, if the field operations are not polynomial time, then even deciding if a set of columns is a basis will likely be hard.

Note that all fields of characteristic 0 contain the rationals as a subfield. Suppose it is $\#P$ -complete to count bases of matroids representable over some subfield of a field F . Then it follows that it is $\#P$ -hard to count bases of matroids representable over F . Thus, if it is $\#P$ -complete to count bases of matroids representable over the rationals, then it is $\#P$ -hard to count bases of matroids representable over any fixed field with characteristic 0. Furthermore, if F can be described to a Turing machine and operations are in polynomial time, then it is $\#P$ -complete to count bases of matroids representable over F .

Lemma 4.2.5. *Assume M is a rational representable matroid with a totally unimodular representation $M[A]$ where $A = [I_r|C]$. Let X be an $r \times r$ r -polynomial Vandermonde matrix where $X_{i,j} = x^{ij}$. Furthermore, let X' be the matrix obtained by substituting x with the rational number $(r! + 1)$. Then $M' = M[A|X']$ is the rational representable matroid obtained by adding r elements freely to M .*

Proof. Let $[A'|X'']$ be an $r \times r$ submatrix of $[A|X]$, where A' is a linearly independent subset of columns of A and X'' is a subset of columns of X . From Lemma 4.2.4 we know that $\det[A'|X'']$ is a non-zero absolutely monic polynomial of degree less than r^3 and coefficients of absolute size no greater than $n = r!m^r = r!$ as all coefficients in X are 1. Note that $\sum_{i=1}^k n \cdot (n+1)^i = (n+1)^{k+1} - 1$ for all $k > 0$. Thus if we substitute $x = n+1$ into the polynomial corresponding to the determinant of $[A'|X'']$, then the absolute value of the largest power is larger than the rest of the polynomial. Thus there can be no cancellation and therefore the determinant of any $r \times r$ submatrix of $[A|X']$ is non-zero if the columns from A are linearly independent. Therefore $M' = M[A|X']$ is a rational representation of the matroid obtained by adding r elements freely to M . $\square$

Lemma 4.2.6. *The matrix $[A|X']$ in Lemma 4.2.5 can be constructed in polynomial time given the totally unimodular matrix $A = [I_r|C]$.*

Proof. To show this, all we need is that the size of $(r!m^r + 1)^{r^2}$ is polynomial in terms of $\max(r + |C|, \log(m))$ where $|C|$ is the number of columns in C . The size of $(r!m^r + 1)^{r^2}$ is

$$\begin{aligned} \log((r!m^r + 1)^{r^2}) &= r^2 \log(r!m^r + 1) < r^2 \log((rm)^r + 1) \\ &< r^2 \log((2rm)^r) = r^3 \log(2rm). \end{aligned}$$

This is polynomial in $\max(r + |C|, \log(m))$. Therefore the matrix $[A|X']$ can be constructed in polynomial time. $\square$

Theorem 4.2.7. *It is $\#P$ -complete to count the number of bases of a matroid representable over the rationals.*

Proof. This will be done from a reduction of $\#Forests$. Let G be a graph for which we want to count the number of forests. Without loss of generality we can assume that G is connected. We can construct a totally unimodular representation A of the rank $r = |V(G)| - 1$ cycle matroid M of G in polynomial time. Then the

number of forests of G is the sum of the number of independent sets of size k for $k = \{0, \dots, r\}$ of M . Now construct the matrix $[A|X']$ from Lemma 4.2.5 and let $M' = M[A|X']$.

Let M_k be the matroid obtained from M' by $k \in \{0, \dots, r\}$ truncations. Note that $M_0 = M$. Representations for these matroids can be constructed from $[A|X']$ by simply contracting the first k columns of X' in $[A|X']$ and deleting the remaining $r - k$ columns of X' . Then the number of independent sets of size k in M is the number of bases of the matroid M_{r-k} . From Lemma 4.2.5, we know that a rational representation of M' and thus M_k can be obtained in polynomial time. Therefore it is #P-complete to count bases of matroids representable over the rationals. $\square$

Corollary 4.2.8. *Char-0 #Bases is #P-hard.*

Proof. As any field of characteristic 0 contains the rationals as a subfield, it follows that the basis counting problem on matroids representable over a fixed field of characteristic 0 is #P-hard. $\square$

Note that if the fixed field in question can be described to a Turing machine and worked with in polynomial time, then we can replace #P-hard with #P-complete.

If we are working over a finite field of large enough size then the above reduction may still work. However we cannot fix a finite field and then use the above result as there will always be cases where the fixed finite field is not large enough to add elements freely by the above method.

This covers the case of counting bases in matroids representable over fixed fields of characteristic 0. We now move on to the case of counting bases in matroids representable over fields of fixed characteristic.

FIXED CHAR-P #BASES

INSTANCE: A representation of a matroid M over some field of characteristic p .

QUESTION: How many bases does M have?

Our method for showing that this problem is #P-complete will be similar to the one used to show that Char-0 #Bases is #P-hard. We will modify the matrix X where $X_{i,j} = x^{ij}$ in a way that creates a representation for a matroid obtained by adding elements freely to a representable matroid. Using this construction, we can then produce a reduction from the forest counting problem to the problem Fixed Char-p #Bases. To do this, our construction must produce a matroid representable over an appropriate field. The required construction will be given by Lemma 4.2.9.

We will treat elements of the fields $\text{GF}(p^k)$ as polynomials in the variable x with coefficients in $\text{GF}(p)$ and maximum degree $k - 1$ modulo some irreducible polynomial of degree k . If $f \in \text{GF}(p^k)$, then $\deg(f)$ is the degree of f when considered as a polynomial. For example, the elements of $\text{GF}(4)$ are $\{0, 1, x, x+1\}$ modulo $x^2 + x + 1$. We will be interested in the fields $\text{GF}(p^{r^3})$. Let $g_{p,r}(x)$ be the degree r^3 polynomial such that multiplication in $\text{GF}(p^{r^3})$ is reduced modulo $g_{p,r}(x)$. Moreover, let $\phi_{p,r} : \mathbb{Z}[x] \rightarrow \text{GF}(p^{r^3})$ be the homomorphism

$$\begin{aligned} \phi_{p,r}(\alpha_0 + \alpha_1 x^1 + \dots + \alpha_n x^n) = \\ ((\alpha_0 \bmod p) + \dots + (\alpha_n \bmod p)x^n) \bmod g_{p,r}(x). \end{aligned}$$

Lemma 4.2.9. *Let $A = [I_r | C]$ be a totally unimodular matrix over $\mathbb{Z}[x]$ and let X be the $r \times r$ r -polynomial Vandermonde matrix where $X_{i,j} = x^{ij}$. If $M = M[A]$, then $\phi_{p,r}([A|X])$ is the $\text{GF}(p^{r^3})$ representation of the matroid obtained by adding r elements freely to M .*

Proof. From Lemma 4.2.2, we see that $\det[X]$ is a non-zero monic polynomial with degree less than r^3 . Thus $\phi_{p,r}(\det[X])$ is a non-zero element of $\text{GF}(p^{r^3})$ and therefore the columns in X are all linearly independent. Let $N' = [A' | X']$ be some $r \times r$ square submatrix of $[A|X]$ where A' is a linearly independent subset of columns of A and X' is a submatrix of X . It follows from Lemma 4.2.4 that $\det(N')$ is a non-zero absolutely monic polynomial of degree less than r^3 . Thus $\phi_{p,r}(\det[N'])$ is a non-zero element of the field $\text{GF}(p^{r^3})$. As this holds for all possible N' and $\phi_{p,r}(\det[X]) \neq 0$, we see that $\phi_{p,r}([A|X])$ is the $\text{GF}(p^{r^3})$ representation of the matroid obtained by adding r elements freely to M . $\square$

We now have a method of creating representations for matroids obtained by adding elements freely to representable matroids such that the created representation is over a field with the same characteristic. We will now use this in a similar reduction to that of Theorem 4.2.7 to show that Fixed Char- p #Bases is #P-complete.

Theorem 4.2.10. *Fixed Char- p #Bases is #P-complete.*

Proof. Let G be an instance of the forest counting problem on graphs and let M be the rank r cycle matroid of G . Suppose we can count bases of matroids representable over fields of characteristic p . We can create a totally unimodular representation A of M over the field $\mathbb{Z}[x]$ in polynomial time. Now create the $\mathbb{Z}[x]$

matrix $[A|X]$ where $X_{i,j} = x^{ij}$. By Lemma 4.2.9, the matroid M' represented by the $\text{GF}(p^{r^3})$ matrix $\phi_{p,r}([A|X])$ is isomorphic to the matroid obtained by adding r elements freely to M .

Now by contracting k columns of $\phi_{p,r}(X)$ and deleting the remaining $r - k$ columns of $\phi_{p,r}(X)$ for $k \in \{0, \dots, r\}$ from M' , we obtain a representation of a matroid whose number of bases is the same as the number of independent sets of M of size $r - k$. This is just truncating M k times. Thus by doing this for $k = 0$ to $k = r$ we can count all the independent sets of M and thus the forests of G . Note that the field $\text{GF}(p^l)$ has the same characteristic as $\text{GF}(p)$ for all positive integers l . Thus it is $\#P$ -complete to count the number of bases of a representable matroid over fields of fixed characteristic. $\square$

By using a similar argument as the one used in Theorem 4.2.10, we can show that the following problem is $\#P$ -complete.

INFINITE CHAR P $\#B$ ASES

INSTANCE: A representation of a matroid M over a fixed infinite field of non-zero characteristic p .

QUESTION: How many bases does M have?

Note that for this problem, we assume that we have some way of describing the infinite field to our Turing machine. Furthermore, because the problem is defined for a fixed field, we can assume that we know all the properties of the field. In particular, we know whether or not it has a transcendental element.

Lemma 4.2.11. *Let F be an infinite field with non-zero characteristic that has a transcendental element α . Then it is $\#P$ -complete to count bases of matroids representable over F .*

Proof. Because α is transcendental, we can make a matrix X similar to the one from Theorem 4.2.10 with $\phi(X_{i,j}) = \alpha^{ij}$. We can then use the reduction from Theorem 4.2.10 to show that this is $\#P$ -complete. $\square$

A *Steinitz number* is a number of the form $N = p_1^{x_1} \cdot p_2^{x_2} \cdot \dots = \prod_{i=1}^{\infty} p_i^{x_i}$ where p_i is the i th prime and $x_i \in \{0, 1, 2, \dots, \infty\}$ [15]. This is a generalization of integers that allows for infinite numbers. For some Steinitz number N , $\text{GF}(p^N) = \bigcup_{d|N} \text{GF}(p^d)$ where $d \in \mathbb{Z}$.

Lemma 4.2.12. *Let F be an infinite field with non-zero characteristic p that has no transcendental element α . Then it is $\#P$ -complete to count bases of matroids representable over F .*

Proof. Let G be a graph for which we want to count the number of forests and r be the rank of the cycle matroid of G . We will prove this by showing that there is a set of subfields of F such that it is $\#P$ -complete to count bases of matroids representable over them. This will imply that it is $\#P$ -complete to count the number of bases representable over F . If F is infinite with no transcendental element, every element must be algebraic. Thus it must be a subfield of the algebraic closure of $\text{GF}(p)$, denoted $\overline{\text{GF}(p)}$. Brawley and Schnibben showed that all subfields of $\overline{\text{GF}(p)}$ are of the form $\text{GF}(p^N)$ for some Steinitz number N [15]. If F is infinite, then either there must be some power $x_i = \infty$ in N or there is an infinite number of x_i 's that are not equal to zero. First, suppose we have some $x_i = \infty$. Then choose some k such that $p_i^k > r^3$. This gives a subfield $F' = \text{GF}(p^{p_i^k}) \subset F$. We can now work over F' and use the reduction from Theorem 4.2.10. Thus it is $\#P$ -complete to count the number of bases of matroids representable over fields of the form $\text{GF}(p^{p_i^k})$ for some positive integer k .

Now suppose that N has an infinite number of primes p_i with $x_i \neq 0$. Let P be the set of all such primes. When given G , we can now work over the field $\text{GF}(p^{p_i})$ where $p_i \in P$ and $p_i > r^3$. We can then apply that same reduction used in Theorem 4.2.10. This shows that it is $\#P$ -complete to count the number of bases of matroids representable over the fields $\text{GF}(p^{p_i})$ where $p_i \in P$.

In either case, we have a family of subfields of F such that it is $\#P$ -complete to count the number of bases of matroids representable over them. Thus it is $\#P$ -complete to count the number of bases of matroids representable over F . $\square$

Theorem 4.2.13. *Infinite Char P $\#Bases$ is $\#P$ -complete*

Proof. An infinite field F of non-zero characteristic p either has a transcendental element or is a subfield of $\overline{\text{GF}(p)}$. Recall that as the problem is for fixed fields, we know if we have a transcendental element. If F has a transcendental element, then Lemma 4.2.11 shows that Infinite Char P $\#Bases$ is $\#P$ -complete. Alternatively, if F is a subfield of $\overline{\text{GF}(p)}$, then it follows from Lemma 4.2.12 that it is $\#P$ -complete to count bases of matroids representable over F . Therefore Infinite Char P $\#Bases$ is $\#P$ -complete. $\square$

Combining Theorems 4.2.7 and 4.2.13 we see that it is #P-complete to count the number of bases for matroids representable over any fixed infinite field. This just leaves the finite case. Theorem 4.2.10 provides a partial answer for this case. However, there is still work to be done to resolve Question 4.2.1.

4.3 Other Basis Counting Problems

We now consider a number of different variations of the basis counting problem.

Let M be a matroid and let $\{p_1, p_2, \dots, p_k\}$ be a partition of $E(M)$. Then a *partition basis* of M is a basis B where if $e \in p_i$ and $e \in B$ then $p_i \subseteq B$.

In this section we will show that the following basis counting problems are #P-complete:

- (i) Counting partition bases of $\text{GF}(q)$ representable matroids for any fixed q .
- (ii) Counting common bases of $\text{GF}(q)$ representable matroids for any fixed q .
- (iii) Counting the bases of paving matroids.
- (iv) Counting the bases of polymatroids.

#GF(q) PARTITION BASES

INSTANCE: A $\text{GF}(q)$ representation of a matroid M for a fixed q and a partition $\{p_1, p_2, \dots, p_k\}$ of the ground set.

QUESTION: How many partition bases does M have?

If each $|p_i| = 1$, then this is just the same as counting bases. To show that counting partition bases for $\text{GF}(q)$ matrices is #P-complete for $q = p^k$, we will create $\text{GF}(p)$ matrices from $\text{GF}(p^k)$ matrices such that there is a correspondence between the partition bases of the matroid over $\text{GF}(p)$ and the bases of the original matroid over $\text{GF}(p^k)$. We will then show that if we have an algorithm $\mathcal{A}$ for counting the partition bases for matroids representable over $\text{GF}(p)$ then we could use this algorithm to count the bases of matroids representable over fields of fixed characteristic p . This was shown to be #P-complete in Theorem 4.2.10.

Recall that we treat elements of the field $\text{GF}(p^k)$ as polynomials with coefficients in $\text{GF}(p)$ and maximum degree $k - 1$.

Suppose we are given a $r \times n$ matrix representation $A = [c_1, c_2, \dots, c_n]$ of a matroid over the field $GF(p^k)$. Then we will construct the following $rk \times nk$ $GF(p)$ matrix. From each column $c = [g_1(x), g_2(x), \dots, g_r(x)]^T$ of A , let $\alpha_{w,y,z}$ be the coefficient of x^z in $x^y \cdot g_w(x)$. For each column c we will make the $rk \times k$ matrix $P(c)$ with entries in $GF(p)$. We will describe how we calculate the entries of $P(c)$ and follow this up with an example. For the matrix $P(c)$, we have $P(c)_{i,j} = \alpha_{m+1,j-1,l}$, where $m = i \bmod r$ and l is the quotient of i/r . This definition is a bit of a mouthful, so we will go over an example before moving on. Consider the column $c = [x+1, x, 2]^T$ over the field $GF(3^2)$. As we are working over $GF(3^2)$ our multiplication will be mod x^2+1 . The matrix $P(c)$ can be constructed as follows. We have $p=3$ and $k=2$ so $P(c)$ will be a 6×2 matrix. As $k=2$, we will construct the columns x^0c and x^1c . When we do this, we get $x^0c = [x+1, x, 2]^T = c$ and $x^1c = [x+2, 2, 2x]^T$. We will combine these into a 6×2 matrix M where the coefficients we are interested in are in bold.

$$M = \begin{bmatrix} 1x + \mathbf{1} & 1x + \mathbf{2} \\ 1x + \mathbf{0} & 0x + \mathbf{2} \\ 0x + \mathbf{2} & 2x + \mathbf{0} \\ \mathbf{1}x + 1 & \mathbf{1}x + 2 \\ \mathbf{1}x + 0 & \mathbf{0}x + 2 \\ \mathbf{0}x + 2 & \mathbf{2}x + 0 \end{bmatrix}$$

From this we see that the first column of $P(c)$ will be $[1, 0, 2, 1, 1, 0]^T$ where the first three entries correspond to the coefficients of x^0 in x^0c and the next three entries correspond to the coefficient of x^1 in x^0c . Similarly, the second column will be $[2, 2, 0, 1, 0, 2]^T$ where the first three entries correspond to the coefficient of x^0 in x^1c and the next three entries correspond to coefficients of x^1 in x^1c . This gives us the following matrix.

$$P(c) = \begin{bmatrix} 1 & 2 \\ 0 & 2 \\ 2 & 0 \\ 1 & 1 \\ 1 & 0 \\ 0 & 2 \end{bmatrix}$$

This is how we construct the matrix $P(c)$. We will denote the i th column

of $P(c)$ by $P(c, i)$. Note that $P(c, i) = P(x^{i-1}c, 1)$. Now for the $\text{GF}(p^k)$ matrix $A = [c_1, c_2, \dots, c_n]$, create the $rk \times nk$ $\text{GF}(p)$ matrix A' by combining the matrices $P(c_i)$ created for each column of A . That is, $A' = [P(c_1)|P(c_2)|\dots|P(c_n)]$. The idea of this matrix A' is to create a $\text{GF}(p)$ matrix from a $\text{GF}(p^k)$ matrix A such that A' in some way simulates some of the dependencies of A . In particular, there is a relationship between the bases of $M[A]$ and the partition bases of $M[A']$.

Before we make use of this construction, we will prove some basic properties of the matrix $P(c)$.

Lemma 4.3.1. *For $\alpha \in \text{GF}(p)$ and the column c over the field $\text{GF}(p^k)$, $\alpha P(c, i) = P(\alpha c, i)$.*

Proof. This will be true if and only if $\alpha P(c, 1) = P(\alpha c, 1)$ as if this is true for $P(c, 1)$, then

$$\alpha P(c, i) = \alpha P(x^{i-1}c, 1) = P(\alpha(x^{i-1}c), 1) = P(x^{i-1}(\alpha c), 1) = P(\alpha c, i).$$

Suppose c is a column of r entries. Then consider the first r entries of $P(c, 1)$. Suppose that for some $j \in \{0, \dots, r-1\}$, the j th entry of c is $\alpha_0 + \alpha_1 x + \dots + \alpha_k x^k$. Then the j th entry of $P(c, 1)$ will be α_0 . Then we see that the j th entry of $\alpha P(c, 1)$ is $\alpha \alpha_0$. Now consider the column αc . The j th entry of αc will be $\alpha \alpha_0 + \alpha \alpha_1 x + \dots + \alpha \alpha_k x^k$ and it follows that the j th entry of $P(\alpha c, 1)$ is $\alpha \alpha_0$. Thus, for the first r entries, we see that $\alpha P(c, 1) = P(\alpha c, 1)$.

Now consider the $(j+r)$ th entry of $P(c, 1)$. As the j th entry of c is $\alpha_0 + \alpha_1 x + \dots + \alpha_k x^k$, the $(j+r)$ th entry of $P(c, 1)$ will be α_1 . Then we see that the $(j+r)$ th entry of $\alpha P(c, 1)$ is $\alpha \alpha_1$. Furthermore, the j th entry of αc is $\alpha \alpha_0 + \alpha \alpha_1 x + \dots + \alpha \alpha_k x^k$ and so the $(j+r)$ th entry of $P(\alpha c, 1)$ will be $\alpha \alpha_1$. Again for the next r entries we have $\alpha P(c, 1) = P(\alpha c, 1)$.

Using the same argument, we can show that for the remaining entries $\alpha P(c, 1) = P(\alpha c, 1)$. $\square$

Lemma 4.3.1 shows that $P(c, i)$ is linear in the first coordinate.

Lemma 4.3.2. *If $c_1 + c_2 = c_3$, then $P(c_1, i) + P(c_2, i) = P(c_3, i)$ for all $i \in \{0, \dots, k-1\}$*

Proof. This will be satisfied if $P(c_1, 1) + P(c_2, 1) = P(c_3, 1)$ as $P(c, i) = P(x^{i-1}c, 1)$ for all columns c and $c_1 + c_2 = c_3$ if and only if $x^{i-1}c_1 + x^{i-1}c_2 = x^{i-1}c_3$ for

$i \in \{1, \dots, k\}$. Suppose that c_1, c_2 and c_3 are columns of r entries. Let $\gamma_{a,b,d}$ be the coefficient of x^d in b th entry of c_a . Now suppose that in row j we have

$$P(c_1, 1) + P(c_2, 1) \neq P(c_3, 1)$$

where $j = rl + m$ and $m \in \{0, \dots, r-1\}$. Then we see that $\gamma_{1,m,l} + \gamma_{2,m,l} \neq \gamma_{3,m,l}$. However, this would imply that in the m th row of c_1, c_2 and c_3 we have $c_1 + c_2 \neq c_3$ and is hence a contradiction. Therefore if $c_1 + c_2 = c_3$, then $P(c_1, i) + P(c_2, i) = P(c_3, i)$ for all $i \in \{0, \dots, k-1\}$ $\square$

Corollary 4.3.3. $\sum_{l=1}^j \alpha_l P(c_l, i) = P\left(\sum_{l=1}^j \alpha_l c_l, i\right).$

Proof. From Lemma 4.3.1 we have

$$\sum_{l=1}^j \alpha_l P(c_l, i) = \sum_{l=1}^j P(\alpha_l c_l, i).$$

Furthermore, from Lemma 4.3.2 it follows that

$$\sum_{l=1}^j P(\alpha_l c_l, i) = P\left(\sum_{l=1}^j \alpha_l c_l, i\right).$$

$\square$

Lemma 4.3.4. Let $g(x) = \alpha_0 + \dots + \alpha_{k-1}x^{k-1}$ be some $\text{GF}(p^k)$ polynomial. Then

$$P(g(x)c, 1) = \sum_{i=0}^{k-1} \alpha_i (P(c, i+1)).$$

Proof. Note that $P(x^i c, 1) = P(c, i+1)$ for all columns c . Then we see that

$$P(g(x)c, 1) = P\left(\sum_{i=0}^{k-1} \alpha_i x^i c, 1\right)$$

and from Corollary 4.3.3,

$$P\left(\sum_{i=0}^{k-1} \alpha_i x^i c, 1\right) = \sum_{i=0}^{k-1} \alpha_i P(x^i c, 1) = \sum_{i=0}^{k-1} \alpha_i P(c, i+1).$$

$\square$

Lemma 4.3.5. *Let A' be the $rk \times nk$ matrix constructed from some $r \times n$ matrix A over the field $GF(p^k)$ using the above method. The subset $C = \{c_1, c_2, \dots, c_m\}$ is a circuit of the matroid $M = M[A]$ if and only if there exists a circuit in the matroid $M' = M[A']$ that contains at least one element from each of $P(c_1), P(c_2), \dots, P(c_m)$ and no other elements from any other $P(c_i)$.*

Proof. By relabelling and row swapping, we can assume without loss of generality, if the circuit we are interested in has size m , then the circuit contains elements corresponding to the first m columns of A . Suppose we have the circuit $C = \{c_1, c_2, \dots, c_m\}$. As C is a circuit, we know that $g_1(x)c_1 + g_2(x)c_2 + \dots + g_m(x)c_m$ is a column of 0's, denoted $\bar{0}$, for some $GF(p^k)$ polynomials $g_j(x)$. That is $P(g_1(x)c_1 + g_2(x)c_2 + \dots + g_m(x)c_m, i)$ is a column of 0's for all i . It follows from Corollary 4.3.3 and Lemma 4.3.4 that

$$\begin{aligned} \bar{0} = P(g_1(x)c_1 + g_2(x)c_2 + \dots + g_m(x)c_m, 1) &= \sum_{j=1}^m P(g_j(x)c_j, 1) \\ &= \sum_{j=1}^m \left(\sum_{i=0}^{k-1} \alpha_{j,i} P(c_j, i+1) \right), \end{aligned}$$

where $\alpha_{j,i}$ is the coefficient of x_i in $g_j(x)$.

Thus the columns in $P(c_1), \dots, P(c_m)$ must contain a circuit. Therefore there exists a circuit that only contains elements from $P(c_1), P(c_2), \dots, P(c_m)$.

Now suppose we have a circuit containing elements from $P(c_1), P(c_2), \dots, P(c_m)$. Then as A' is a $GF(p)$ matrix, there exists constants $\alpha_{j,i}$ such that

$$\sum_{j=1}^m \left(\sum_{i=0}^{k-1} \alpha_{j,i} P(c_j, i+1) \right) = \bar{0}.$$

However, from Lemmas 4.3.3 and 4.3.4 it follows from this that there exists $GF(p^k)$ polynomials $g_j(x)$ such that

$$\sum_{i=0}^{k-1} \alpha_{j,i} P(c_j, i+1) = P(g_j(x)c_j, 1).$$

Then this implies that there exist $GF(p^k)$ polynomials such that

$$\sum_{j=1}^m P(g_j(x)c_j, 1) = \bar{0}.$$

Thus there exists a circuit contained in $\{c_1, \dots, c_m\}$.

So suppose $C = \{c_1, \dots, c_m\}$ is a circuit of M . Then there exists a circuit of M' contained in $P(c_1), \dots, P(c_m)$. Suppose that there is a $P(c_i)$ for $i \in \{1, \dots, m\}$ that does not contain an element in this circuit. Then without loss of generality, we can assume that $i = m$ and therefore $\{P(c_1), \dots, P(c_{m-1})\}$ contains a circuit. But then, from the above argument we see that $\{c_1, \dots, c_{m-1}\}$ contains a circuit. This is a contradiction as this implies that $C = \{c_1, \dots, c_m\}$ contains another circuit. Therefore it follows that $C = \{c_1, c_2, \dots, c_m\}$ is a circuit if and only if there exists a circuit C' in M' such that C' contains at least one element from each of $P(c_1), P(c_2), \dots, P(c_m)$ and no other elements from any other $P(c_i)$. $\square$

This covers all the required properties of the matrix $P(c)$. We will use $P(c)$ to show that it is $\#P$ -complete to count partition bases of a representable matroid over any fixed field $\text{GF}(q)$.

Lemma 4.3.6. *$\# \text{GF}(q)$ partition bases is $\#P$ -complete.*

Proof. This will be a reduction from counting bases of matroids with fixed characteristic p , which was shown to be $\#P$ -complete in Theorem 4.2.10. Let A be the $r \times n$ $\text{GF}(p^k)$ representation of the matroid we want to count bases for. Create the matrix $A' = [P(c_1)|P(c_2)|\dots|P(c_n)]$ with partitions $P(c_i)$. From Lemma 4.3.5, we know that $\{c_{j_1}, \dots, c_{j_m}\}$ is a circuit if and only if $P(c_{j_1}), \dots, P(c_{j_m})$ is a minimal set of partitions that contains a circuit. Therefore, if $\{c_{b_1}, \dots, c_{b_r}\}$ is a basis of $M[A]$, then $\{P(c_{b_1}), \dots, P(c_{b_r})\}$ is independent in $M[A']$. Furthermore $M[A']$ has rank rk , so $\{P(c_{b_1}), \dots, P(c_{b_r})\}$ is a basis if and only if $\{c_{b_1}, \dots, c_{b_r}\}$ is a basis of $M[A]$. Thus the number of bases of $M[A]$ is the number of partition bases of $M[A']$. It follows that as we can create A' in polynomial time given A , if we have an algorithm $\mathcal{A}$ for counting the number of partition bases of matroids representable over $\text{GF}(p)$, we can use $\mathcal{A}$ to count the number of bases of a matroid representable over a field of fixed characteristic p . Since the basis counting problem is $\#P$ -complete for matroids representable over fields of fixed characteristic p , counting the number of partition bases of $\text{GF}(p)$ matroids is also $\#P$ -complete. As $\text{GF}(p)$ is a subfield of $\text{GF}(p^k)$ it follows that it is $\#P$ -complete to count partition bases for $\text{GF}(q)$ representable matroids where $q = p^k$. $\square$

Now suppose that we have two $\text{GF}(q)$ matroids on the same ground set and we are interested in counting their common bases.

#COMMON $\text{GF}(q)$ BASES

INSTANCE: Two $\text{GF}(q)$ matroids M_1 and M_2 with a common ground set and their respective matrix representations over a fixed field $\text{GF}(q)$.

QUESTION: How many common bases do M_1 and M_2 have?

Given two $\text{GF}(q)$ matroids on the same ground set, we can find a common basis in polynomial time [49]. If instead we are given three matroids and asked for a common basis, then this becomes NP-complete. This can be shown by a simple reductions from 3 dimensional matching. We will show that despite being able to find a common basis for two matroids, it is #P-complete to count the number of common bases. A *matching* of a graph is a set of edges M such that no vertex is incident with two edges in M . A *perfect matching* is a matching such that every vertex of the graph is incident with an edge in the matching. We will use a reduction from the #P-complete problem of counting perfect matchings in bipartite graphs [71] to show that this is #P-complete.

#PERFECT MATCHINGS

INSTANCE: A bipartite graph G .

QUESTION: How many perfect matchings does G have?

Lemma 4.3.7. *#Common $\text{GF}(q)$ Bases is #P-complete.*

Proof. Suppose we are given a bipartite graph $G = (\mathcal{E}(G), U, V)$ for which we want to count perfect matchings. We can assume without loss of generality that $|U| = |V|$ and that G has at least one perfect matching. We can associate this with two $\text{GF}(q)$ representable matroids with ground set $\mathcal{E}(G)$ by doing the following. The set of circuits of M_1 will be all pairs of edges that are incident with a common vertex in U . The matrix representation of M_1 will be just the identity with a number of parallel elements. Therefore, it is representable over any field $\text{GF}(q)$. Similarly, the circuits of M_2 will be all pairs of edges that are incident with a common vertex in V . Again, M_2 is representable over all fields $\text{GF}(q)$. Now we see that a basis of M_1 will be a set of $|U|$ edges such that no two are incident with a common vertex in U and a basis of M_2 will be a set of $|V|$ edges such that no two are incident with a common vertex in V . Thus a common basis of M_1 and M_2 will correspond to a unique perfect matching of G .

Now take any perfect matching. This will contain $|U|$ edges. Furthermore, no two edges are incident with a common vertex. So this set will be independent in both M_1 and M_2 . As both M_1 and M_2 have rank $|U|$, it follows that a perfect matching produces a unique common basis. Therefore the number of perfect matchings of G is the same as the number of common bases of M_1 and M_2 . As counting perfect matchings is $\#P$ -complete, counting common bases of $\text{GF}(q)$ representable matroids must also be $\#P$ -complete. $\square$

Recall that a *sparse paving matroid* is a matroid in which all its non-spanning circuits are hyperplanes. Mark Jerrum showed that it is $\#P$ -complete to count bases of sparse paving matroids when the input is a graph, the ground set is the edges and the circuit hyperplanes are the Hamiltonian cycles of the graph [43]. Below is a proof that it is $\#P$ -complete to count the number of bases of sparse paving matroids for a different method of input. The input we use will be a bipartite graph with independent sets corresponding to sets of edges that do not contain a perfect matching. We will use a reduction from $\#P$ Perfect Matchings on simple bipartite graphs to show that this is $\#P$ -complete. This follows from [71] that it is $\#P$ -complete to count perfect matchings in a simple bipartite graph. However, we will provide an alternative proof that counting perfect matchings of simple bipartite graphs is $\#P$ -complete.

Lemma 4.3.8. *$\#P$ Perfect Matchings is $\#P$ -complete when restricted to simple bipartite graphs.*

Proof. Let $G = (\mathcal{E}(G), U, V)$ be a bipartite graph (not necessarily simple) for which we wish to count perfect matchings. Because G is bipartite, we know that G has no loops. Create the graph $G' = (\mathcal{E}(G'), U', V')$ by taking each edge $e = (u, v) \in \mathcal{E}(G)$ and replacing it with the path $u, a_e, x_e, b_e, y_e, c_e, v$. Note that $U' = U \cup \{y_e | e \in \mathcal{E}(G)\}$, $|U'| = |U| + |\mathcal{E}(G)|$ and that G' is simple. Now suppose that M is a set of edges that is a perfect matching of G . Then let M' be the set of edges of G' defined by $\{\{a_e, c_e\} | e \in M\} \cup \{\{b_e\} | e \notin M\}$. Suppose M' is not a matching. Then there must exist two edges in M' that are incident with a common vertex. By the construction, these two edges must be a_e and $a_{e'}$ or c_e and $c_{e'}$ for two edges e and e' that are incident in G . But then we see that $e, e' \in M$ and M is therefore not a matching. This is a contradiction and therefore M' is a matching of G' . Furthermore,

$$|M'| = 2|M| + (|\mathcal{E}(G)| - |M|) = 2|U| + (|\mathcal{E}(G)| - |U|) = |U| + |\mathcal{E}|.$$

Therefore M' is a perfect matching.

Now suppose we have some perfect matching M' of G' . If $a_e \in M'$ then we cannot also have $b_e \in M'$ as they are incident with a common vertex. Therefore we must have $c_e \in M'$. Otherwise the vertex y_e will not be incident with a edge in M' . Now let M'' be the set of edges of G such that $e \in M''$ if and only if both a_e and c_e are in M' . Suppose we have two edges $e, e' \in M''$ that are incident to a common vertex. Then the edges a_e and $a_{e'}$ or c_e and $c_{e'}$ must be incident with a common vertex in M' . But M' is a perfect matching so this is not possible. Therefore no two edges in M'' are incident with a common vertex. Thus M'' is a matching of G . Assume that we have some vertex $v \in V$ that is not incident with any edge in M'' . Then there cannot be any edges incident with v in M' which again is a contradiction as M' is a perfect matching. Thus M'' is a perfect matching.

Therefore the number of perfect matchings of G is the same as the number of perfect matching of G' . As G' is simple, this shows that when restricted to simple graphs, #Perfect Matchings is #P-complete. $\square$

Theorem 4.3.9. *It is #P-complete to count bases of paving matroids.*

Proof. Take a simple bipartite graph $G = (\mathcal{E}(G), U, V)$ for which we want to count perfect matchings. We can assume that without loss of generality $|U| = |V|$, $|U| < |\mathcal{E}(G)|$ and that G has at least one perfect matching. Define the set $\mathcal{B}$ to be all sets of $|U|$ edges that are not a perfect matching. We will show that $\mathcal{B}$ is the set of bases of a sparse paving matroid. As long as G has more than $|U|$ edges, there will exist at least one set of $|U|$ edges that are not a perfect matching. Take any two sets of $|U|$ edges in $\mathcal{B}$, denoted B_1 and B_2 , and let $x \in B_1 - B_2$. Suppose that there is no $y \in B_2 - B_1$ such that $B_1 - x + y$ is not a perfect matching. Then $B_1 - x$ must correspond to a matching containing $2|U| - 2$ vertices and for all $y \in B_2 - B_1$, y must correspond to an edge between the two vertices not incident with an edge corresponding to an element in $B_1 - x$. Because B_2 contains no parallel edges (as G is simple), there is at most a unique $y \in B_2 - B_1$ that is incident with the two vertices that are not incident with an edge that corresponds to an element in $B_1 - x$. But then $B_2 - B_1 = y$ and $B_1 - B_2 = x$. Thus $B_2 = B_1 - x + y$. If this is the case, then if $B_1 - x + y$ corresponds to a perfect matching then so does B_2 and we have a contradiction. Thus $\mathcal{B}$ is the set of bases of a matroid M . Any circuit of M must contain a perfect matching and thus must have at least $r(M)$ elements. Take a non spanning circuit C of M . Then $|C| = r(M)$ and therefore C must correspond to a perfect matching of G . Adding any edge to this set will give

a set of size $r(M) + 1$ that contains a set of $r(M)$ edges that do not correspond to a perfect matching. This increases the rank and thus C is a hyperplane. Therefore $\mathcal{B}$ is the set of bases of a sparse paving matroid.

Now, there are $\binom{|\mathcal{E}(G)|}{|U|}$ sets of edges of size $|U|$. If such a set is not a perfect matching then it corresponds a basis of M . Moreover, if a set of $|U|$ elements is not a basis, then it corresponds to a perfect matching of G . That is $PM = \binom{|\mathcal{E}(G)|}{|U|} - |\mathcal{B}|$ where PM is the number of perfect matchings of G . As it is #P-complete to count perfect matching of simple bipartite graphs, it must also be #P-complete to count bases of sparse paving matroids. $\square$

Given a ground set E , a function f is *submodular* if

$$f(A \cap B) + f(A \cup B) \leq f(A) + f(B)$$

for all $A, B \subseteq E$. Such a function is *increasing* if $f(A) \leq f(B)$ for all $A \subseteq B \subseteq E$. Let E be some ground set and f be some increasing submodular function such that $f(\emptyset) = 0$. Then f is the rank function of the *polymatroid* with ground set E . Such a polymatroid is a *k-polymatroid* if $f(e) \leq k$ for all $e \in E$. Let r be the rank function of a matroid and let $f(A) = r(A) + k$. Then we say that the polymatroid with rank function f is a *strict k-polymatroid*. We will denote the strict polymatroid made by increasing the rank of every set of the matroid M by k as M_k . A basis of a polymatroid is a set A such that $|A| = f(A)$. We will show that it is #P-complete to count the bases of strict polymatroids.

#GF(q) STRICT K-POLY BASES

INSTANCE: An integer k and a representation of a matroid M over a GF(q) for a fixed q .

QUESTION: How many bases does M_k have?

Theorem 4.3.10. #GF(q) Strict k -Poly Bases is #P-complete.

Proof. Now suppose we are given a graph G and asked to count its forests. This is the same as counting the number of independent sets of the cycle matroid M associated with G . Note that we can create a representation for M in polynomial time for any GF(q). We will take M and make the following representable polymatroids $(M^*)_0, (M^*)_1, \dots, (M^*)_{|\mathcal{E}(G)|-r(M)}$. Note that $(M^*)_0 = M^*$ and that M^* is representable over GF(q) if and only if M is. Also note that the number

of spanning sets of M^* of size m is the number of independent sets of M of size $|\mathcal{E}(G)| - m$.

Now consider a basis B of $(M^*)_k$. As B is a basis, $|B| = r(B) = r((M^*)_k) = r(M^*) + k$. Therefore, the corresponding set in M^* must have rank $r(M^*)$. Thus it is a spanning set of M^* . Furthermore, it must have size $r(M^*) + k$. Therefore the bases of $(M^*)_k$ are the spanning sets of M^* of size $r(M^*) + k$. So to calculate the number of spanning sets of M^* , all we need to do is count the number of bases of the polymatroids $(M^*)_0, (M^*)_1, \dots, (M^*)_{|\mathcal{E}(G)| - r(M)}$. As the number of spanning sets of M^* is the same as the number of independent sets of M , this would allow us to count the number of independent sets of M . Therefore counting the number of bases of representable k -polymatroids is #P-complete. $\square$

4.4 Circuit Counting

We finish this chapter with counting circuits. The problem of counting minimum sized circuits for transversal matroids was shown to be #P-complete [20]. This was used in a proof that it is #P-complete to count bases of transversal matroids [21]. It has also been shown that it is #P-complete to count the number of circuits of graphic matroids [25]. This implies that it is #P-complete to count the number of circuits of matroids representable over any field. The hope is that this or similar circuit counting results on representable matroids can be of use in showing that basis counting is #P-complete for any fixed field. Recall that we have shown that it is #P-complete to count bases of matroids representable over fixed infinite fields and fields of fixed characteristic. However, the case of fixed finite fields is still open.

Finding circuits is often required in coding theory where a circuit corresponds to a minimum weight code word. Because of this, a number of coding theory results can be applied to finding circuits. For example, it is NP-complete to find the minimum distance of a linear code [72]. This implies that it is NP-complete to find the minimum circuit size of a representable matroid. However, little is known about the counting versions of these problems. We will show that it is #P-complete to count the number of circuits of size no greater than w for a given representable matroid and integer w over a fixed field. That is,

#GF(q) CIRCUIT

INSTANCE: A matrix A over $\text{GF}(q)$ and integer w .

QUESTION: How many circuits of size at most w does $M[A]$ have?

Note that this result follows from the fact that it is $\#P$ -complete to count the number of circuits of graphic matroids [25]. However, we will provide an alternative reduction. To do so, we will first show that the following problem is $\#P$ -complete.

$\#GF(q)$ e -CIRCUIT OF SIZE w

INSTANCE: A matrix A over the fixed field $\text{GF}(q)$ with a distinguished column e and integer w .

QUESTION: How many circuits of size w does $M[A]$ have that include e ?

We will do this with a reduction from the $\#P$ -complete problem $\#3DM$ [22].

$\#3DM$

INSTANCE: A subset $U \subset X \times Y \times Z$ for some finite sets $|X| = |Y| = |Z|$.

QUESTION: How many sets $W \subset U$ are there such that $|W| = |X|$ and no two elements of W agree in any coordinate?

The reduction used will be very similar to the reduction used by Berlekamp, McEliece and Tilborg to show that COSET WEIGHTS is NP-complete [12].

Lemma 4.4.1. *$\#GF(q)$ e -circuit of size w is $\#P$ -complete.*

Proof. Let $U \subseteq X \times Y \times Z$ be an instance of 3 dimensional matching. Construct the matrix N with three sets of rows, each labelled by a single element in either X, Y or Z . Set $N(i, j)$ to 1 if element i is in triple j and set all other entries to 0. Add a final column labelled e consisting of all 1's. Now set $w = |X| + 1$. Consider some circuit C of size $w + 1$ that contains e . This will give $\sum_{i=1}^w a_i x_i = \alpha e$ where $a_i, \alpha \in \text{GF}(q)$ and x_i are columns of N . By multiplying by α^{-1} we obtain $\sum_{i=1}^w \alpha^{-1} a_i x_i = e$. As e is a column of all 1's of length $3w$ and each x_i only has 3 non-zero entries, we know that no two columns x_i and x_j can have a non-zero entry in the same row. Thus $\alpha^{-1} a_i x_i = x_i$ for all i . Therefore in any circuit C that contains e , the sum of the columns in $C \setminus e$ will be a vector of all 1's, for all fields $\text{GF}(q)$. Now suppose we have a circuit C such that $|C| = |X| + 1$ and $e \in C$. Then the elements of $C \setminus e$ will be a unique solution to the 3 dimensional matching problem.

Now suppose we have a solution S to the 3 dimensional matching problem. Note that $|S| = |X| = w - 1$. The sum of the columns in S will produce a column of all 1's and $e \notin S$. Thus $S \cup e$ is a circuit of size w of the matroid $M = M[N]$. Thus the number of circuits containing e of size w is the same as the number of 3 dimensional matchings. Therefore $\#GF(q)$ e -circuit of size w is $\#P$ -complete. $\square$

This shows that counting circuits of size w containing a given element is $\#P$ -complete. However, we are interested in counting all circuits, not just ones containing a certain element. We will use a reduction from $\#GF(q)$ e -circuit of size w to the following problem, showing that it remains $\#P$ -complete when we are looking for all circuits of size w .

$\#GF(q)$ CIRCUIT OF SIZE w

INSTANCE: A matrix A over the fixed field $GF(q)$ and integer w .

QUESTION: How many circuits of size w does $M[A]$ have?

Lemma 4.4.2. *$\#GF(q)$ Circuit of size w is $\#P$ -complete.*

Proof. Suppose we have a polynomial time algorithm $\mathcal{A}$ for solving $\#GF(q)$ circuit of size w . Take an instance of $\#GF(q)$ e -circuit of size w for the matroid M . Let $sol(M)$ be the solution produced by $\mathcal{A}$ for matroid M . Then $sol(M) - sol(M \setminus e)$ will be the number of circuits of M of size w that contain e . Therefore $\#GF(q)$ circuit of size w is $\#P$ -complete. $\square$

Lemma 4.4.3. *$\#GF(q)$ circuit is $\#P$ -complete.*

Proof. Take an instance of $\#GF(q)$ circuit of size w for a matroid M and suppose we have a polynomial time algorithm $\mathcal{A}$ for solving $\#GF(q)$ circuit. Let $sol(M, w)$ be the solution produced by $\mathcal{A}$ for the matroid M with integer w . Then $sol(M, w) - sol(M, w - 1)$ will be the number of circuits of M with size w . Thus $\#GF(q)$ circuit is $\#P$ -complete. $\square$

So we see that it is $\#P$ -complete to count all circuits of size no greater than w for representable matroids.

Our last result will be on the difficulty of deciding if a representable matroid is uniform. While this is not circuit counting, the proof is a reduction from a related problem in coding theory. A matroid is a *free spike* if it has ground set $\{e_1, f_1, e_2, f_2, \dots, e_n, f_n\}$ and set of circuits $\mathcal{C} = \{\{e_i, f_i, e_j, f_j\} | 1 \leq i < j \leq n\}$.

A matroid is a *spike* if its set of circuits is $\mathcal{C}$ plus some additional circuits of size n containing one of e_i and f_i for each $1 \leq i \leq n$. Hlineny showed that given a representation of a spike over some infinite field, it is NP-hard to decide if the matroid is the free spike [39]. This is similar to the problem of deciding if a matroid is a uniform matroid as we are trying to decide if the matroid in question has a circuit of size $r(M)$. We note that Hlineny's result was for infinite fields, while we consider the finite case.

$\text{GF}(q)$ UNIFORM

INSTANCE: A matrix A over the field $\text{GF}(q)$ for some fixed prime power q .

QUESTION: Is $M[A]$ uniform?

We will use a reduction from the known NP-complete problem MDS code [72].

MDS CODE

INSTANCE: A fixed prime power $q \geq 2$, positive integers n, r and an $r \times n$ matrix H over $\text{GF}(q)$.

QUESTION: Is there a non-zero vector x of length n over $\text{GF}(q)$, such that $Hx^t = 0$ and $\text{wt}(x) \leq r$?

Lemma 4.4.4. $\text{GF}(q)$ Uniform is NP-hard.

Proof. Let H, r, n and q be an instance of MDS CODE. We can assume without loss of generality that H has rank r . Otherwise there will always be such a vector x . Note that such a vector x exists if and only if $M[H]$ has a non-spanning circuit. Which in turn exists if and only if $M[H]$ is not uniform. Thus $\text{GF}(q)$ UNIFORM is NP-hard. $\square$

Chapter 5

Nonsuccinct Descriptions

5.1 Introduction

Historically the approach to complexity theory in matroids has taken one of two approaches. The first is to describe the matroid via some sort of oracle, such as an independence oracle. That is, to use an oracle Turing machine instead of a deterministic Turing machine. Doing so removes the possibility of NP-completeness results. However, these can still be used to get an idea of the worst case running time of a problem.

The second approach is to describe the matroid via some sort of succinct description. For example, we could describe the matroid by giving a graph or matrix representation. However, not all matroids have such a description. This does not cause problems if we are only interested in specific classes of matroids with a succinct description. However, if we are interested in all matroids, then this method of description does not work.

There is a third approach we could take. A matroid is just a structured collection of subsets. So we can describe any matroid just by listing these subsets. There is a concern that such a description could be very large and hide the complexity of problems as everything may become artificially easy. However, this may not be the case as problems have been shown to be NP-complete for some such descriptions and in P for others [55].

The effect the choice of description has on the complexity of a problem gives rise to a natural hierarchy of matroid descriptions. We will have $A \leq B$ if given description B , we can always create description A in polynomial time. Then we see that if a problem is NP-complete for description B , then it must be NP-hard

for description A . Similarly, if a problem can be solved in polynomial time given description A , then the same problem must be in P when given description B .

5.2 Additions to the Hierarchy

A hierarchy for the following 10 descriptions was created by Mayhew [55]: rank, independent sets, spanning sets, bases, flats, hyperplanes, dependent hyperplanes, circuits, nonspanning circuits, and cyclic flats. For these ten matroid descriptions, the hierarchy shown in figure 5.1 was constructed where there is an arrow (or path of arrows) from description A to description B if given A , we can always create description B in polynomial time. We note that the lattice shown is the other way up from the one in [55]. This is because this direction makes more sense when we study the hierarchy of all such matroid descriptions, which will be done in Chapter 6. This is because it is analogous to the Turing degrees where the more powerful Turing degree is at the top. We have a similar notion of the more powerful matroid description where B is more powerful than A if any problem that can be solved when given description A can also be solved when given description B .

Figure 5.1: The ordering of the 10 inputs.

We are interested in adding the following two descriptions to this hierarchy

- (i) Circuit Closures and
- (ii) Connected Cyclic flats.

A set is independent in a matroid if and only if its intersection with any circuit

closure is less than equal to the rank of the circuit closure. So the collection of circuit closures and their ranks completely determines the matroid.

Note that if a matroid has a loop, then no cyclic flat will be connected and thus the list of connected cyclic flats for the matroid will be empty. This would mean that a large number of matroids would not have a description via this description method. We get around this by defining a cyclic flat to be connected if minus loops, the cyclic flat is connected. Note that by this definition the cyclic flat of all loops is vacuously connected. We will show in the next section that the collection of connected cyclic flats does completely determine the matroid.

Determining where these two descriptions fit in the hierarchy will be done in the following lemmas.

Lemma 5.2.1. *Circuits $\geq$ circuit closures.*

Proof. Given a list of all the circuits of a matroid, the circuit closure for any circuit C can be constructed in polynomial time. Simply check for each circuit $C' \neq C$, if $|C \cap C'| = |C'| - 1$, then add the single element in $C' - C$ to $cl(C)$. Once this is done, remove C' from the list and repeat. For each circuit, this will take a polynomial number of checks. $\square$

Lemma 5.2.2. *Suppose Z is a cyclic flat of a matroid M . If T is a separator of $M|Z$, then T is a cyclic flat.*

Proof. We can write Z as $Z = T \cup V$. Because Z is a union of circuits, T and V must also be unions of circuits. Consider the closure of T and V . It is clear that, $cl(T) \subseteq cl(Z) = Z$ and $cl(V) \subseteq Z$. Suppose that we have an element $v \in V$ such that $v \in cl(T)$. Then $r(T) = r(T \cup v)$ and therefore there exists a circuit C containing v such that $C \subseteq T \cup v$. This is a contradiction as T would no longer be a separator. Therefore $cl(T) = T$ and it follows that T is a flat. Moreover, as T is made from a union of circuits, T is a cyclic flat. $\square$

Lemma 5.2.3. *Cyclic flats $\geq$ connected cyclic flats.*

Proof. A cyclic flat is connected if and only if it does not have a non-trivial separator. From Lemma 5.2.2, any separator of a cyclic flat is a cyclic flat. Given a list of cyclic flats and their ranks, it can be easily checked, by construction of a lattice under inclusion, whether any of them are separators for another cyclic flat in the list. $\square$

Lemma 5.2.4. *Connected cyclic flats/circuit closures $\not\geq$ cyclic flats.*

Proof. Consider the matroid $M = U_{2,3} \oplus \cdots \oplus U_{2,3}$ with $|E(M)| = 3n$. This will have n connected cyclic flats/circuit closures (consisting of one of the copies of $U_{2,3}$). However, it will have 2^n cyclic flats $\square$

Lemma 5.2.5. *Connected cyclic flats/circuit closures $\not\geq$ nonspanning circuits.*

Proof. Consider the matroid M obtained by taking $U_{n-1,n} \oplus U_{n-1,n}$ and replacing every element with n parallel elements. M will have a connected cyclic flat/circuit closure for each parallel class plus one for all of the n element circuits of each of the two components. This gives a total of $2n+2$ connected cyclic flats/circuit closures. This is polynomial in n . However, if we want to construct an n element circuit, we have n choices from each of the n parallel classes. This gives n^n such circuits. We will also have $n\binom{n}{2}$ two element circuits from each component. Because M has two components and no circuit meets both components, none of M 's circuits will be spanning. This gives $2(n^n + n\binom{n}{2})$ nonspanning circuits, which is not polynomial in n . $\square$

Lemma 5.2.6. *Circuits $\not\geq$ connected cyclic flats.*

Proof. Consider the matroid M with $E(M) = \{1, 2, \dots, 2n+1\}$ for some $n \in \mathbb{Z}$ with circuits $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$ where

$$\mathcal{C}_1 = \{\{1, 2, 3\}, \{1, 4, 5\}, \{1, 6, 7\}, \dots, \{1, 2n, 2n+1\}\}$$

and

$$\mathcal{C}_2 = \{\{a, b, c, d\} \mid \{1, a, b\}, \{1, c, d\} \in \mathcal{C}_1\}.$$

There will be n circuits in $\mathcal{C}_1$ and no more than n^2 circuits in $\mathcal{C}_2$ giving no more than $n(n+1)$ circuits in M . However, any combination of circuits from $\mathcal{C}_1$ will give a connected cyclic flat. This cyclic flat will only consist of the circuits that were combined and so will be unique. There will be 2^n such combination of circuits from $\mathcal{C}_1$. Thus, M has at least 2^n connected cyclic flats which is not polynomial in terms of $n(n+1)$. $\square$

Lemma 5.2.7. *Hyperplanes $\not\geq$ connected cyclic flats/circuit closures.*

Proof. Consider the following graphic matroid. Take a cycle of n edges and add an edge in parallel to every edge. Now subdivide every edge. Let M be the

corresponding graphic matroid. The dual of this matroid will be the graphic matroid obtained from taking the complete bipartite graph $K_{2,n}$ and adding an edge in parallel with every edge. Consider a circuit of M^* . There will be $2n$ parallel pairs giving $2n$ circuits of two elements. All other circuits must have size 4. There will be $4n(n-1)$ such circuits. This gives a total of $4n^2 - 2n$ circuits in M^* . Therefore M has $4n^2 - 2n$ hyperplanes. However by adding parallel edges and then subdividing the edges to create M , we turned the cycle of length n into 2^n cycles of length $2n$. Each of the corresponding circuits is also a flat. Therefore M has at least 2^n circuit closures and thus at least 2^n connected cyclic flats. This is not polynomial in terms of $4n^2$. $\square$

Lemma 5.2.8. *Nonspanning circuits $\geq$ circuit closures.*

Proof. Given a list of the nonspanning circuits of a matroid, we can easily construct the set $\mathcal{C}_{ns}$ of closures of the nonspanning circuits in polynomial time. Now by Proposition 6.1.7 [54], we can check to see if M has a spanning circuit in polynomial time when given the list of nonspanning circuits. If there exists a spanning circuit then the circuits closures of M are $\mathcal{C}_{ns} \cup \{E\}$. Otherwise they are just $\mathcal{C}_{ns}$. $\square$

Note that each circuit closure is also a connected cyclic flat. It seems likely that given the connected cyclic flats, one could create a list of the circuit closures in polynomial time. This problem is equivalent to deciding if a matroid has a spanning circuit when given the connected cyclic flats. This is because if we want to decide if a connected cyclic flat is a circuit closure, we can simply restrict the ground set to the connected cyclic flat. This new matroid will have a spanning circuit if and only if the connected cyclic flat is a circuit closure. However, finding a method of deciding if a matroid has a spanning circuit given the list of connected cyclic flats has proven troublesome.

Conjecture 5.2.9. *Connected cyclic flats $\geq$ circuit closures.*

Combining these lemmas, we get the hierarchy shown in figure 5.2 where the dotted line exists if conjecture 5.2.9 is true.

5.3 Connected Cyclic Flats Axioms

In this section we will provide axioms to completely describe a matroid via its connected cyclic flats and show that these axioms do indeed capture the notion of

Figure 5.2: The ordering of the 12 inputs.

connected cyclic flats. We note that axioms for cyclic flats have been constructed [14]. Recall that we define a cyclic flat to be connected if, minus loops, it is connected. This allows us to describe all matroids in this way.

Let E be a set, $\mathcal{Z}$ be a collection of subsets of E and r be a function that maps $\mathcal{Z}$ to the set of non-negative integers. We claim that the collection $\mathcal{Z}$ is the set of connected cyclic flats of a matroid and r is their rank function if the following five axioms are satisfied for all $Z_1, Z_2 \in \mathcal{Z}$.

1. There exists exactly one $Z_0 \in \mathcal{Z}$ with $r(Z_0) = 0$. Furthermore, $Z_0 \subset Z$, for all $Z \in \mathcal{Z}, Z \neq Z_0$.
2. Suppose $Y = Z_1 \cap Z_2 - Z_0 \neq \emptyset$. Let X be a set $X \subseteq Y$ such that $X \cap Z' < r(Z')$ for all $Z' \in \mathcal{Z}$. Then there exists a unique minimal $Z_3 \in \mathcal{Z}$ with the property that $Z_1 \cup Z_2 \subseteq Z_3$ and $r(Z_1) + r(Z_2) \geq r(Z_3) + |X|$.
3. If $Z_3 = Z_1 \cup Z_2$ and $Z_1 \cap Z_2 \subseteq Z_0$ then $r(Z_1) + r(Z_2) > r(Z_3)$.
4. If $Z_1 \subset Z_2$ then $0 < r(Z_2) - r(Z_1) < |Z_2 - Z_1|$.

5. If $e \notin Z$ and there exists $I \subseteq Z$ such that $|I \cap Z'| \leq r(Z')$ for all $Z' \in \mathcal{Z}$ then $|(I \cup e) \cap Z'| \leq r(Z')$ for all $Z' \in \mathcal{Z}$.

We briefly note what some of these axioms are implying. The first axiom shows that there is exactly one rank 0 connected cyclic flat and this is contained in all other connected cyclic flats. The third shows that if the union of two connected cyclic flats is in the set, then it must also be connected. While the fourth shows that if $Z_1 \subset Z_2$, then $Z_2 - Z_1$ cannot contain coloops.

We will begin by showing that the set of connected cyclic flats of a matroid obey these axioms. We will then move on to showing that any collection of subsets that obey these axioms is the collection of connected cyclic flats for some matroid.

Lemma 5.3.1. *The set of connected cyclic flats $\mathcal{Z}$ of a matroid M obey 1, ..., 5.*

Proof. (1) If $\mathcal{Z}$ is the set of connected cyclic flats of M then there is exactly one set Z_0 with $r(Z_0) = 0$. This set is the set of all M 's loops.

(2) Take two connected cyclic flats whose overlap contains non-loop elements and consider their closure. This will be a flat. Note that the property of being in a common circuit is an equivalence relation ([58] Prop 4.1.2). As every element of Z_1 is in a circuit with every other element in Z_1 and the same for Z_2 , every element in $Z_1 \cup Z_2$ will be in a circuit with every element in $Z_1 \cap Z_2$. Therefore every element in $Z_1 \cup Z_2$ will be in a circuit with every element of $Z_1 \cup Z_2$. Now consider any non-loop elements e such that $e \in cl(Z_1 \cup Z_2)$, $e \notin Z_1 \cup Z_2$. Because e is in the closure, there must exist a circuit $C \in (Z_1 \cup Z_2) \cup e$ that contains e . This circuit must contain elements from both $Z_1 - Z_2$ and $Z_2 - Z_1$. Therefore e is in a circuit with every element in $Z_1 \cup Z_2$ and it follows that $cl(Z_1 \cup Z_2) = Z_3$ is a connected cyclic flat.

Now let X be a basis for $Z_1 \cap Z_2$ and B_i be a basis of Z_i that contains X for $i = 1, 2, 3$. Also, let B_{12} be a basis for $Z_1 \cup Z_2$ containing X . We know that $|B_{12}| \leq |B_1 \cup B_2|$ and so $|B_3| \leq |B_1 \cup B_2|$. Now we have

$$\begin{aligned}
 r(Z_1) + r(Z_2) - |X| &= |B_1| + |B_2| - |X| \\
 &= |B_1 \cup B_2| + |B_1 \cap B_2| - |X| \\
 &\geq |B_3| \\
 &= r(Z_3).
 \end{aligned}$$

Where X is the largest possible independent set in $Z_1 \cap Z_2$. Therefore

$$r(Z_1) + r(Z_2) \geq r(Z_3) + |X|$$

will hold for all independent sets in $X \in (Z_1 \cap Z_2)$. Now suppose we have two minimal connected cyclic flats Z_3, Z_4 such $Z_1 \cup Z_2 \subseteq Z_3, Z_4$. Then, by the above argument $r(Z_3) = r(Z_4) = r(Z_1 \cup Z_2)$. If Z_3 and Z_4 are flats, then $Z_4 = cl(Z_1 \cup Z_2)$ and $Z_3 = cl(Z_1 \cup Z_2)$ and it follows that $Z_3 = Z_4$.

(3). If $Z_3 = Z_1 \cup Z_2$, $Z_1 \cap Z_2 \subseteq Z_0$ and $r(Z_1) + r(Z_2) = r(Z_3)$ then Z_1 is a separator of Z_3 and Z_3 cannot be connected. We know that $r(Z_1) + r(Z_2) \geq r(Z_3)$ and therefore, if $Z_3 = Z_1 \cup Z_2$ and $Z_1 \cap Z_2 \subseteq Z_0$, then $r(Z_1) + r(Z_2) > r(Z_3)$.

(4) Suppose we have $Z_1 \subset Z_2$. We cannot have $r(Z_2) - r(Z_1) > |Z_2 - Z_1|$ because this would give $|Z_2 - Z_1| < r(Z_2 - Z_1)$. So assume that $r(Z_2) - r(Z_1) = |Z_2 - Z_1|$. Then

$$r(Z_1) + r(Z_2 - Z_1) = r(Z_2)$$

and Z_1 is a separator of Z_2 . This gives us a contradiction which shows that $r(Z_2) - r(Z_1) < |Z_2 - Z_1|$.

(5) A set I is independent if and only if $|I \cap Z| \leq r(Z)$ for all connected cyclic flats. Take any independent set $I \subset Z$. If $e \notin Z$ then $I \cup e$ must be independent showing that $|(I \cup e) \cap Z'| \leq r(Z')$ for all $Z' \in \mathcal{Z}$. $\square$

We will define the set $\mathcal{I}$ to consist of all $I \subseteq E$ such that $|I \cap Z| \leq r(Z)$ for all $Z \in \mathcal{Z}$. Moreover, let $\mathcal{C}$ be the set of the minimal subsets of E that are not in $\mathcal{I}$.

Lemma 5.3.2. *There exists a matroid $M_{\mathcal{C}}$ with circuits $\mathcal{C}$.*

Proof. The first two circuit axioms follow from the definition of $\mathcal{C}$. Assume that we have two distinct sets $C_1, C_2 \in \mathcal{C}$ such that $e \in C_1 \cap C_2$ and $e \notin Z_0$. Let Z_1 be minimal with the property that $C_1 \subseteq Z_1$ and Z_2 be minimal with the property that $C_2 \subseteq Z_2$. Suppose that $Z_1 = Z_2$. By our definition of $\mathcal{C}$, there exists v such that $v \in C_2 - C_1$ and $|C_1| = |C_2| = r(Z_1) + 1$. Then

$$|C_1 \cup C_2 - e| \geq |C_1 - e \cup v| = |C_1| > r(Z_1).$$

But then $(C_1 \cup C_2 - e) \subseteq Z_1$ where $|C_1 \cup C_2 - e| > r(Z_1)$ and therefore contains a circuit.

Now we will assume $Z_1 \neq Z_2$. By the definition of $\mathcal{C}$, and 2 we know there exists

a Z_3 such that $(Z_1 \cup Z_2) \subseteq Z_3$ with

$$r(Z_1) + r(Z_2) \geq r(Z_3) + |C_1 \cap C_2|.$$

Now consider the set $C_1 \cup C_2 - e$. We see that

$$\begin{aligned} |C_1 \cup C_2 - e| &= |C_1 - e| + |C_2 - e| - |(C_1 \cap C_2) - e| \\ &= r(Z_1) + r(Z_2) - |(C_1 \cap C_2) - e| \\ &> r(Z_3) \text{ (by the equation above)} \end{aligned}$$

where $(Z_1 \cup Z_2) \subseteq Z_3$. Therefore Z_3 contains a circuit and the circuit elimination axiom is satisfied. □

It follows from our definition of $\mathcal{I}$ and $\mathcal{C}$ that the independent sets of $M_{\mathcal{C}}$ are the sets in $\mathcal{I}$.

Lemma 5.3.3. *The ranks of the sets in $\mathcal{Z}$ are given by $r(Z)$.*

Proof. By the way we have defined our circuits, $\text{rank}(Z) \leq r(Z)$. Now suppose we have a connected cyclic flat Z_1 where for any maximum sized independent set $I \subseteq Z_1$ we have $|I| < r(Z_1)$. Then for each $e \in (Z_1 - I)$, there must exist a Z_2 such that $|I \cap Z_2| \leq r(Z_2)$ but $|(I \cup e) \cap Z_2| > r(Z_2)$. From 1 and 4 we know that $r(Z_1) < |Z_1| - |Z_0|$ so there is always an $e \in (Z_1 - I), \notin Z_0$. If there exists an $e \in Z_1, \notin Z_2$ then by 5 we have $|(I \cup e) \cap Z_2| \leq r(Z_2)$. Thus, $Z_1 \subset Z_2$ and it follows from 4 that $r(Z_1) < r(Z_2)$. Because $|I \cap Z_2| \leq r(Z_2)$ and there exists an e such that $|(I \cup e) \cap Z_2| > r(Z_2)$ we must have $r(Z_2) = |I|$. However, $I \subseteq Z_1$ and $|I \cap Z_1| = |I| < r(Z_1) < r(Z_2) = |I|$ this is a contradiction and so the ranks of the sets in $\mathcal{Z}$ are given by $r(Z)$. □

Lemma 5.3.4. *The sets in $\mathcal{Z}$ are connected cyclic flats.*

Proof. Let Z be a connected cyclic flat, I be the largest independent set in Z and suppose we have $x \notin Z$. Then by 5, $I \cup x$ is independent and therefore $x \notin \text{cl}(Z)$. This shows that $\text{cl}(Z) = Z$. It follows from 3 that the sets in $\mathcal{Z}$ are connected. Because the sets in $\mathcal{Z}$ are connected, they are cyclic. Therefore, the sets in $\mathcal{Z}$ are connected cyclic flats. □

Lemma 5.3.5. *All the connected cyclic flats of M are in $\mathcal{Z}$.*

Proof. By definition, the unique connected cyclic flat Z_0 with $r(Z_0) = 0$ is in $\mathcal{Z}$ plus all the circuit closures of M . Now let Z^* be the smallest connected cyclic flat of M that it has not been shown that $Z^* \in \mathcal{Z}$. We know Z^* cannot be a circuit closure. Thus, we can find at least two circuits in Z^* that do not span Z^* . Let Z_1 and Z_2 be two such circuit closures. Because Z^* is connected, if $Z_1 \cap Z_2 = Z_0$ then there exists a circuit that overlaps Z_1 and Z_2 . Because of this, we know that $cl(Z_1 \cup Z_2)$ is a connected cyclic flat. Call this $Z_{1,2}$. Because $(Z_1 \cup Z_2) \subseteq Z^*$, we have

$$Z_{1,2} = cl(Z_1 \cup Z_2) \subseteq cl(Z^*) = Z^*.$$

Because Z^* is the minimal connected cyclic flat that we do not know is in $\mathcal{Z}$, we know that all connected cyclic flats contained in Z^* are in $\mathcal{Z}$. So we can successively take the closure of $Z_{1,2} \cup Z_3 = Z_{1,2,3}$ and then $Z_{1,2,3} \cup Z_4 = Z_{1,2,3,4}$ and so on. Eventually we will take the closure of $Z_{1,\dots,j} \cup Z_{j+1}$ to obtain $Z_{1,\dots,j+1}$ such that $Z_{1,\dots,j+1} \in \mathcal{Z}$ and $Z^* \subseteq Z_{1,\dots,j+1}$. By 2, $Z_{1,\dots,j+1}$ is unique and $r(Z_{1,\dots,j}) + r(Z_{j+1}) - |X| \geq r(Z_{1,\dots,j+1})$ where X is a basis for $Z_{1,\dots,j} \cap Z_{j+1}$. But $r(Z^*) \geq r(Z_{1,\dots,j} \cup Z_{j+1})$ and therefore as $Z^* \subseteq Z_{1,\dots,j+1}$ and $r(Z^*) \geq r(Z_{1,\dots,j+1})$ we have $Z^* = Z_{1,\dots,j+1}$ and $Z^* \in \mathcal{Z}$. $\square$

Chapter 6

Input Hierarchy

6.1 Introduction

A well known property of matroids is that they have many different methods of description. Furthermore these descriptions are often not polynomially equivalent. That is, given one description we cannot create another description for the same matroid in polynomial time (in terms of the size of the first description). For example, given the list of independent sets of a matroid, we can produce the list of circuits in polynomial time. However, given the list of circuits, we cannot list the independent sets in polynomial time. Due to this, the complexity of solving matroid problems can vary greatly depending on the method of description used. A problem may be NP-complete when given one description, but in P for another. For example 3-matroid intersection can be solved in polynomial time given the list of independent sets, but is NP-complete when given the list of circuits [55]. Suppose that given description A , we can create description B in polynomial time. Then if a problem is NP-complete for description A , it must be NP-hard for description B . The way these relations affect the complexity of problems is the motivation for studying the hierarchy of matroid descriptions.

In [55] a hierarchy was created for ten common matroid descriptions. We added two more to this in Chapter 5. However, this is only a small part of the hierarchy of matroid descriptions. We are interested in the structure in which the matroid descriptions are embedded. As matroids are countable, their descriptions can be thought of as injective functions that take natural numbers to binary strings. A function f is *computable* if there exists a Turing machine that can calculate $f(n)$ in a finite amount of time. We can assume that we have a computable injective func-

tion that takes matroids to the natural numbers. We can construct such a function by doing the following. First order all matroids by the size of their ground set so that $M_1 < M_2$ if $|E(M_1)| < |E(M_2)|$. Now for each matroid, label its ground set so that when we list the independent sets lexicographically, we get the minimal possible list. Now order all matroids with the same sized ground set by their list of independent sets. We will impose the additional requirement that if f is a matroid description, then given $f(n)$ for some injective function, we can simulate an independence oracle in polynomial time for the matroid n for all $n \in \mathbb{N}$. So we can assume that our matroid description is an injective function that takes the natural numbers to binary strings such that given $f(n)$ we can simulate an independence oracle for the matroid n in polynomial time. This is how we will think about matroid descriptions in this chapter. We wish to know how the hierarchy of such functions behaves. However, when examining this hierarchy, there is no reason to restrict to just matroid descriptions. We will consider a more general hierarchy of all injective functions that take natural numbers to binary strings. This hierarchy will contain the hierarchy of matroid descriptions. Studying this hierarchy will not only give us insight into how the matroid description hierarchy behaves, but how many other description hierarchies behave. In light of this, for the time being, we will forget about matroid descriptions and focus on the structure of the hierarchy of injective functions. Once we know how this hierarchy behaves, we will use this information to deduce how the matroid description hierarchy behaves. So for the majority of this chapter, the hierarchy we are interested in will consist of injective functions that take natural numbers to binary strings. This hierarchy has similarities with both the Turing degrees and the hierarchy of many-to-one functions. We will compare our hierarchy with these two hierarchies.

We will call a function f *succinct* if there exists a constant c_f such that $|f(n)| \leq c_f^n$ for all n . Let $U(n)$ be the unary representation of the natural number n . Also, let Σ^* be the collection of all finite binary strings. We will define $\mathcal{N}$ to be the set of all succinct injective functions $f : \mathbb{N} \rightarrow \Sigma^*$ such that there exists a finite Turing machine that on input $U(n)$ will output $f(n)$ for all $n \in \mathbb{N}$. Note that this Turing machine need not run in polynomial time. For $A, B \in \mathcal{N}$ we say that $A \leq B$ if there exists a polynomial time Turing machine that on input $B(n)$ will output $A(n)$ for all $n \in \mathbb{N}$. We will not differentiate between the set $\mathcal{N}$ and the hierarchy defined by the set $\mathcal{N}$ and the relation $\leq$. We enforce succinctness as a way of preventing the creation of infinite ascending chains by simply padding the

functions. For example, take a function f and create an infinite ascending chain $f_0 = f < f_1 < f_2, \dots$ above f where $f_i(n)$ is made by repeating $f_{i-1}(n)$ $2^{|f_{i-1}(n)|}$ times. If for two natural numbers $n_1 \neq n_2$ we have $f(n_1) = f(n_2)$ then we have a *collision*. This cannot happen if f is injective.

We will show that $\mathcal{N}$ has the following properties:

Theorem 6.1.1. $\mathcal{N}$ is a semi-lattice.

Theorem 6.1.2. $\mathcal{N}$ is dense.

Theorem 6.1.3. Any countable distributive lattice can be embedded in $\mathcal{N}$.

Note that Theorem 6.1.3 implies the existence of infinitely ascending and descending chains and infinite anti-chains.

This hierarchy is similar to the Turing degrees and many-to-one functions. We will briefly describe these and their similarities below. Let A and B be subsets of Σ^* . We will not differentiate between a subset A and its characteristic function. So $x \in A$ if and only if $A(x) = 1$. A subset A is *Turing reducible to B* (denoted $A \leq_T B$) if there is an oracle Turing machine that when given an oracle for membership of B , can decide membership of A . Note that in the case of the Turing degrees we do not require that the Turing machine runs in polynomial time. The Turing degrees are equivalence classes of sets. Two subsets A and B are in the same Turing degree if $A \leq_T B$ and $B \leq_T A$. A subset A is *many-to-one reducible to B* (denoted $A \leq_m^p B$) if for all x , $A(x) = B(f(x))$ for some polynomial time function f .

We note that both the Turing degrees and hierarchy of many-to-one functions have received a great deal of study. By comparison to what we show in this chapter, for the Turing degrees it is known that

1. The Turing degrees are a semi-lattice [47, 82].
2. The Turing degrees are not dense [68].
3. Any countable distributive lattice can be embedded in the Turing degrees [69].

Furthermore, for the hierarchy of many-to-one reductions we have.

1. The hierarchy of many-to-one reductions is a semi-lattice [48].

2. The hierarchy of many-to-one reductions is dense [48].
3. Any countable distributive lattice can be embedded in the hierarchy of many-to-one reductions [4].

Given the above results, one would be forgiven for the assumption that $\mathcal{N}$ is the same as the many-to-one hierarchy. However, there is an important difference. Let A and B be two many-to-one functions such that $A \leq_p^m B$. Then there exists a polynomial time function f such that $A(x) = B(f(x))$ where $A(x)$ and $B(x)$ are the respective characteristic functions. That is $x \in A$ if and only if $f(x) \in B$. However, the function f could take all $x \in A$ to a single $y \in B$. That is, $f(x) = y$ for all $x \in A$. In our hierarchy, this is not allowed as this would create a collision. The function f must take each $x \in A$ to a distinct $y \in B$. Moreover, our functions come with an ordering on the elements in the sets. So if we have $f : A \rightarrow B$ we must have $f(A(a_1)) = B(b_1)$, $f(A(a_2)) = B(b_2)$, etc where $A = (a_1, a_2, \dots)$ and $B = (b_1, b_2, \dots)$. Note that because of this, we can have sets A and B such that $A \leq_m^p B$ but $A \not\leq B$. For example, let $A = \{2n | n \in \mathbb{N}\}$ and let B be any computable but non-polynomial set. Then let

$$f(x) = \begin{cases} b_i & \text{if } x = 2n, n \in \mathbb{N} \\ c_i & \text{otherwise,} \end{cases}$$

where b_i is the smallest number in B and c_i is the smallest number not in B . As B is computable, we can find b_i and c_i . Thus we see that $A(x) = B(f(x))$. However, there is no polynomial function such that $f(B(b_i)) = f(a_i)$ for all i .

6.2 The structure of $\mathcal{N}$

Here we will provide preliminary results that will combine into proofs for Theorems 6.1.1, 6.1.2 and 6.1.3. A number of the results that follow will require a computable but non-polynomial set of natural numbers. We will construct such a set using Lemmas 6.2.1 and 6.2.2.

Lemma 6.2.1. *Let X be a finite set of natural numbers. Then given a number $U(n)$, there exists a Turing machine that can decide if $n \in X$ in no more than $n + 2$ steps.*

Proof. Consider the following Turing machine T . Let x be the largest natural number in X . Then for $i \in \{1, \dots, x\}$ T will have x states T_i and x halting states S_i , plus a single starting state S . On any input, state S_i will halt and output 1 if $i \in X$ and halt and output 0 otherwise. The Turing machine T will begin in state S and read the first digit of n . If it is a 1, it will go to state T_1 and shift once to the right. Otherwise it will output 0 and halt. The states T_j for $j \in \{1, \dots, x-1\}$ will go to state T_{j+1} and shift once to the right on reading a 1 and will go to state S_j if it reads a 0. The state T_x will go to state S_x if it reads a 0 and will output 0 and halt if it reads a 1 (as this would mean that $n > x$ and therefore not in X).

Now suppose we are given $U(n)$ for some natural number n . Then the Turing machine will read a 1 and then go to state T_1 . It will then go to state T_2 and then T_3 and so on, as long as it keeps reading 1's. If $n \leq x$ then when the Turing machine reaches state T_n it will read a 0 and then go to state S_n . This will then output whether or not $n \in X$. Alternatively, if $n > x$ then from state S , the Turing machine will go through each state T_i until it reaches state T_x . It will then read a 1 as $n > x$ and will therefore output a 0 and halt.

If $n \leq x$ then this process will take $n + 1$ steps to get to state S_n and then one step to output 0 or 1. If $n > x$ then this will take $x + 2 < n + 2$ steps to output 0. Thus this process will take at most $n + 2$ steps. $\square$

As there is a countable number of Turing machines, there is a correspondence between the natural numbers and Turing machines. Due to this, we can just refer to a Turing machine by the natural number associated with it. A well known non-computable problem is that of deciding if a Turing machine n will halt. If instead, we want to know if the Turing machine n halts in 2^n steps, this problem becomes computable, but non-polynomial. This can be proven using a modification of the proof that the halting problem is non-computable.

Lemma 6.2.2. *There is no polynomial Turing machine that given input $U(n)$ and $U(m)$, will decide if the Turing machine n halts in 2^n steps on input m .*

Proof. Suppose there is such a polynomial Turing machine H . That is, the following Turing machine defined by

$$H(n, m) = \begin{cases} 1 & \text{if } n \text{ halts in no more than } 2^n \text{ steps on input } m, \\ 0 & \text{otherwise.} \end{cases}$$

We will choose the Turing machine 1 to be the Turing machine that immediately halts on any input. Thus the difficulty is in deciding if the Turing machines $n \geq 2$ halt on input m . For each binary polynomial program $f(i, j)$ create the following program

$$g_f(i) = \begin{cases} 0 & \text{if } f(i, i) = 0, \\ \text{loop forever} & \text{otherwise.} \end{cases}$$

Note that even though H runs in polynomial time, it can still take 2^n or more steps for a finite number of cases. Let X be the finite set where $x \in X$ if $H(x, x)$ runs in time greater than $2^x - (x + 2)$. Now consider the Turing machine H' defined by

$$H'(i, j) = \begin{cases} b & \text{if } i = j \in X, \\ H(i, j) & \text{otherwise,} \end{cases}$$

where b is the value of $H(i, j)$. Note that H and H' are genuinely different. If $i = j \in X$, then H' has a lookup table and simply outputs 0 or 1 while H will run for a number of steps (greater than $2^i - (i + 2)$) and then output 0 or 1. From Lemma 6.2.1, we know that we can decide if $i \in X$ in $i + 2$ steps. Thus for all $i \geq 2$, $g_{H'}(i)$ either loops forever or halts in no more than 2^i steps as it takes no more than $i + 2 \leq 2^i$ steps to test if $i \in X$ and then no more than $2^i - (i + 2)$ steps to run $H(i, i)$ if $i \notin X$. Note that H' exists if and only if H exists. Now let n' be the natural number associated with $g_{H'}$. Suppose that $H'(n', n') = 1 = H(n', n')$. Then $g_{H'}(n')$ will loop forever and thus $H(n', n') = 0$ and we have a contradiction. Now suppose $H'(n', n') = 0 = H(n', n')$. Then $g_{H'}(n') = 0$. Furthermore, we know that $g_{H'}(n')$ halts in no more than $2^{n'}$ steps and thus $H(n', n') = 1$ which is again a contradiction. Thus H cannot exist and therefore there is no polynomial time Turing machine that can decide if a Turing machine n halts in 2^n steps on input m . $\square$

Now let $H \subset \mathbb{N}$ be the set of natural numbers such that $h \in H$ if and only if the Turing machine h halts in 2^h steps on input h . From Lemma 6.2.2 we know that membership of this set cannot be decided in polynomial time. Thus H is a computable but non-polynomial set.

We will begin our study of $\mathcal{N}$ with the case of joins and meets. We will show that while every pair of functions do indeed have a join, not every pair of functions have a meet. This will show that $\mathcal{N}$ is a semi-lattice.

We will first show that not every pair of functions in $\mathcal{N}$ have a meet. This will

be done in Lemma 6.2.4. To do so we will want to permute binary strings. However, as permutations are for a fixed length, given a permutation p , we cannot always permute a given binary representation of a natural number. So for the binary representation $B(n)$ of n and permutation p of length k , we will define $p(B(n))$ to be the following. Take $B(n)$ and remove the last digit which corresponds to the highest power of 2 (which will always be a 1 in $B(n)$). Then divide this into blocks of length k and apply p to each full block of length k . Then finally add the last digit back on. The reason we remove the last digit before the permutation and then add it back on is we want to make sure that $|p(B(n))| = |B(n)|$. Otherwise we could have $p(B(2)) = p(01) = 10 = 1 = B(1)$ where we see that the length of the string has changed. Note that removing the last bit before the permutation means that for $n = 2^j$, $p(B(n)) = B(n)$. Let p be some bit permutation of length k such that $p \neq p^{-1}$ and p can be expressed as a single cycle. Consider the following two functions f_1 and f_2 .

$$f_1(n) = \begin{cases} p(B(n)) & \text{if } |B(n)| \in H \\ B(n) & \text{otherwise.} \end{cases}$$

$$f_2(n) = \begin{cases} p(B(n)) & \text{if } |B(n)| \notin H \\ B(n) & \text{otherwise.} \end{cases}$$

These two functions are both succinct and injective and therefore in $\mathcal{N}$.

Lemma 6.2.3. *The functions f_1 and f_2 are incomparable in $\mathcal{N}$.*

Proof. Suppose that given $f_1(n)$, we have an algorithm $\mathcal{A}$ that can create $f_2(n)$ in polynomial time. Recall that $p \neq p^{-1}$ and p can be expressed as a single cycle. Therefore if $p(f_1(n)) = f_2(n)$ and $n \neq 2^j$, then $|B(n)| \notin H$. Alternatively, if $p^{-1}(f_1(n)) = f_2(n)$, then $|B(n)| \in H$. Moreover, if $f_1(n) = p(B(n))$ then $f_1(n') = p(B(n'))$ where n' is the natural number with binary representation $p(B(n))$. Therefore if given some $m \in \mathbb{N}$, we can create any $B(n)$ where $n \neq 2^j$ and $|B(n)| = m$ and then test if $\mathcal{A}(B(n)) = p(B(n))$ or $\mathcal{A}(B(n)) = p^{-1}(B(n))$. If $\mathcal{A}(B(n)) = p(B(n))$ then $m \notin H$. Otherwise if $\mathcal{A}(B(n)) = p^{-1}(B(n))$ then $m \in H$. So the algorithm $\mathcal{A}$ would give a method of deciding if $m \in H$. Therefore $\mathcal{A}$ cannot exist. A similar argument shows that given $f_2(n)$, there is no polynomial algorithm that can produce $f_1(n)$. Thus these two functions are incomparable in $\mathcal{N}$. $\square$

Lemma 6.2.4. *There exists functions in $\mathcal{N}$ that do not have a meet.*

Proof. This will be proven by showing that the two functions f_1 and f_2 do not have a meet. So suppose these two functions do have a meet M . Then because they are incomparable, there exists two distinct polynomial time functions F_1 and F_2 such that $F_1(f_1(n)) = M(n) = F_2(f_2(n))$. We will show that existence of these functions would allow us to decide if $m = |B(n)| \in H$ for any $m \in \mathbb{N}$ and therefore cannot exist. Note that if $|B(n)| \in H$, then

$$F_1(p(B(n))) = F_1(f_1(B(n))) = M(n) = F_2(f_2(B(n))) = F_2(B(n)).$$

Furthermore, if $|B(n)| \notin H$, then

$$F_1(B(n)) = F_1(f_1(B(n))) = M(n) = F_2(f_2(B(n))) = F_2(p(B(n))).$$

Recall that p is a permutation of length k . Suppose we have some $|B(n)| \in H$ where $|B(n)| > k$ and $n = 1 \pmod 2$. If we know that this means that we cannot have $F_1(B(n)) = F_2(p(B(n)))$ then we have a test to decide if $|B(n)| \in H$. That is, test if $F_1(B(n)) = F_2(p(B(n)))$. If so then $|B(n)| \notin H$. Otherwise $|B(n)| \in H$. So assume that if $|B(n)| \in H$ (and thus $F_1(p(B(n))) = F_2(B(n))$), we can also have $F_1(B(n)) = F_2(p(B(n)))$. Note that if $|B(n)| \in H$ then $|p(B(n))| \in H$. Then because of this, we know that

$$\begin{aligned} F_1(B(n)) = F_1(p(p^{-1}(B(n)))) &= F_1(f_1(p^{-1}(B(n)))) \\ &= F_2(f_2(p^{-1}(B(n)))) = F_2(p^{-1}(B(n))). \end{aligned}$$

But by our assumption, $F_1(B(n)) = F_2(p(B(n)))$ and we see that $F_2(p(B(n))) = F_2(p^{-1}(B(n)))$. But $p \neq p^{-1}$ and p is a cycle. Hence, for $n = 1 \pmod 2$, $p(B(n)) \neq p^{-1}(B(n))$ and therefore we have $M(n_1) = M(n_2)$ for natural numbers $n_1 \neq n_2$. This is a collision. Therefore M is not in $\mathcal{N}$ and thus cannot be the meet of f_1 and f_2 . Thus if $|B(n)| \in H$, $n \neq 2^j$ and $|B(n)| > k$, then we cannot have $F_1(B(n)) = F_2(p(B(n)))$. Therefore for $|B(n)| > k$ and $n \neq 2^j$, $|B(n)| \notin H$ if and only if $F_1(B(n)) = F_2(p(B(n)))$. Similarly $|B(n)| \in H$ if and only if $F_1(p(B(n))) = F_2(B(n))$. For any given $m \in \mathbb{N}$, we can construct some $B(n)$ such that $|B(n)| = m$ and $n = 1 \pmod 2$ in polynomial time. If $m > k$ we could then use F_1 and F_2 to determine if $m \in H$. If $m \leq k$, then we only have a finite number of cases to check to determine if $m \in H$. This shows that the existence of the meet

M would give a way of deciding if $m = |B(n)| \in H$ in polynomial time. Therefore, M cannot exist. $\square$

So we see that $\mathcal{N}$ has no meet. What about a join? We will use the following operation to create a join for every pair of functions. Suppose we are given a function f . An *embiggening* of f , denoted $E(f)$ is the following function. Take $f(n)$ and replace each 0 with 00 and each 1 with 11. This creates the function $E(f)$. We will use embiggening as a way of concatenating two strings together in such a way that we can still recover the original strings.

Lemma 6.2.5. *Any two functions in $\mathcal{N}$ have a join.*

Proof. Take any two functions f and g in $\mathcal{N}$. Define the new function fg to be the following. Let $fg(n) = E(f(n))010g(n)$. By the construction $fg \geq f$ and $fg \geq g$. This function fg is succinct as we can just choose $c_{fg} = 3 \max\{c_f, c_g\} + 3$. Furthermore, fg is injective as both f and g are. This shows that $fg \in \mathcal{N}$. We claim that this is the unique minimum function that is above f and g . To see this, suppose we have a function $h \in \mathcal{N}$ such that $h \geq f$ and $h \geq g$. Then given $h(n)$ we can construct $f(n)$ and $g(n)$ in polynomial time. But then we could construct $fg(n)$ and thus $h \geq fg$. Thus every pair of functions in $\mathcal{N}$ have a unique join. $\square$

It follows from Lemmas 6.2.4 and 6.2.5 that $\mathcal{N}$ is a semi-lattice but not a lattice. This proves Theorem 6.1.1. We now move on to showing that $\mathcal{N}$ is dense. The proof for this will use ideas very similar to the proof by Ladner that the hierarchy of many-to-one reductions is dense [48]. To show that $\mathcal{N}$ is dense, we will show that if $B \not\leq A$, then there exists a polynomial time recognizable set D such that:

1. There exist no polynomial time Turing machine that when given $A(n)$, will output $B(n)$ for all $n \in D$.
2. There exist no polynomial time Turing machine that when given $A(n)$, will output $B(n)$ for all $n \in \overline{D}$.

This will show that when restricting to either D or $\overline{D}$, we still have $B \not\leq A$. To do this, we will construct a polynomial time Turing machine T that will give us such a set D . Given T , we will let $D = \{n : |T(n)| \text{ is even}\}$. So as long as T is polynomial time, this set can be recognized in polynomial time when given n .

Let $f_0, f_1, f_2, \dots$ be the list of all polynomial time functions that take the natural numbers to binary strings. As $B \not\leq A$, we know that for each f_i and

all $N \in \mathbb{N}$, there exists an $n \in \mathbb{N}, n \geq N$ such that $f_i(A(n)) \neq B(n)$. Otherwise, by modifying f_i we could create a polynomial time function that when given $A(n)$, it would produce $B(n)$. However, knowing that $f_i(A(n)) \neq B(n)$ does not tell us anything about whether or not $n \in D$. The goal of the machine T will be to show that for each f_i , there exists natural numbers $n_1 \in D$ and $n_2 \in \overline{D}$ such that

1. $f_i(A(n_1)) \neq B(n_1)$; and
2. $f_i(A(n_2)) \neq B(n_2)$.

If this is true for all f_i , then when restricting to D or $\overline{D}$, we still have $B \not\leq A$. On input $U(n)$, the machine T will run for $2n$ steps. It will begin by systematically looking for n_1 and n_2 for the function f_0 . Once these have been found, it will then move onto f_1 and so on. To do this, we will define T to be the following machine. Let $T(1) = \emptyset$. For $n \neq 1$, first T will run for n steps systematically constructing $T(1), T(2), \dots$ until it runs out of steps or calculates $T(n-1)$. Let $d(n)$ be the last number for which this sequence is computed. Then T will do one of two things.

1. If $|T(d(n))| = 0 \pmod{2}$, then let $i = |T(d(n))|/2$. For n moves try to find a $z \in D$ such that $f_i(A(z)) \neq B(z)$. If no such z is found after n steps, output $U(2i)$. Alternatively, if such a z is found, output $U(2i+1)$.
2. If $|T(d(n))| = 1 \pmod{2}$, let $i = (|T(d(n))| - 1)/2$. For n moves try to find a $z \in \overline{D}$ such that $f_i(A(z)) \neq B(z)$. If no such z is found after n steps, output $U(2i+1)$. However, if such a z is found, then output $U(2i+2)$.

As $T(n)$ runs for no more than $2n$ steps, this gives us a polynomial time Turing machine for deciding if $n \in D$.

Lemma 6.2.6. $|T(n)| \leq |T(n+1)| \leq |T(n)| + 1$.

Proof. This will be done by induction. We will consider $|T(1)| = |\emptyset| = 0$. Now consider $|T(2)|$. In the first stage, the first move of $T(2)$ will be used to output $T(1)$. It will then stop here as for $n = 2$, $T(1) = T(n-1)$. Now consider the second stage. As $|T(d(n))| = |T(1)| = 0$, $i = 0$ and therefore $T(2)$ will output $2i = 0$ or $2i+1 = 1$. Therefore we see that $|T(1)| \leq |T(2)| \leq |T(1)| + 1$.

Now suppose that this holds for $n \leq k$ and consider $T(k)$ and $T(k+1)$. Note that we have $d(k) \leq d(k+1) \leq k$ and therefore by the induction hypothesis $|T(d(k+1))| \leq |T(k)|$. By the construction, we also have

$$|T(d(k))| \leq |T(k)| \leq |T(d(k))| + 1$$

and

$$|T(d(k+1))| \leq |T(k+1)| \leq |T(d(k+1))| + 1.$$

First suppose that $|T(d(k))| = |T(d(k+1))|$. Then $|T(k)|, |T(k+1)| \in \{|T(d(k))|, |T(d(k))| + 1\}$. So $|T(k+1)| \leq |T(k)| + 1$ and the value of i is the same in both $T(k)$ and $T(k+1)$. Thus in $T(k)$ and $T(k+1)$, we are looking for a suitable z such that $f_i(A(z)) \neq B(z)$ for the same f_i . If we can find such a z in k steps, then we can find such a z in $k+1$ steps. Therefore if $|T(k)| = |T(d(k))| + 1$, then

$$|T(k+1)| = |T(d(k))| + 1 = |T(k)|.$$

Therefore $|T(k)| \leq |T(k+1)|$ and it follows that $|T(k)| \leq |T(k+1)| \leq |T(k)| + 1$.

Now suppose that $|T(d(k))| \neq |T(d(k+1))|$. Then as $d(k) \leq d(k+1) \leq k$, we see that $|T(d(k))| < |T(d(k+1))|$ by the induction hypothesis and therefore $|T(d(k))| + 1 \leq |T(d(k+1))|$. This gives

$$|T(k)| \leq |T(d(k))| + 1 \leq |T(d(k+1))| \leq |T(k+1)|$$

and thus $|T(k)| \leq |T(k+1)|$. Now as $d(k+1) \leq k$, by the induction hypothesis we get $|T(d(k+1))| \leq |T(k)|$. Combining this with the construction of T , we get $|T(k+1)| \leq |T(d(k+1))| + 1 \leq |T(k)| + 1$ and therefore $|T(k)| \leq |T(k+1)| \leq |T(k)| + 1$.

Therefore by induction we see that $|T(n)| \leq |T(n+1)| \leq |T(n)| + 1$. $\square$

Lemma 6.2.7. *If $\{U(2i+1), U(2i+2)\} \in \text{range}(T)$, then there exists a $n_1 \in D$ and a $n_2 \in \overline{D}$ such that $f_i(A(n_1)) \neq B(n_1)$ and $f_i(A(n_2)) \neq B(n_2)$.*

Proof. To begin, assume we have some n such that $|T(n-1)| + 1 = |T(n)| = U(2i+1)$. That is, n is the smallest natural number such that $T(n) = U(2i+1)$. Recall that $|T(d(n))| \leq |T(n)| \leq |T(d(n))| + 1$. Also, note that if $|T(d(n))| \neq |T(n)|$, then there exists some z such that $f_i(A(z)) \neq B(z)$. As $d(n) \leq (n-1)$, we see that $|T(d(n))| \leq |T(n-1)| = |T(n)| - 1$. Thus $|T(d(n))| \neq |T(n)|$ and we see that some z has been found by $T(n)$ such that $f_i(A(z)) \neq B(z)$. As $|T(n)| = U(2i+1)$, we see that $|T(d(n))| = 0 \pmod{2}$ and thus the z found is in D . Therefore if $U(2i+1) \in \text{range}(T)$, then there exists a $n_1 \in D$ such that $f_i(A(n_1)) \neq B(n_1)$.

A similar argument shows that if $U(2i+2) \in \text{range}(T)$, then there exists a $n_2 \in \overline{D}$ such that $f_i(A(n_2)) \neq B(n_2)$. $\square$

This shows that we have a set D that is polynomially time recognizable given $n \in \mathbb{N}$. However it could be the case that $T(B(n))$ is even for all $n \in \mathbb{N}$. The goal of D is to effectively split the problem of creating $B(n)$ given $A(n)$ into two sub problems that are still hard. This will not work if $T(B(n))$ is even for all $n \in \mathbb{N}$ (as the problem would not be ‘split’ into two problems). So to get around this, we create the function C_B that maps $B(n)$ to $\mathbb{N}$ such that $\text{range}(C_B) = \mathbb{N}$. For some function $B \in \mathcal{N}$, let C_B be the polynomial time Turing machine that on input $B(n)$ does the following. If $B(n) = B(1)$, then $C_B(B(n)) = 0$. Otherwise, for $|B(n)|$ steps, C_B will attempt to create $B(1)$ then $B(2)$ and so on until it runs out of steps. Note that $C_B(B(n))$ will always run out of steps before it creates $B(n)$ as it only has $|B(n)|$ steps to work with and it takes $|B(n)|$ steps to write $B(n)$. Let $d(n)$ be the largest number such that $B(d(n))$ was successfully created. If no $B(d(n))$ was successfully created, then $C_B(B(n)) = 0$. Otherwise, note that $|B(1)|, |B(2)|, \dots, |B(d(n))| < |B(n)|$. Now calculate $C_B(B(1)), C_B(B(2)), \dots, C_B(B(d(n)))$ and let $C_B(B(n)) = \max\{C_B(B(1)), \dots, C_B(B(d(n)))\} + 1$. Note that $C_B(B(n))$ runs in no more than $|B(n)|^2$ steps and so runs in polynomial time.

Lemma 6.2.8. $\text{range}(C_B) = \mathbb{N}$.

Proof. First suppose that $m \in \text{range}(C_B)$. Then there exists some n such that $C_B(B(n)) = m$. As $C_B(B(n)) = \max\{C_B(B(1)), \dots, C_B(B(d(n)))\} + 1$, we know there must exist some $n' \in \{1, \dots, d(n)\}$ such that $C_B(B(n')) = m - 1$. Recall that $C_B(B(1)) = 0$. Therefore, if $m \in \text{range}(C_B)$, then so is $\{0, 1, \dots, m-1\}$. So if $\text{range}(C_B)$ is not $\mathbb{N}$, then there must exist some N such that $N = \max\{C_B(B(n))\}$ for all $n \in \mathbb{N}$. As there is no upper bound on the size of $B(n)$, there will exist some n'' such that $|B(n'')|$ is large enough so that $\max\{C_B(B(1)), \dots, C_B(B(d(n'')))\} = N$. But then $C_B(B(n'')) = N + 1 > N$ and we have a contradiction. Thus $\text{range}(C_B) = \mathbb{N}$. $\square$

This gives the required function that takes $B(n)$ to $\mathbb{N}$. As $\text{range}(C_B) = \mathbb{N}$, we can redefine the set D to be $D = \{C_B(B(n)) : |T(C_B(B(n)))| \text{ is even}\}$ without any changes to the above results.

Lemma 6.2.9. Suppose $B \not\leq A$ for $A, B \in \mathcal{N}$. Then there exists a set D that is polynomial time recognizable when given $B(n)$ such that

1. there is no polynomial time Turing machine that given $A(n)$, can produce $B(n)$ for all $n \in D$; and

2. there is no polynomial time Turing machine that given $A(n)$, can produce $B(n)$ for all $n \in \overline{D}$.

Proof. We will show that the set $D = \{C_B(B(n)) : |T(C_B(B(n)))| \text{ is even}\}$ for the Turing machine T described above has the desired property. By Lemma 6.2.7 if $\{U(2i+1), U(2i+2)\} \in \text{range}(T)$, then there exists a $n_1 \in D$ and a $n_2 \in \overline{D}$ such that $f_i(A(n_1)) \neq B(n_1)$ and $f_i(A(n_2)) \neq B(n_2)$. So if the range of T is $\{1^*\}$ then this holds for all f_i . As we have listed all polynomial time functions that take natural numbers to binary strings, if $\text{range}(T) = \{1^*\}$, then this implies that when restricting to either D or $\overline{D}$, we still have $B \not\leq A$.

By Lemma 6.2.6 we have $|T(n)| \leq |T(n+1)| \leq |T(n)| + 1$. Recall that for all N and f_i , there exists $n \geq N$ such that $f_i(A(n)) \neq B(n)$. If the range of T is not $\{1^*\}$, then because $|T(n)| \leq |T(n+1)| \leq |T(n)| + 1$, there exists some N such that for all $n \geq N$, $T(n) = U(j)$ for some j . Suppose $j = 0 \pmod{2}$. Then for all $n \geq N$, $n \in D$. Furthermore there exists no $n_1 \in D$ such that $f_i(A(n_1)) \neq B(n_1)$. However, this would then imply that there is no $n \geq N$ such that $f_i(A(n)) \neq B(n)$. This is a contradiction as we know that $B \not\leq A$ and thus such an $n_1 \in D$ must exist for f_i .

Now suppose $j = 1 \pmod{2}$. Then there exists no $n_2 \in \overline{D}$ such that $f_i(A(n_2)) \neq B(n_2)$. Then for all $n \geq N$, $n \notin D$ and therefore $n \in \overline{D}$. But then we have no $n \geq N$ such that $f_i(A(n)) \neq B(n)$ which again is a contradiction as we know that $B \not\leq A$.

Therefore the range of T is $\{1^*\}$ and as C_B and T are both polynomial time, it follows that we have a polynomial time recognizable set D such that

1. there is no polynomial time Turing machine that given $A(n)$, can produce $B(n)$ for all $n \in D$; and
2. there is no polynomial time Turing machine that given $A(n)$, can produce $B(n)$ for all $n \in \overline{D}$.

□

Corollary 6.2.10. *Suppose we have $A < B$ for $A, B \in \mathcal{N}$. Then there exist functions $C_1 \in \mathcal{N}$ and $C_2 \in \mathcal{N}$ such that $A < C_i < B$.*

Proof. As $A < B$, we have $B \not\leq A$. So let D be as in Lemma 6.2.9. Define $C_1(n) = E(A(n))010B(n)$ if $C_B(B(n)) \in D$ and $C_1(n) = E(A(n))$ otherwise. Similarly, let $C_2(n) = E(A(n))$ if $C_B(B(n)) \in D$ and $C_2(n) = E(A(n))010B(n)$

otherwise. Note that $C_i \in \mathcal{N}$. As D is polynomial time recognizable given $B(n)$, we see that $C_i \leq B$. Furthermore, from Lemma 6.2.9 we see that $B \not\leq C_i$ which implies that $C_i < B$. By the construction of C_i , we have $A \leq C_i$. Moreover, again from Lemma 6.2.9 we see that $C_i \not\leq A$. Thus $A < C_i < B$. $\square$

Corollary 6.2.10 shows that $\mathcal{N}$ is dense, proving Theorem 6.1.2.

Let L be a lattice with joins (least upper bound) denoted by $\vee$ and meets (greatest lower bound) denoted by $\wedge$. Then the lattice L is distributive if

$$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$$

for all $A, B, C \in L$. We now move on to the final property we are interested in. We will show that any countable distributive lattice can be embedded in $\mathcal{N}$. To do this, we make use of the following function. Define $S(n)$ to be a string of length n where the i th bit is a 1 if and only if $i \in H$. Note that $S(n)$ cannot be computed in polynomial time as this would give a method of determining membership of H in polynomial time. Let B be a finite binary string and define B^* to be the string B repeated an infinite number of times. We will use B^* as the characteristic function for subsets N of the natural numbers where $n \in N$ if and only if the n th bit of B^* is a 1. Because of this, we will often refer to B^* as a set even though strictly speaking it is only the characteristic function. We will call such a subset a *cyclic subset*.

Lemma 6.2.11. *Let B be some non-zero binary string. Then there exists no polynomial Turing machine that will construct $S(n)$ for all $n \in B^*$.*

Proof. Suppose there is such an algorithm $\mathcal{A}$ that when given $m \in B^*$ in unary, will output $S(m)$ in polynomial time. Now suppose we are given some n and asked to construct $S(n)$. If $n \in B^*$, then simply apply $\mathcal{A}$ to it and we will have $S(n)$. So assume that $n \notin B^*$ and suppose B is a binary string of k bits. Then as B^* is a cyclic subset, there will be an element $n' \in B^*$ such that $0 < n' - n \leq k$. Thus $U(n')$ is polynomial in size of $U(n)$. We can then apply $\mathcal{A}$ to n' to make $S(n')$. This will produce a string of length n' where the first n bits will be $S(n)$. Thus such an algorithm cannot exist for any non-zero cyclic subset. $\square$

A *Boolean algebra* is a complemented distributed lattice with a minimum and maximum element. That is, a distributive lattice L such that for all $A \in L$, there exists a B such that $A \vee B = 0$ and $A \wedge B = 1$ where 0 and 1 are the minimum and

maximum elements respectively. We denote this by $B = \neg A$. We say that A is an atom of L if $0 < A$ and there does not exist any $B \in L$ such that $0 < B < A$. A lattice is *atomless* if it contains no atom. Any countable distributive lattice can be embedded in the (unique up to isomorphism) countable atomless boolean algebra [73]. Next we show that $\mathcal{N}$ contains the countable atomless boolean algebra. This will show that any countable distributive lattice can be embedded in $\mathcal{N}$. We can construct the countable atomless boolean algebra in the following method [73]. The classes B^* form a countable atomless boolean algebra under subset inclusion when we treat the strings B^* as the characteristic function of the set. There will be one such set for each natural number. Thus there is a countable number of them. To see that there is no atom, take the binary string B . Let B' be the string B followed by $|B|$ zeros. Then we see that $0 < B'^* < B^*$. Therefore there is no atom. The maximum element is given by the binary string $B_1 = 1$ while the minimum is given by $B_0 = 0$.

We will define $B \vee B'$ to be the binary string of length $\text{lcm}\{|B|, |B'|\}$ where the i th bit is a 1 if and only if $i \in B^*$ or B'^* . This gives $i \in (B \vee B')^*$ if and only if $i \in B^*$ or B'^* . We will define $B \wedge B'$ in a similar fashion. That is, $B \wedge B'$ is a binary string of length $\text{lcm}\{|B|, |B'|\}$ where the i th bit is a 1 if and only if i is in both B^* and B'^* . This gives $i \in (B \wedge B')^*$ if and only if $i \in B^*$ and $i \in B'^*$. We then have $B^* \vee B'^* = (B \vee B')^*$ and $B^* \wedge B'^* = (B \wedge B')^*$.

We now have the tools required to prove Theorem 6.1.3.

Proof. To show this, all we need to do is show that we can embed the countable atomless boolean algebra we constructed above. Take some function f in $\mathcal{N}$ that can be constructed in polynomial time given $U(n)$ and given $f(n)$, we can recover n in polynomial time. Let $f'(n) = E(f(n))010S(n)$. For $B \neq 0$, create the functions f_B where

$$f_B(n) = \begin{cases} f'(n) & \text{if } n \in B^* \\ E(f(n)) & \text{otherwise.} \end{cases}$$

We claim that the mapping $\phi(B^*) = f_B$ is an isomorphism and thus $\mathcal{N}$ contains the countable atomless boolean algebra. Suppose we have $B'^* < B^*$ and we are given $f_B(n)$ where $n \in B^*$ but $n \notin B'^*$. To create $f_{B'}(n)$, we simply remove the string $S(n)$ from $f_B(n)$. Given $f_{B'}(n) = f(n)$, to create $f_B(n)$, we would need to be able to construct $S(n)$. However, the set of elements in $B^* - B'^*$ is a cyclic subset. Therefore by Lemma 6.2.11 there is no polynomial time algorithm that

can construct $S(n)$ for all $n \in B^* - B'^*$. Assume we have a polynomial algorithm $\mathcal{A}$ that given input $f_{B'}(n)$, it will output $f_B(n)$. As $f(n)$ can be constructed in polynomial time from n we could use this algorithm to create $S(n)$ for any n in $B^* - B'^*$. Simply construct $E(f(n)) = f_{B'}(n)$ in polynomial time for the given $n \in B^* - B'^*$ and then use $\mathcal{A}$ to create $f_B(n)$. This is a contradiction and therefore $\mathcal{A}$ cannot exist. Thus $f_{B'} < f_B$.

From the above argument, we see that the maximum element is f_1 and the minimum element is $f_0 = E(f)$.

Using similar reasoning, we can see that $f_B \vee f_{B'} = f_{B \vee B'}$. This is because from above we see that $f_B \leq f_{B \vee B'}$ and $f_{B'} \leq f_{B \vee B'}$ (as $B - B'$ and $B' - B$ are both cyclic subsets). Now given $f_B \vee f_{B'}(n)$ we can compute $f_B(n)$ and $f_{B'}(n)$ and therefore can compute $f_{B \vee B'}(n)$. Thus $f_{B \vee B'} \leq f_B \vee f_{B'}$. Therefore $f_B \vee f_{B'} = f_{B \vee B'}$.

Now consider $f_B \wedge f_{B'}$. We know that $f_{B \wedge B'} \leq f_B$ and $f_{B \wedge B'} \leq f_{B'}$. Now suppose we are given $f_{B \wedge B'}(n)$ and consider $f_{B''}$ where $f_{B''} = f_B \wedge f_{B'}$. As $f_{B''} \leq f_B$, we know that if $i \in B''^*$ then $i \in B^*$. Similarly, if $i \in B''^*$, then $i \in B'^*$ and therefore if $i \in B''^*$ then $i \in B^* \wedge B'^*$. It follows that $f_B \wedge f_{B'} \leq f_{B \wedge B'}$ and therefore $f_B \wedge f_{B'} = f_{B \wedge B'}$.

Now if $\neg f_B = f_{B''}$, then $f_B \vee f_{B''} = f_1$ and $f_B \wedge f_{B''} = f_0$. This can only happen if $B'' = \neg B$.

Therefore the functions f_B form a countable atomless boolean algebra. As any countable distributive lattice can be embedded in the countable atomless boolean algebra, any countable distributive lattice can be embedded in $\mathcal{N}$. $\square$

6.3 The structure of $\mathcal{M}$

We now restrict to only functions that describe matroids. Recall that we have a computable function that takes matroids to the natural numbers. To restrict to just matroid descriptions, we add the following restrictions to our succinct injective functions. We require that our functions describe a matroid and that given $f(n)$ for the matroid n , we can simulate an independence oracle in polynomial time. All reasonable matroid descriptions will have this property. So we will define $\mathcal{M}$ to be the set of all succinct injective functions $f : \mathbb{N} \rightarrow \Sigma^*$ such that there exists a finite Turing machine that on input $U(n)$ will output $f(n)$ for all $n \in \mathbb{N}$ and that given $f(n)$ for the matroid n , we can simulate an independence oracle for n in polynomial time. Note that $\mathcal{M} \subset \mathcal{N}$.

When we restrict to $\mathcal{M}$, we see that Lemma 6.2.5 and Corollary 6.2.10 remain true without change. As for the proofs of Theorem 6.1.3 and Lemma 6.2.4, consider the following alterations in Lemmas 6.3.1 and 6.3.2.

For the matroid with k loops and n elements, by the ordering used to create the computable function from the matroids to natural numbers, it will be the case that the elements $1, 2, \dots, n - k$ are not loops and the elements $n - k + 1, \dots, n$ are loops. Recall that the ordering of matroids induced by their correspondence with the natural numbers is such that $M_1 < M_2$ if $|E(M_1)| < |E(M_2)|$.

Let I be the following matroid description. For the matroid n with ground set $E(M) = \{1, 2, \dots, m\}$, $I(n)$ will be a string of length 2^m where the i th bit corresponds to the i th subset of $E(M)$ when ordered lexicographically and the i th bit is 1 if and only if the i th subset is independent. This description is succinct as the matroid n will have less than 2^n subsets.

Lemma 6.3.1. *Not every pair of functions in $\mathcal{M}$ have a meet.*

Proof. Recall the functions f_1 and f_2 used in Lemma 6.2.4 and consider the following matroid descriptions h_1 and h_2

$$h_1(n) = \begin{cases} 010E(B(m_1))010f_1(m_2) & \text{if } n = U_{m_1, m_1} \oplus_{m_2} \text{ loops} \\ E(I(n)) & \text{otherwise} \end{cases}$$

and

$$h_2(n) = \begin{cases} 010E(B(m_1))010f_2(m_2) & \text{if } n = U_{m_1, m_1} \oplus_{m_2} \text{ loops} \\ E(I(n)) & \text{otherwise} \end{cases}$$

where $B(n)$ is the binary representation of n . If $h_1(n) = E(I(n))$, then we can simulate an independence oracle as $I(n)$ can. Furthermore, if $h_1 \neq I(n)$, then we know a subset is independent if and only if it is a subset of $\{1, 2, \dots, m_1\}$. Similarly for h_2 . Thus these two functions can simulate an independence oracle in polynomial time. Furthermore, they are injective as I is injective and there can be no collisions between $E(I(n))$ and $010E(B(m_1))010f_1(m_2)$ because the first bit of $E(I(n))$ is always a 1 as the empty set is always independent. These functions are also succinct. Thus h_1 and h_2 are in $\mathcal{M}$.

Suppose these two functions have a meet M . Then there exists two polynomial time functions F_1 and F_2 such that $F_1(h_1(n)) = M(n) = F_2(h_2(n))$. Using the

same argument as in Lemma 6.2.4, we see that

$$F_1(010E(B(m_1))010p(B(m_2))) = F_2(010E(B(m_1))010B(m_2))$$

if and only if $|B(m_2)| \in H$. Now suppose we are given some n and asked if $n \in H$. Then to decide this, all we need to do is construct $010E(B(m_1))010B(m_2)$ and $010E(B(m_1))010p(B(m_2))$ for some integer m_2 such that $|B(m_2)| = n$. Note that any such binary string will be the description of some matroid. Then to decide if $n \in H$, simply test if

$$F_1(010E(B(m_1))010p(B(m_2))) = F_2(010E(B(m_1))010B(m_2)).$$

If so, then we know $n \in H$. Otherwise, $n \notin H$. Therefore F_1 and F_2 cannot exist and thus h_1 and h_2 do not have a meet. $\square$

Lemma 6.3.2. *Let L be any countable distributive lattice. Then L can be embedded in $\mathcal{M}$.*

Proof. Let B^* be some cyclic subset and $B(n)$ be the binary representation of the natural number n . Then define the function f_B to be

$$f_B(n) = \begin{cases} 010E(B(m))010S(m) & \text{if } n = U_{m,m} \text{ and } m \in B^* \\ 010E(B(m)) & \text{if } n = U_{m,m} \text{ and } m \notin B^* \\ E(I(n)) & \text{otherwise.} \end{cases}$$

Note that given m , we can easily create $010E(B(m))$ in polynomial time. Thus, by the same argument as used in the proof of Theorem 6.1.3, we see that any countable infinite lattice can be embedded in $\mathcal{M}$. This is because, given any $m \in B$ in unary, we can construct $010E(B(m)) = f(n)$ for some n in polynomial time. We will not know what n is, but this does not matter as for each $010E(B(m))$ there exists some n such that $010E(B(m)) = f(n)$. So if we have an algorithm that takes $f(n)$ to $f_B(n)$ we can just apply it to $010E(B(m))$ for the given $m \in B$ and this will give us $S(m)$. $\square$

So we see that when we restrict to $\mathcal{M}$, we still have a dense semi-lattice such that any countable distributive lattice can be embedded in it. Thus Theorems 6.1.1, 6.1.2, and 6.1.3 still hold when restricting to $\mathcal{M}$.

Chapter 7

Future Work

We have seen in chapters 2 and 3 that a large number of deletion problems are NP-hard. Usually the next step when one finds a problem is NP-hard is to consider fixed-parameter tractability or kernelization. We note that certain deletion problems are already known to be fixed-parameter tractable. For example, the problem ODD CYCLE TRANSVERSAL is a deletion problem where we are trying to obtain a bipartite graph. This has been shown to be fixed-parameter tractable by fixing the number of deletions allowed [61]. Furthermore, a recent result has provided a polynomial kernel [45]. Moreover, one of the poster boys for fixed-parameter tractability, the vertex cover problem, can be restated as a deletion problem. When we do this, we see that the vertex cover problem is fixed-parameter tractable by fixing the number of deletions allowed. This gives the impression that the deletion problems considered in Chapters 2 and 3 may well be fixed-parameter tractable. A likely candidate would be the number of deletion allowed. In the matroid case, one could also likely fix the corank of the matroids in question. However, because the properties considered were non-trivial, this is just artificially fixing the number of deletions allowed.

While nobody is doubting the result by Vertigan that it is $\#P$ -complete to count bases of matroids representable over any field, it would be nice to have a proof written down. Chapter 4 provides explicit proofs for special cases of Vertigan's result. Clearly the future goal would be to provide a written proof for the entire result.

It would be of interest to expand the results of Chapter 6 into other areas of computation. In particular, whether or not we can apply kernelization techniques to the inputs in the given hierarchy. With the number of matroids being doubly

exponential in the size of their ground set, it seems unlikely that a polynomially sized kernel will exist. However, even finding non-polynomial kernels would be interesting.

Bibliography

- [1] *Open problem forum of the egyptian research group*. http://lemon.cs.elte.hu/egres/open/Partial_vertex_cover_in_bipartite_graphs.
- [2] FATIMA AFFIF CHAOUCHE, CARRIE RUTHERFORD, AND ROBIN WHITTY, *Objective functions with redundant domains*, Journal of Combinatorial Optimization, (2012), pp. 1–13.
- [3] N. ALON, A. SHAPIRA, AND B. SUDAKOV, *Additive approximation for edge-deletion problems*, in 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05), Pittsburgh, PA, USA, pp. 419–428.
- [4] KLAUS AMBOS-SPIES, *Sublattices of the polynomial time degrees*, Information and Control, 65 (1985), pp. 63–84.
- [5] MATTHEW AMY, DMITRI MASLOV, AND MICHELE MOSCA, *Polynomial-time t -depth optimization of clifford+ t circuits via matroid partitioning*. Preprint, arXiv:1303.2042v1, 2013.
- [6] N. APOLLONIO AND B. SIMEONE, *The maximum vertex coverage problem on bipartite graphs*. preprint.
- [7] SANJEEV ARORA AND BOAZ BARAK, *Computational Complexity: A Modern Approach*, Cambridge University Press, New York, NY, USA, 1st ed., 2009.
- [8] TAKAO ASANO, *An application of duality to edge-deletion problems*, SIAM Journal on Computing, 16 (1987), pp. 312–331.
- [9] TAKAO ASANO AND TOMIO HIRATA, *Edge-deletion and edge-contraction problems*, in Proceedings of the fourteenth annual ACM symposium on Theory of computing, San Francisco, California, United States, 1982, ACM, pp. 245–254.

- [10] VERA ASODI AND CHRISTOPHER UMANS, *The complexity of the matroid/greedoid partition problem*, Theor. Comput. Sci., 410 (2009), pp. 859–866.
- [11] ANDRÉ BERGER, VINCENZO BONIFACI, FABRIZIO GRANDONI, AND GUIDO SCHÄFER, *Budgeted matching and budgeted matroid intersection via the gasoline puzzle*, in Proceedings of the 13th international conference on Integer programming and combinatorial optimization, IPCO'08, Berlin, Heidelberg, 2008, Springer-Verlag, pp. 273–287.
- [12] E. BERLEKAMP, R. MCELIECE, AND H. VAN TILBORG, *On the inherent intractability of certain coding problems*, Information Theory, IEEE Transactions on, 24 (1978), pp. 384–386.
- [13] ANDREAS BJÖRKLUND, THORE HUSFELDT, PETTERI KASKI, AND MIKKO KOIVISTO, *Computing the tutte polynomial in vertex-exponential time*, in Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science, FOCS '08, 2008, pp. 677–686.
- [14] JOSEPH BONIN AND ANNA DE MIER, *The lattice of cyclic flats of a matroid*, Annals of Combinatorics, 12 (2008), pp. 155–170.
- [15] JOEL V. BRAWLEY AND GEORGE E. SCHNIBBEN, *Infinite Algebraic Extensions of Finite Fields*, American Mathematical Soc., 1989.
- [16] DANIEL BRGMANN, CHRISTIAN KOMUSIEWICZ, AND HANNES MOSER, *On generating triangle-free graphs.*, Electronic Notes in Discrete Mathematics, 32 (2009), pp. 51–58.
- [17] PABLO BURZYN, FLAVIA BONOMO, AND GUILLERMO DURN, *NP-completeness results for edge modification problems*, Discrete Applied Mathematics, 154 (2006), pp. 1824–1844.
- [18] BUGRA CASKURLU AND K. SUBRAMANI, *On partial vertex cover on bipartite graphs and trees*, CoRR, abs/1304.5934 (2013).
- [19] DANNY Z. CHEN, JIAN LI, HONGYU LIANG, AND HAITAO WANG, *Matroid and knapsack center problems*, in Proceedings of the 16th international conference on Integer Programming and Combinatorial Optimization, IPCO'13, Springer-Verlag, 2013, pp. 110–122.

- [20] CHARLES J. COLBOURN AND EHAB S. ELMALLAH, *Reliable assignments of processors to tasks and factoring on matroids.*, Discrete Mathematics, 114 (1993), pp. 115–129.
- [21] CHARLES J. COLBOURN, J. SCOTT PROVAN, AND DIRK VERTIGAN, *The complexity of computing the Tutte polynomial on transversal matroids*, Combinatorica, (1995).
- [22] NADIA CREIGNOU AND MIKI HERMANN, *On $\#P$ completeness of some counting problems*, Research Report RR-2144, INRIA, 1993.
- [23] PETER DAMASCHKE, MER EGECIOGLU, AND LEONID MOLOKOV, *Fixed-parameter tractability of error correction in graphical linear systems.*, in WALCOM, Subir Kumar Ghosh and Takeshi Tokuyama, eds., vol. 7748 of Lecture Notes in Computer Science, Springer, 2013, pp. 245–256.
- [24] RODNEY G. DOWNEY AND MICHAEL R. FELLOWS, *Parameterized Complexity*, Springer-Verlag, 1999.
- [25] G E FARR AND C J H MCDIARMID, *The complexity of counting homeomorphs*, Theoretical Computer Science, (1985), pp. 345–348.
- [26] TAMÁS FLEINER AND NAOYUKI KAMIYAMA, *A matroid approach to stable matchings with lower quotas*, in Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '12, SIAM, 2012, pp. 135–142.
- [27] J. FLUM AND M. GROHE, *Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
- [28] MICHAEL GAREY AND DAVID JOHNSON, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, New York, 1979.
- [29] M. R. GAREY AND D. S. JOHNSON, *The rectilinear steiner tree problem is np-complete*, SIAM Journal on Applied Mathematics, 32 (1977), pp. 826–834.
- [30] SERGE GASPERS, MIKKO KOIVISTO, MATHIEU LIEDLOFF, SEBASTIAN ORDYNYIAK, AND STEFAN SZEIDER, *On finding optimal polytrees*, CoRR, abs/1208.1692 (2012).

- [31] OMER GIMÉNEZ AND MARC NOY, *On the complexity of computing the Tutte polynomial of bicircular matroids*, Comb. Probab. Comput., 15 (2006), pp. 385–395.
- [32] EMERIC GIOAN AND MICHEL LAS VERGNAS, *On the evaluation at (j, j^2) of the tutte polynomial of a ternary matroid*, J. Algebraic Comb., 25 (2007), pp. 1–6.
- [33] LESLIE ANN GOLDBERG AND MARK JERRUM, *Inapproximability of the tutte polynomial*, Information and Computation, 206 (2008), pp. 908–929.
- [34] ———, *Inapproximability of the tutte polynomial of a planar graph*, CoRR, abs/0907.1724 (2009).
- [35] ———, *Approximating the partition function of the ferromagnetic potts model*, J. ACM, 59 (2012), pp. 25:1–25:31.
- [36] ———, *Approximating the tutte polynomial of a binary matroid and other related combinatorial polynomials*, J. Comput. Syst. Sci., 79 (2013), pp. 68–78.
- [37] JIONG GUO, *Problem kernels for np -complete edge deletion problems: split and related graphs*, in Proceedings of the 18th international conference on Algorithms and computation, ISAAC’07, Berlin, Heidelberg, 2007, Springer-Verlag, pp. 915–926. ACM ID: 1781675.
- [38] NICHOLAS J. A. HARVEY, *Algebraic algorithms for matching and matroid problems*, SIAM J. Comput., (2009).
- [39] PETR HLINENY, *Some hard problems on matroid spikes*, Theor. Comp. Sys., 41 (2007), pp. 551–562.
- [40] FRANOIS JAEGER, *Tutte polynomials and bicycle dimension of ternary matroids*, Proceedings of the American Mathematical Society, 107 (1989), pp. pp. 17–25.
- [41] F. JAEGER, D. L. VERTIGAN, AND D. J. A. WELSH, *On the computational complexity of the jones and tutte polynomials*, Math. Proc. Camb. Phil. Soc., (1990).
- [42] PER M. JENSEN AND BERNHARD KORTE, *Complexity of matroid property algorithms*, SIAM J. Comput., 11 (1982), pp. 184–190.

- [43] M. JERRUM, *Two remarks concerning balanced matroids*, *Combinatorica*, 26 (2006).
- [44] GWENAËL JORET AND ADRIAN VETTA, *Reducing the rank of a matroid*, *CoRR*, abs/1211.4853 (2012).
- [45] STEFAN KRATSCH AND MAGNUS WAHLSTRÖM, *Compression via matroids: a randomized polynomial kernel for odd cycle transversal*, in *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '12*, SIAM, 2012, pp. 94–103.
- [46] STEFAN KRATSCH AND MAGNUS WAHLSTROM, *Representative sets and irrelevant vertices: New tools for kernelization*, in *Proceedings of the 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science, FOCS '12*, Washington, DC, USA, 2012, IEEE Computer Society, pp. 450–459.
- [47] A. H. LACHLAN, *Lower bounds for pairs of recursively enumerable degrees*, *Proc. London Math. Soc.*, 16 (1966), pp. 537–569.
- [48] RICHARD E. LADNER, *On the structure of polynomial time reducibility*, *J. ACM*, 22 (1975).
- [49] EUGENE L. LAWLER, *Matroid intersection algorithms*, *Mathematical Programming*, 9 (1975), pp. 33–56.
- [50] JON LEE, MAXIM SVIRIDENKO, AND JAN VONDRAK, *Matroid matching: the power of local search*, in *Proceedings of the 42nd ACM symposium on Theory of computing, STOC '10*, New York, NY, USA, 2010, ACM, pp. 369–378.
- [51] JOHN M. LEWIS AND MIHALIS YANNAKAKIS, *The node-deletion problem for hereditary properties is np-complete*, *J. Comput. Syst. Sci.*, 20 (1980), pp. 219–230.
- [52] L. LOVSZ, *The matroid matching problem*, *Algebraic methods in graph theory*, 1 (1981), pp. 495–517.
- [53] DÁNIEL MARX, *A parameterized view on matroid optimization problems*, *Theor. Comput. Sci.*, 410 (2009), pp. 4471–4479.
- [54] DILLON MAYHEW, *Matroids and Complexity*, PhD thesis, University of Oxford, 2005.

- [55] DILLON MAYHEW, *Matroid complexity and nonsuccinct descriptions*, SIAM J. Discret. Math., 22 (2008), pp. 455–466.
- [56] DILLON MAYHEW AND GORDON F. ROYLE, *Matroids with nine elements*, Journal of Combinatorial Theory, Series B, 98 (2008), pp. 415–431.
- [57] ASSAF NATANZON, RON SHAMIR, AND RODED SHARAN, *Complexity classification of some edge modification problems*, in Proceedings of the 25th International Workshop on Graph-Theoretic Concepts in Computer Science, WG '99, London, UK, 1999, Springer-Verlag, pp. 65,77.
- [58] J. G. OXLEY, *Matroid Theory*, Oxford University Press, 2 ed., 2011.
- [59] R. A. PENDAVINGH, *On the evaluation at $(-i, i)$ of the tutte polynomial of a binary matroid*, Journal of Algebraic Combinatorics, (2013).
- [60] RAGHAVENDRA RAO B.V. AND JAYALAL SARMA M.N., *On the complexity of matroid isomorphism problems*, in Proceedings of the Fourth International Computer Science Symposium in Russia on Computer Science - Theory and Applications, CSR '09, Springer-Verlag, 2009, pp. 286–298.
- [61] BRUCE REED, KALEIGH SMITH, AND ADRIAN VETTA, *Finding odd cycle transversals*, Oper. Res. Lett., 32 (2004), pp. 299–301.
- [62] JOACHIM REICHEL AND MARTIN SKUTELLA, *Evolutionary algorithms and matroid optimization problems*, in Proceedings of the 9th annual conference on Genetic and evolutionary computation, GECCO '07, New York, NY, USA, 2007, ACM, pp. 947–954.
- [63] G. C. ROBINSON AND D. J. A. WELSH, *The computational complexity of matroid properties*, Mathematical Proceedings of the Cambridge Philosophical Society, 87 (1980), pp. 29–45.
- [64] P.D. SEYMOUR, *Recognizing graphic matroids*, Combinatorica, 1 (1981), pp. 75–78.
- [65] PD SEYMOUR AND PN WALTON, *Detecting matroid minors*, Journal of the London Mathematical Society, 2 (1981), p. 193.
- [66] P. D. SEYMOUR, *Recognizing graphic matroids*, Combinatorica, 1 (1981), pp. 75–78.

- [67] JOSÉ A. SOTO, *A simple ptas for weighted matroid matching on strongly base orderable matroids*, CoRR, abs/1102.3491 (2011).
- [68] CLIFFORD SPECTOR, *On degrees of recursive unsolvability*, Annals of Mathematics, 64 (1956), pp. pp. 581–592.
- [69] S. K. THOMASON, *Sublattices of the recursively enumerable degrees*, Z. Math. Logik Grundlag. Math, 17 (1971), pp. 273–280.
- [70] LESLIE G VALIANT, *The complexity of computing the permanent*, Theoretical Computer Science, 8 (1979).
- [71] ———, *The complexity of enumeration and reliability problems*, SIAM Journal on Computing, 8 (1979), pp. 410–421.
- [72] ALEXANDER VARDY, *Algorithmic complexity in coding theory and the minimum distance problem*, in Proceedings of the twenty-ninth annual ACM symposium on Theory of computing, STOC '97, New York, NY, USA, 1997, ACM, pp. 92–109.
- [73] S. VERMEEREN, *Embeddings into the countable atomless boolean algebra*. <http://stijnvermeeren.be/download/mathematics/embeddings.pdf>.
- [74] DIRK VERTIGAN, *The computational complexity of tutte invariants for planar graphs*, SIAM J. Comput., 35 (2005), pp. 690–712.
- [75] DIRK VERTIGAN AND GEOFF WHITTLE, *Recognizing polymatroids associated with hypergraphs*, Combinatorics, Probability and Computing, 2 (1993), pp. 519–530.
- [76] DIRK L. VERTIGAN, *Bicycle dimension and special points of the tutte polynomial*, J. Comb. Theory, Ser. B, 74 (1998).
- [77] TOSHIMASA WATANABE, TADASHI AE, AND AKIRA NAKAMURA, *On the removal of forbidden graphs by edge-deletion or by edge-contraction*, Discrete Applied Mathematics, 3 (1981), pp. 151–153.
- [78] ———, *On the np-hardness of edge-deletion and -contraction problems*, Discrete Applied Mathematics, 6 (1983), pp. 63–78.
- [79] DOMINIC WELSH, *Complexity: Knots, Colourings and Counting*, Cambridge University Press, 1993.

- [80] MIHALIS YANNAKAKIS, *Node-and edge-deletion np-complete problems*, in STOC '78: Proceedings of the tenth annual ACM symposium on Theory of computing, New York, NY, USA, 1978, ACM, pp. 253–264.
- [81] ———, *Edge-deletion problems.*, SIAM J. Comput., 10 (1981), pp. 297–309.
- [82] C. E. M. YATES, *A minimal pair of recursively enumerable degrees*, J. Symbolic Logic, 2 (1966), pp. 159–168.